Current trends in deterministic scheduling

Chung-Yee Lee
Department of Industrial Engineering, Texas A&M University, College Station, TX 77843-3131, USA

Lei Lei
Graduate School of Management, Rutgers University, Newark, NJ 07102, USA

Michael Pinedo
Department of Industrial Engineering and Operations Research, Columbia University, New York, NY 10027-6699, USA

Scheduling is concerned with allocating limited resources to tasks to optimize certain objective functions. Due to the popularity of the Total Quality Management concept, on-time delivery of jobs has become one of the crucial factors for customer satisfaction. Scheduling plays an important role in achieving this goal. Recent developments in scheduling theory have focused on extending the models to include more practical constraints. Furthermore, due to the complexity studies conducted during the last two decades, it is now widely understood that most practical problems are NP-hard. This is one of the reasons why local search methods have been studied so extensively during the last decade. In this paper, we review briefly some of the recent extensions of scheduling theory, the recent developments in local search techniques and the new developments of scheduling in practice. Particularly, we survey two recent extensions of theory: scheduling with a 1-job-on-*r*-machine pattern and machine scheduling with availability constraints. We also review several local search techniques, including simulated annealing, tabu search, genetic algorithms and constraint guided heuristic search. Finally, we study the robotic cell scheduling problem, the automated guided vehicles scheduling problem, and the hoist scheduling problem.

0. Introduction

Scheduling is concerned with allocating limited resources to tasks to optimize certain objective functions. Due to the popularity of the Total Quality Management concept, on-time delivery of jobs has become one of the crucial factors for customer satisfaction. Scheduling plays an important role in achieving this goal. In the last four decades, many papers have been published in the scheduling area (for example, see the work by Graves (1981), Lawler et al. (1993), Herrmann, Lee and Snowdon (1993),

Morton and Pentico (1993), Blazewicz et al. (1994, 1996), Tanaev, Gordon and Shafransky (1994), and Tanaev, Sotskov and Strusevich (1994), Baker (1995), Brucker (1995), and Pinedo (1995)).

In deterministic scheduling models, a set of jobs has to be processed by a set of machines and certain performance measures have to be optimized. Recent developments in scheduling theory have focused on extending the models to include more practical constraints. Furthermore, due to the complexity studies conducted during the last two decades, it is now widely understood that most practical problems are NP-hard. This is one of the reasons why local search methods have been studied so extensively during the last decade.

In this paper, we review briefly some of the recent extensions of scheduling theory (section 1), the recent developments in local search techniques (section 2) and the new developments of scheduling in practice (section 3).

To refer to any given problem in a concise way, we follow the notation of Pinedo (1995). This notation consists of three fields $\alpha|\beta|\gamma$, where α denotes the machine (resource) configuration, β denotes processing restrictions and constraints, and γ represents the performance measure to be optimized. In several problems studied in this paper, we will describe the problem by adding special constraints to the second field.

1. Recent developments in scheduling theory

The new trend in scheduling theory is to extend results of classical algorithms to models that are more closely related to real problems. Even though many results may not be applicable immediately, these new models are at least motivated by industrial problems and have a greater potential for applications. Along this line, several topics have been reviewed in the literature (see for example, Baker and Scudder (1990) about sequencing with earliness and tardiness penalties, Potts and Van Wassenhove (1992) about integrating scheduling with batching and lot-sizing, Trietsch and Baker (1993) about lot streaming, Lee and Variraktarakis (1993) about hierarchical scheduling, Webster and Baker (1995) about scheduling groups of jobs on a single machine and Hall and Sriskandarajah (1996) about scheduling with blocking and no-wait in process).

In this section, we review briefly two other deterministic scheduling areas that have not received much attention until the last decade. These two areas are (1) scheduling with a 1-job-on-r-machine pattern, and (2) machine scheduling with availability constraints. In (1), jobs may need to be processed simultaneously on several machines (r is a positive integer) or several jobs can be processed by a single processor simultaneously ($0 < r \leq 1$), in contrast to classical scheduling where one machine can process one job and each job can be processed by one machine at any time ($r = 1$). In (2), machines may not be available at all times, due to machine maintenance, or machines may only be available in given time windows as in applications of computing systems.

In order to describe the problem more precisely, we introduce the following notation. There are n jobs to be processed on m machines, and

J_j : job j, $j = 1,\ldots,n$,
M_i : machine i, $i = 1,\ldots,m$,
C_j : the completion time for J_j,
w_j : the weight for J_j,
d_j : the due date of J_j,
L_j : the lateness of $J_j = C_j - d_j$,
L_{max} : maximum lateness = $\max\{L_j, j = 1,\ldots,n\}$,
C_{max} : makespan = $\max\{C_j: j = 1,\ldots,n\}$.

1.1. Scheduling with 1-job-on-r-machine pattern

In the last four decades, numerous papers have been published in many different scheduling areas. However, most of the scheduling literature assumes a 1-job-on-1-machine pattern. Namely, at each point of time a job can be processed on one and only one machine. With the rapid development of parallel computing systems, such as diagnosable microprocessor systems (Krawczyk and Kubale (1985)), a job must be performed on parallel processors in order to detect faults. In such a system, a job may need to be processed simultaneously by several processors (1-job-on-r-machine problem, where r is a positive integer). Semiconductor circuit design team workforce planning, where a modular task can be worked on by an entire team simultaneously, is also an example. Another example is the berth allocation problem where a large vessel may occupy several berths for loading and/or unloading (r is a positive integer) (Lee and Cai (1996)), or several vessels may share one berth ($0 < r \leq 1$ and r is a real number) (Li, Cai and Lee (1996)).

The case where r is a positive integer has been studied extensively over the last decade. This type of a 1-job-on-r-machine problem has been called a *multiprocessor task system* in the literature and will be one of the main topics in this review. There are two main classes of such scheduling problems. The first one assumes that each job may require a fixed *number* of machines working simultaneously, yet the machines required are not specified (Blazewicz, Drabowski and Weglarz (1986), Du and Leung (1989), Plehn (1990), and Lee and Cai (1996)). The second class of problems specifically fixes the *set* of machines for particular jobs (Kubale (1987), Blazewicz et al. (1992), Bianco et al. (1993), Hoogeveen, van de Velde and Veltman (1994), Bianco et al. (1995), Cai, Lee and Li (1996)). Following the notation used by Hoogeveen et al. (1994), we use *nonfix* and *fix* in the second field of $\alpha|\beta|\gamma$ to denote the first and second classes of problems, respectively. Hence, $Pm|nonfix|C_{max}$ denotes an m-parallel-machine scheduling problem where each job can be processed simultaneously

by a fixed number of machines with the objective minimizing the makespan. Similarly, $P2|fix|\sum w_jC_j$ denotes a 2-parallel-machine scheduling problem where each job can be processed simultaneously by a specific set of machines, and the objective is to minimize the total weighted completion time. $Pm|prmp,nonfix|C_{max}$ denotes the same problem as $Pm|nonfix|C_{max}$ except that preemption of jobs is allowed.

1.1.1. The machine set not fixed

(a) $Pm|nonfix|C_{max}$

Blazewicz, Weglarz and Drabowski (1984) provide polynomial algorithms for the $Pm|prmp,nonfix|C_{max}$ problem with only two types of jobs: one-machine jobs and two-machine jobs. Blazewicz, Drabowski and Weglarz (1986) extend the results to the problem with more than two types of jobs. They show that for $Pm|nonfix|C_{max}$, polynomial algorithms exist only for unit processing times. Note that for each job, r is a fixed number and $r \leq m$. For the $Pm|nonfix, p_j = 1|C_{max}$ problem, since the total number of possible combinations of tasks in each unit time interval is fixed (function of r), the problem can be solved by either linear integer programming or dynamic programming polynomial in n. They also provide polynomial algorithms for the $Pm|prmp,nonfix|C_{max}$ problem.

Du and Leung (1989) study the problem where a job can be executed by one or more machines at the same time, and the processing time is a nonincreasing function of the number of machines used. However, the number of machines is decided before it is processed, i.e. the number cannot change during its processing. They call their problem a *parallel task system*, and note that if a multiprocessor task system is NP-hard in the strong sense, then the corresponding parallel task system is also NP-hard in the strong sense. On the other hand, if their parallel task problem can be solved by a pseudopolynomial algorithm, then the corresponding multiprocessor task problem is also solvable in pseudopolynomial time. For the nonpreemptive case, they show that the problem is NP-hard in the strong sense even for two machines in the case where jobs are subject to precedence constraints. For independent tasks, it is pseudopolynomially solvable for two and three machine problems. From the fact that even the special case $P2||C_{max}$ is NP-hard, it follows that $P2|nonfix|C_{max}$ and $P3|nonfix|C_{max}$ are also NP-hard in the ordinary sense. They also show that $P5|nonfix|C_{max}$ is NP-hard in the strong sense. The case $m = 4$, $P4|nonfix|C_{max}$, is an open question. The preemptive case is NP-hard in the strong sense if the number of machines is arbitrary and pseudopolynomially solvable if the number of machines is fixed. They also note that the NP-hardness of $P|prmp,nonfix|C_{max}$ remains an open question even though it is NP-hard in the strong sense for the corresponding parallel task problem.

Blazewicz et al. (1990) study the problem $Pm|nonfix|C_{max}$ on systems consisting of processors with the same speed, where there are only two types of tasks; one-processor tasks and two-processor tasks. They provide an $O(nm + n \log n)$ algorithm

to solve the problem optimally. Later, they provide an algorithm of the same complexity $O(nm + n \log n)$ to solve the same problem, yet on uniform k-processor systems consisting of k processors with the same speed, with only two types of tasks; one-processor tasks and k-processor tasks.

(b) $Pm|nonfix|\sum w_j C_j$

Lee and Cai (1996) show that the problem is strongly NP-hard even with two machines. Two special cases, (i) $w_j = w$ for all j and (ii) $p_j = p$ for all j, are also studied. For case (i) they show that $P2|nonfix|\sum C_j$ is NP-hard and provide a dynamic programming algorithm to solve the problem in $O(nP^{3s+1})$, where s is the number of two-machine jobs and P is the sum of all processing times. Namely, if the number of two-machine jobs is fixed then the problem is pseudopolynomially solvable and is NP-hard in the ordinary sense. If the number of two-machine jobs is not fixed, then whether the problem is strongly NP-hard remains an open question. For the second case, they provide an $O(n \log n)$ algorithm to solve the problem if $p_j = p$ for all one-machine jobs (the processing time of two-machine jobs can still vary). They also provide a heuristic algorithm that has an error bound of 1 for the $Pm|nonfix|\sum w_j C_j$ problem and an error bound of $1/2$ for the $Pm|nonfix|\sum C_j$ problem.

(c) $Pm|nonfix|L_{max}$

Plehn (1990) studies the problem with release dates, due dates, and preemption allowed. He uses linear programming to check whether there exists a feasible schedule where each job finishes before its due date. Lee and Cai (1996) show that the problem $Pm|nonfix|L_{max}$ is NP-hard in the strong sense even if $m = 2$. For the $P2|nonfix|L_{max}$ problem, they show that (i) between any two consecutive two-machine jobs, one-machine jobs must follow the EDD rule on each machine, and (ii) two-machine jobs should follow the EDD rule. They provide a dynamic programming algorithm for solving the problem in $O(nP^{3s+1}\log P)$, where s is the number of two-machine jobs and P is the sum of all processing times. Hence, if the number of two-machine jobs is fixed, then the problem is NP-hard in the ordinary sense. They also present an $O(n \log n)$ algorithm for the special case $P2|nonfix, p_j = 1|L_{max}$, as well as a heuristic algorithm with an error bound analysis.

1.1.2. The machine set fixed

(a) $Pm|fix|C_{max}$

The general problem $Pm|fix|C_{max}$ was studied by Bozoki and Richard (1970). Even though they were motivated by a manufacturing environment, their results did not attract attention in the scheduling field until the recent advent of parallel computing systems. They provide a branch and bound algorithm to solve the problem optimally.

Krawczyk and Kubale (1985) analyze this problem in their study of diagnosable microprocessor systems, where a job should be performed in parallel in order to detect

faults. They address issues on NP-hardness and solution procedures. In particular, they show that $P|fix, p_j = 1|C_{max}$ is NP-hard. Kubale (1987) shows that the problem $Pm|fix|C_{max}$ is strongly NP-hard even if there are only 2-machine jobs. Blazewicz et al. (1992) study a model similar to that of Kubale, except that they consider three machines in parallel, i.e., the $P3|fix|C_{max}$ problem. They show that the problem is NP-hard in the strong sense. They also present some special cases that are polynomially solvable, as well as heuristics for the general problem.

Hoogeveen, van de Velde and Veltman (1994) investigate the complexity of several models with the makespan objective. In particular, they show that $P3|fix|C_{max}$ subject to block-constraints is NP-hard in the ordinary sense, where a block means that all two-machine jobs of the same type are scheduled consecutively. They provide a pseudopolynomial algorithm with complexity $O(nP)$ to solve the problem, where P is the sum of all processing times. They also show that $P|fix, p_j = 1|C_{max}$ is NP-hard in the strong sense, while the problem is polynomially solvable if the number of machines is fixed, i.e., $Pm|fix, p_j = 1|C_{max}$. On the other hand, if jobs are released at different times, then the problem $P2|fix, r_j|C_{max}$ is NP-hard in the strong sense. Furthermore, if the distinct release dates are fixed, then the problem $Pm|fix, r_j, p_j = 1|C_{max}$ can still be solvable in polynomial time. Note that in the special case where $p_j = 1$ for all j, the maximum possible number of different types of jobs is $2^m - 1$. Hence, for such special cases, either dynamic programming, linear integer programming, or a network representation with a shortest path algorithm can be used to solve the problem in a time that is polynomial in n. For example, Brucker (1995) solves the problem $Pm|fix, p_j = 1, r_j|C_{max}$ in $O(R2^R n^{R+1})$, where $R \leq 2^m - 1$ is the maximum possible number of different type of jobs. When m is fixed, the algorithm is polynomial.

Recently, Krämer (1995) provided three branch and bound algorithms for solving the general problem $Pm|fix|C_{max}$ with precedence constraints.

(b) $Pm|fix|\sum w_j C_j$

Dobson and Karmarkar (1989) provide two integer programming formulations for the problem $Pm|fix|\sum w_j C_j$. They develop Lagrangian relaxation and surrogate relaxation to provide lower bounds. Heuristic algorithms and computational results are also provided. For a special case where $m = 2$, all jobs with unit processing time, and the same number of jobs to be assigned to machines 1 and 2, a polynomially optimal solution is also provided. Dobson and Khosla (1995) study the same problem of minimizing weighted flow times, yet with an additional decision of choosing batch sizes of the tasks. Both a Lagrangian relaxation and a surrogate relaxation are developed.

Hoogeveen, van de Velde and Veltman (1994) show that $P2|fix|\sum C_j$ is NP-hard, and state that whether it is NP-hard in the strong sense is still an open question. They also show that the three problems $P2|fix|\sum w_j C_j$, $P3|fix|\sum C_j$, and $P2|chain, fix, p_j = 1|\sum C_j$ are all NP-hard in the strong sense. Furthermore, they prove that $P|fix, p_j = 1|\sum C_j$ is NP-hard in the strong sense and state that the complexity of

the problem is open if the number of machines is fixed. Recently, Cai, Lee and Li (1996) showed that $P2|\,fix|\sum C_j$ is actually NP-hard in the strong sense. They also develop an efficient heuristic for $P2|\,fix|\sum w_jC_j$ with a relative error of at most 100%. For the $P2|\,prmp, fix|\sum C_j$ problem, they provide an $O(n \log n)$ algorithm to solve the problem optimally. Based on an approach similar to the one for $Pm|\,fix, p_j = 1, r_j|C_{max}$, Brucker (1995) shows that $Pm|\,fix, p_j = 1, r_j|C_j$, $Pm|\,fix, p_j = 1|\sum w_jC_j$ can be solved in $O(R2^Rn^{R+1})$, where $R \le 2^m - 1$. Therefore, $Pm|\,fix, p_j = 1|\sum C_j$ is polynomially solvable.

(c) $Pm|\,fix|\sum L_{max}$, $Pm|\,fix|\sum T_j$, and $Pm|\,fix|\sum w_jU_j$

Based on an approach similar to that for $Pm|\,fix, p_j = 1, r_j|C_{max}$, Brucker (1995) solves the problems $Pm|\,fix, p_j = 1|\sum T_j$ and $Pm|\,fix, p_j = 1|\sum w_jU_j$, both in $O(R2^Rn^{R+1})$ time, where $R \le 2^m - 1$. Note that $P2|\,fix|L_{max}$ can be shown to be NP-hard in the strong sense by the following argument. Given any instance of the 3-partition problem (a_j, $j = 1,\ldots,3k$ with $\sum a_j = kB$), construct an instance for our problem as follows: Let the total number of jobs be $n = 5k$. The first $4k$ jobs, $J_1,\ldots,J_{4k}$, can be assigned to any machine and the last k jobs, $J_{4k+1},\ldots,J_{5k}$, need to be processed simultaneously by two machines. Furthermore, their processing times are

$$\begin{aligned} p_j &= a_j, && j = 1,\ldots,3k,\\ p_j &= B, && j = 3k+1,\ldots,4k,\\ p_j &= B, && j = 4k+1,\ldots,5k. \end{aligned}$$

The due dates are

$$\begin{aligned} d_j &= 2kB, && j = 1,\ldots,3k,\\ d_j &= (2(j-3k)-1)B, && j = 3k+1,\ldots,4k,\\ d_j &= 2((j-4k)B, && j = 4k+1,\ldots,5k. \end{aligned}$$

It can be shown easily that there exists a solution for the 3-partition problem if and only if there exists a solution to our problem with $L_{max} = 0$.

Bianco et al. (1993) provide a polynomial time algorithm based on linear programming procedures to solve the $Pm|\,prmp, r_j, fix|L_{max}$ and $Pm|\,prmp, r_j, set|L_{max}$ problems, where *set* means that a job can choose a set of alternatives where each alternative contains several dedicated machines (i.e., *fix* is the special case of *set* when there is only one alternative for each job). Recently, Bianco et al. (1997) studied the preemptive deterministic problem with job release-dates and machine available time windows. They provide low-order polynomial time algorithms for several special problems with C_{max} and L_{max} criteria. They also provide linear programming algorithms to solve the general case.

(d) *Other problems*

Brucker and Krämer (1995, 1996) also discuss some other machine configurations including flow shop, job shop and open shop with the *fix* characteristic. Since

most general classical open shop and job shop problems are already NP-hard, it is not surprising that most problems in the 1-job-on-r-machine pattern are NP-hard except for some special cases either with constant processing time or with $n = k$. Motivated by the application of berth allocation, Li, Cai and Lee (1996) recently studied another type of problem where r is less than 1, i.e., a job can share a machine with other jobs. They assume that the job processing times and job sizes are agreeable. Heuristic algorithms are provided and their worst case performances are analyzed. Other machine configurations, especially the ones motivated by parallel computing systems, have been studied by Blazewicz, Drozdowski and Weglarz (1994).

1.2. Machine scheduling with availability constraints

Most papers in the scheduling area assume that machines are always available. However, in real industry settings this assumption may not be true, i.e. machines may not always be available. For example, machines may not be available because of machine breakdown (stochastic) or preventive maintenance (deterministic) during the scheduling period. In this subsection, we briefly review recent studies of machine scheduling with availability constraints in the *deterministic case*.

We assume that machine i is unavailable from s_{ik} until t_{ik} ($0 \le s_{ik} \le t_{ik}$), where $0 \le k \le n_i$, with n_i being the number of unavailability periods for machine i during the planning horizon. Instead of using "unavailability constraints", some papers state that "machines are available in time windows", which is particularly true in computer systems. Note that in most manufacturing cases, we may have $n_i \le 1$ because it is unlikely that we have more than one preventive maintenance period on the shop floor during the scheduling horizon. Note also that the special case with $s_{i1} = 0$ means that machine i is not available until t_{i1}. This happens, for instance, in the case where the machine has to complete those unfinished jobs that were scheduled during the previous planning period.

Two cases are discussed in the literature, resumable and nonresumable. If a job cannot be finished before the next down period of a machine and the job can continue after the machine has become available again, it is called *resumable*. On the other hand, it is called *nonresumable* if the job has to restart rather than continue.

In this subsection, we will use $r\text{-}a$ in the second field of $\alpha|\beta|\gamma$ to denote resumable availability constraints. Similarly, $nr\text{-}a$ in the β field denotes nonresumable availability constraints. Hence, $F2|r\text{-}a|C_{max}$ denotes the problem of minimizing makespan in the two-machine flowshop problem with a resumable availability constraint.

Schmidt (1984) studies an n job m parallel machine scheduling problem where each machine has different availability intervals. The purpose is to construct a feasible preemptive schedule. He presents an $O(n \log m)$ time algorithm to find a feasible preemptive schedule whenever one exists. Later (Schmidt (1988)) he provides an $O(nm \log n)$ algorithm to solve a generalized problem where jobs may have different deadlines.

Adiri et al. (1989) consider the $1|nr\text{-}a|\sum C_j$ problem. They study both the stochastic case where the location and duration of the unavailability periods are random and the deterministic case where machine unavailability is known in advance. For the deterministic case, they show that the problem is NP-hard under the assumption that there is only one unavailability period. Lee and Liman (1992) study the same problem, and provide a simpler proof of NP-hardness. They also show that the tight error bound for the SPT heuristic is $2/7$.

Lee (1991) studies the parallel machine problem to minimize makespan and machines may not be available at time zero. Thus, there is at most one unavailability period for each machine and this happens at the beginning of the time horizon. He shows that the classical Longest Processing Time (LPT) algorithm has a tight error bound $1/2$, and provides a modified LPT algorithm with error bound equal to $1/3$.

Kaspi and Montreuil (1988) and Liman (1991) show that, when there is at most one unavailability period for each machine and the unavailability period happens at the beginning of the time horizon, the Shortest Processing Time (SPT) algorithm is an optimal policy for the parallel identical machines problem to minimize total completion time. Lee and Liman (1993) study the two machine in parallel scheduling problem of minimizing the total completion time where one machine is available all the time and the other machine is available from time zero up to a fixed point in time. They prove that the problem is NP-hard and use dynamic programming to solve it.

Mosheiov (1994) studies the same problem under the condition that machine i is available in the time window $[x_i, y_i]$, where $0 \le x_i < y_i$. He shows that SPT is asymptotically optimal for the m machines in parallel problem.

Lee (1996b) studies $F2|r\text{-}a|C_{max}$ and $F2|nr\text{-}a|C_{max}$ problems with the assumption that at least one machine is always available. Hence, he considers two cases, with an unavailability constraint imposed on machines 1 and 2, respectively. The notation $F2|r\text{-}a(M_i)|C_{max}$ is used to denote $F2|r\text{-}a|C_{max}$ when an availability constraint is imposed on machine i. In each case, he proves that the problem is NP-hard and proposes a pseudopolynomial dynamic programming algorithm to solve the problem optimally. He also provides two $O(n \log n)$ heuristic algorithms. The first heuristic is used to solve the problem $F2|r\text{-}a(M_1)|C_{max}$ and has a worst-case error bound of $1/2$. The second heuristic is for the problem $F2|r\text{-}a(M_2)|C_{max}$ and has a worst-case error bound of $1/3$.

Lee (1996a) also studies the problem considering different performance measures (makespan, total weighted completion time, tardiness, and number of tardy jobs) for single machine and parallel machines problems. For resumable problems, he shows that SPT solves $1|r\text{-}a|\sum C_j$ and EDD solves $1|r\text{-}a|L_{max}$ optimally. Furthermore, Moore and Hodgson's algorithm can be modified to solve the $1|r\text{-}a|\sum U_j$ problem. However, $1|r\text{-}a|\sum w_j C_j$ is NP-hard and he provides a dynamic programming approach to solve the problem and an heuristic approach with an error bound analysis. Since the classical problem $Pm||C_{max}$ is NP-hard, problem $Pm|r\text{-}a|C_{max}$ is also NP-hard. He then analyzes the worst case performance of the LPT algorithm. He notes that after sorting

the jobs in nonincreasing order of their processing times, there is a performance difference between "assigning a job to the minimum loaded machine, calling it LPT1", and "assigning a job to the machine such that the finishing time of that job is minimized, calling it LPT2" (these two rules result in the same sequence in the classical problem $Pm||C_{max}$). He shows that the error bound of LPT1 can be arbitrary large, while $C_{\mathrm{LPT2}}/C^* \leq 2$, where C_{LPT2} is the makespan under LPT2, and C^* is the optimal makespan. For nonresumable problems, he shows that even $1|nr\text{-}a|C_{max}$ is NP-hard and hence other problems such as $1|nr\text{-}a|L_{max}$ and $1|nr\text{-}a|\sum U_j$ are NP-hard. He develops a pseudopolynomial dynamic programming framework for solving $P2|nr\text{-}a|\sum w_j C_j$ to optimality and heuristic approaches for most problems. In particular, he shows that $C_{\mathrm{LS}}/C^* \leq m$ and $C_{\mathrm{LPT2}}/C^* \leq (\mathrm{m} + 1)/2$, where C_{LS} is the makespan obtained by List Scheduling and C_{LPT2} and C^* are defined above. Note that the preemptive case, $Pm|prmp, r\text{-}a|C_{max}$, can be solved in $O(n + m \log m)$ (Schmidt (1984)).

As mentioned in subsection 1.1.2, Bianco et al. (1997) study the scheduling problem that combines the machine availability constraint (time windows) with the 1-job-on-*r*-machine pattern. They study the preemption case with C_{max} and L_{max} as optimization criteria (see the paper in this volume for a detailed discussion).

2. Recent developments in search algorithms

Many scheduling problems are so complex that they cannot be formulated easily as mathematical programs (e.g. integer programs, disjunctive programs). The fact that they are not easy to formulate makes it difficult to apply classical techniques such as branch and bound or dynamic programming. These problems often do not have to be solved in real time.

With the advent of fast and inexpensive computing power, researchers have begun to experiment during the last decade with search techniques that are easier to implement than the classical operations research techniques. A distinction can be made between two types of search techniques, namely neighbourhood search techniques (frequently used by operations researchers and industrial engineers) and constraint-guided heuristic search techniques (often used by computer scientists and artificial intelligence experts).

The neighbourhood search techniques are based on the concept of local improvement. Given an existing solution of the problem at hand, a (typically minor) modification is made in order to obtain a different (usually better) solution. The programming effort required to implement such a technique often is fairly modest; the structural knowledge needed with regard to the problem is significantly less than the knowledge required for a mathematical programming approach.

In the seventies, researchers started to apply neighbourhood search techniques in scheduling problems. Very early experiments showed that if random swaps were applied to a reasonably good solution, then the payoff is minimal (taking into account

the computation time involved). The neighbourhood search techniques subsequently developed are therefore considerably more sophisticated than random swaps. An early example of a neighbourhood search technique is the *k-opt* approach designed by Lin and Kernighan (1972) for the Travelling Salesman Problem (TSP). The TSP is equivalent to a single machine problem with sequence-dependent setup times and the makespan as objective; this scheduling problem has in the literature been referred to as the $1|s_{jk}|C_{max}$ problem, where s_{jk} denotes the setup time incurred between J_j and J_k. The subsequent research in neighbourhood search has focused mainly on three techniques, namely *simulated annealing*, *tabu search*, and *genetic algorithms*.

The scheduling applications have included a variety of machine environments (see Lee, Bhaskaran and Pinedo (1992), Morton and Ramnath (1995), and Pinedo (1995)). The objective most often considered has been the makespan. Of the three methods, tabu-search has been applied most often to scheduling problems. However, lately researchers have started focusing on genetic algorithms as well; these efforts are often in conjunction with the design of learning mechanisms.

Constraint-guided heuristic search techniques are completely different from neighbourhood search techniques. Constraint-guided heuristic search techniques do not attempt to find optimal schedules; they merely seek to find a good feasible schedule. The problem is basically formulated through a list of rules or constraints that the schedule has to satisfy. In contrast to neighbourhood search, constraint-guided heuristic search focuses on partial solutions and attempts to extend these partial solutions until a complete solution is obtained that is feasible. Constraint-guided heuristic search techniques are often based on measurements of flexibility and constraining factors. In the beginning of the search an attempt is made to satisfy the more stringent constraints; the less stringent constraints are left for the final part of the search process. Many techniques have been developed for managing the constraints and speeding up the search.

Constraint-guided heuristic search techniques have been implemented in a number of scheduling systems, which often have been implemented in programming languages that are specifically designed for this type of search, such as PROLOG. These scheduling systems are often classified as expert systems.

The remaining part of this section is organized as follows. In the first subsection we discuss the general concepts of neighbourhood search. In the second we describe in more detail simulated annealing, tabu search and their applications. In the third subsection we consider genetic algorithms and in the fourth we describe constraint guided heuristic search.

2.1. General concepts in neighbourhood search

The design of the different neighbourhood search techniques tends to be similar in many respects. One can compare the various neighbourhood search techniques based on the following four design criteria.

(1) The mapping of the data in a format suitable for the algorithm.
(2) The neighbourhood design.
(3) The search process within the neighbourhood.
(4) The acceptance-rejection criterion.

With regard to the first point, the description of a schedule has to be both concise and unambiguous. For some problems, a concise description can at times be difficult. For example, in a genetic algorithm a nonpremptive schedule is usually represented by a *chromosome*, which is basically a series of decimal digits. Such a chromosome is often partitioned into subchromosomes, with each subchromosome representing the sequence of the operations on a particular machine. The entire chromosome specifies the complete job shop schedule (see, for example, Della Croce, Tadei and Volta (1995)). This representation is concise and at times referred to as the natural representation; it does not leave any room for ambiguities. However, consider now a variant of the problem with preemption allowed. The representation of a schedule now becomes significantly more complicated since the processing of an operation on a machine may be interrupted a number of times. A larger amount of information has to be carried in the chromosome, and even the length of the chromosomes is no longer fixed.

The neighbourhood design specifies the set of all the neighbours of a given solution. The neighbourhood design usually requires some knowledge of the problem. The knowledge required centers mainly on those aspects of the schedule that have the greatest impact on the objective. Some neighbourhood designs are very simple, e.g., in a single machine scheduling problem, the neighbourhood of any given schedule can consist of all schedules that can be obtained through a (not necessarily adjacent) pairwise swap. In an even simpler design, the neighbourhoods consist of all schedules which can be obtained via an insertion (i.e., a job is taken from somewhere in the schedule and inserted somewhere else). Of course, there are also much more complicated neighbourhood designs. For example, in the job shop scheduling problem $Jm||C_{max}$, the makespan is determined by the length of a critical path. If a neighbouring schedule is the same with respect to the sequence of operations on the critical path, then the value of the objective cannot be lower. It is therefore advantageous to have in a neighbouring schedule some operations on the critical path sequenced differently (see Matsuo, Suh and Sullivan (1988) and van Laarhoven, Aarts and Lenstra (1992)). For job shops with other objectives, e.g., the total weighted tardiness ($Jm||\sum w_j T_j$), even more complicated neighbourhoods have been designed. One such neighbourhood is based on a cluster of operations to be processed on various machines and subject to extensive delays; the neighbourhood is then determined by changes in the sequences of these operations (see Pinedo and Singer (1995)).

Given all the schedules in the neighbourhood, a search has to be conducted that leads to the next schedule in the search process. A simple way is to select schedules in the neighbourhood at random, evaluate these schedules and decide which one to

accept. However, it may pay to do a more organized search and somehow first select the schedules that appear most promising. For example, consider swaps of jobs that affect the objective the most (e.g., jobs with a large weight that are very tardy when the total weighted tardiness has to be minimized).

The acceptance-rejection criterion is closely related with the neighbourhood search process. Whenever a schedule within the neighbourhood is selected, a decision has to be made whether or not to accept the schedule. It is only at this point that simulated annealing and tabu search are different from one another. The acceptance-rejection technique of simulated annealing is a probabilistic process, while the acceptance-rejection technique of tabu search is a deterministic process. Both techniques need settings of parameters that affect the search process significantly (the so-called cooling parameter in simulated annealing and the length of the tabu list in tabu search).

Genetic algorithms, as a search process, are in one important aspect different from simulated annealing and tabu search. The result of each iterative step is a number of different schedules and all are carried over to the next step (in simulated annealing and tabu search, only a single schedule is transferred from one iteration to the next). This diversification scheme is an important characteristic of genetic algorithms. In genetic algorithms, the neighbourhood concept is therefore not based on a single schedule, but rather on a set of schedules. A new schedule can be constructed by combining different parts from different schedules within the set. The design of the neighbourhood of the given set of schedules is based on techniques that are somewhat different from those used in simulated annealing and tabu search.

For a comprehensive overview of the many aspects of neighborhood search, see the survey paper by Vaessens, Aarts and Lenstra (1996).

2.2. Simulated annealing and tabu search

Simulated annealing and tabu search have become very popular over the last decade. Of these two methods, simulated annealing was the first one to appear (see Kirckpatrick, Gelatt and Vecchi (1983)). However, even though tabu search appeared later (see Glover (1989, 1990), Barnes and Laguna (1993), Glover, Taillard and de Werra (1993), and Barnes, Laguna and Glover (1995)), it is currently more widely used than simulated annealing in production scheduling. Since the two techniques have many characteristics in common, they are discussed here in parallel. The neighbourhood design as well as the search pattern within a neighbourhood can be the same for the two techniques.

The difference between simulated annealing and tabu search lies in the acceptance-rejection criterion.

The acceptance-rejection technique in simulated annealing functions according to the following probabilistic process. If at iteration k the process is at schedule S_k and the neighbouring schedule S is considered a candidate to move to, then schedule

S is accepted as the next one if the objective value under S, say $G(S)$, is less (better) than the objective value under S_k, say $G(S_k)$. However, if the objective value under S is larger (worse) than under S_k, then S may still be accepted as the next schedule. Schedule S is then accepted with probability

$$\mathbf{P}(S_k, S) = \exp\left(\frac{G(S_k) - G(S)}{\beta_k}\right)$$

and rejected with probability $1 - \mathbf{P}(S_k, S)$. The parameters $\beta_1 \geq \beta_2 \geq \ldots \geq 0$ are control parameters, often referred to as cooling parameters. Frequently, the parameter β_k is chosen to be α^k, where α is a constant slightly smaller than 1.

In contrast to the acceptance-rejection technique in simulated annealing, the one in tabu search functions according to a deterministic process. The search process keeps track of a so-called tabu list with a fixed number of entries. This number often lies between 5 and 15. Every time the search performs a mutation in order to go from one schedule to a neighbouring schedule, the reverse mutation is placed at the top of the tabu list. All other entries on the list are pushed down one position and the entry at the bottom is deleted. The reason for keeping such a tabu list is based on the fact that it is not desirable to allow the search in a subsequent move to return to a schedule already considered. Since the most basic form of tabu search is a deterministic process, there is always the danger of *cycling*. The cycling phenomenon depends very much on the length of the tabu list. If the length of the tabu list is too small, say 2 or 3, the process may have a high likelihood of cycling. If the length of the list is chosen too large, then the freedom of the search process is curtailed and the process may be less likely to find good solutions.

The first applications of simulated annealing and tabu search focused on Travelling Salesman type problems. The neighbourhood design chosen for the TSP is often based on the 2-opt or k-opt method introduced by Lin and Kernighan (1972).

Simulated annealing has been used subsequently for job shop scheduling problems with the makespan objective, i.e., $Jm||C_{max}$, by Matsuo, Suh and Sullivan (1987). They design a neighbourhood based on pairwise interchanges of operations on the critical path and operations close to the critical path. Their interchanges are called *Multi-Step Look-Ahead* and *Multi-Step Look-Back* interchanges.

Tabu search has been used for single machine, parallel machine, flow shop, flexible flow shop and job shop problems with objectives that include the makespan, the total weighted completion time, as well as the total weighted tardiness. Crauwels, Potts and Van Wassenhove (1997) apply tabu search on the single machine with the total weighted completion time as objective, i.e., $1||\sum w_j C_j$. Laguna, Barnes and Glover (1991,1993) consider single machine problems with sequence dependent setup times, namely $1|s_{jk}|\sum w_j C_j$ and $1|s_{jk}|\sum T_j$. Barnes and Laguna (1992) and Barnes, Laguna and Glover (1995) consider the parallel machine problem with the total weighted completion time as objective, i.e., $Pm||\sum w_j C_j$. For this problem, it is known

that the jobs assigned to any given machine have to be scheduled in decreasing order of the weight divided by the processing time. The problem therefore reduces to the optimal partition of the *n* jobs over the *m* machines. Adenso-Dias (1992) and Nowicki and Smutnicki (1994) apply tabu search on the flow shop problem. Dell'Amico and Trubian (1993), Nowicki and Smutnicki (1993), Taillard (1994), and Dauzère-Pérès and Paulli (1997) apply tabu search on the job shop scheduling problem with the makespan as objective, i.e., $Jm||C_{max}$. Hurink, Jurisch and Thole (1994) apply tabu search on a job shop problem with multi-purpose machines.

Several mechanisms have been devised in the literature to improve the overall efficiency of tabu search (see Hubscher and Glover (1994), Laguna and Glover (1993), Laguna and Gonzalez-Velarde (1991), and Reeves (1993)). A number of hybrid approaches, combining simulated annealing with tabu search, have been developed also.

2.3. Genetic algorithms

A genetic algorithm typically embodies a search process that simulates a natural evolutionary process. The technique was first suggested by Holland (1973, 1975). At the end of each iteration there is a population of feasible solutions. This population is referred to as a generation. Each solution is referred to as an individual and in the subsequent iteration, the next generation of individuals is selected. The least fit individuals of the previous generation die off and the fittest individuals are allowed to reproduce. Individuals are referred to as chromosomes. A chromosome may consist of subchromosomes. Each subchromosome may represent the schedule of operations on a given machine.

A number of different mutations or transformations can be done on the different individuals in the population. One can take a promising individual in the population and perform a mutation that is in a way equivalent to an interchange or insertion in a (sub)sequence of the jobs or operations. Or, one can consider two individuals in the population and create a new individual by combining one part of one chromosome with another part of the other chromosome. Such an operator is in the literature at times referred to as a recombination operator or a cross-over.

One can make an argument that tabu search and simulated annealing are special forms of genetic algorithms with the number of individuals in each generation equal to one. The fact that genetic algorithms keep track of multiple solutions at each iteration may make them more powerful (but at the same time slower) than simulated annealing and tabu search.

The first applications of genetic algortihms on combinatorial problems focused on the Travelling Salesman Problem. However, during the last five years several researchers have applied genetic algorithms to scheduling problems, in particular the job shop problem $Jm||C_{max}$ (see Lawton (1992), Della Croce, Taddei and Volta (1992), Bean (1994), Bierwirth (1995), and Herrmann, Lee and Hinchman (1995)). Lee,

Piramuthu and Tsai (1995) combine the application of a genetic algorithm to a job shop with machine learning. For an extensive treatise focusing on genetic algorithms and machine learning, see Pesch (1994).

Recently, there have been a number of applications and implementations of genetic algorithms in the real world. Bean (1994) develops a very sophisticated form of a genetic algorithm for a scheduling problem in the automotive industry which he implemented on a MASPAR computer. Bean concludes from his research that his specialized genetic algorithm is suitable for real-time scheduling. Mayrand, Lefrancois, Kettani and Jobin (1995) develop a genetic algorithm for a scheduling problem that occurs in rolling mills in the aluminum industry. They showed that when the problems are small, their approach leads to near optimal solutions (within 1%). Herrmann, Lee and Hinchman (1995) use a genetic algorithm to develop a global job shop scheduler for semiconductor manufacturing test operations. They report that the system was successfully implemented in a semiconductor test facility and improved the on-time delivery rating significantly.

2.4. Constraint-guided heuristic search

As stated before, constraint-guided heuristic search is completely different from neighbourhood search. The development of constraint-guided heuristic search techniques has taken place independently from the development of neighbourhood search techniques.

In the early eighties, with the popularization of artificial intelligence techniques and languages (e.g., PROLOG), many rule-based scheduling systems were developed. The developments during this period were spearheaded by research groups at Carnegie-Mellon University and resulted in the development of the ISIS and OPIS scheduling systems (see Smith, Muscettola, Matthys and Ow (1990) and Smith (1992)). Since the scheduling constraints were often incorporated into a system as rules, the necessity arose for the development of constraint management techniques as well as of constraint-guided heuristic search techniques.

The search techniques that came out of this research did not focus on finding *optimal* schedules, they merely focused on finding *feasible* schedules. Actually, an objective function was often not even defined. The problem was basically formulated through a list of rules or constraints that the schedule has to satisfy. As stated before, a constraint-guided heuristic search focuses on partial solutions and attempts to extend these partial solutions until a complete solution is obtained that is feasible. These techniques are often based on measurements of flexibility and constraining factors, i.e., in the beginning of the search an attempt is made to satisfy the most severe constraints first. The least severe constraints are left for the final part of the search process. For an overview of constraint-based scheduling strategies, see Amiri and Smith (1992) and Cheng and Smith (1997).

Frequently, it is not possible to find a feasible schedule and one or more constraints may have to be violated in the schedule. If this is the case, it is advantageous

to make a distinction between soft constraints and hard constraints. If a feasible schedule is not found with the original set of constraints, then one or more of the soft constraints are relaxed and the system tries again to find a feasible schedule (see Cheng and Smith (1997)). A fair amount of research has been done on constraint relaxation techniques.

Another area of research within this field is constraint propagation techniques. Often, having to satisfy two constraints may make it necessary to satisfy a third, not listed, constraint as well. Constraint propagation is very important in the search process. Listing all the additional "implied" constraints as early as possible speeds up the search process considerably. Suppose the system has constructed a partial schedule by assigning a number of operations to machines. This partial schedule, together with the original set of constraints, imposes additional constraints on the operations remaining to be scheduled. It is advantageous, whenever a job has been assigned to a machine (i.e., whenever a partial schedule has been extended), to generate all additional constraints the remaining jobs have to satisfy.

Also, before the time-consuming search for a feasible schedule starts and whenever additional, implied, constraints are generated, it is advantageous to do *consistency checking*. That is, it has to be verified if a feasible schedule actually exists. If two constraints are not consistent, it is advantageous to find this out early in the search process and do the appropriate backtracking. If there are inconsistencies, they have to be dealt with in an efficient way. A significant amount of research has been done on consistency checking. Dealing with inconsistencies is often referred to as *conflict resolution*. Conflict resolution techniques may be based on the distinction between hard constraints and soft constraints. However, it may be that a fair amount of information has to be fed into the system with regard to the possible constraint relaxations.

For a survey of the various constraint-guided heuristic search techniques, see Amiri and Smith (1992). For an overview of the implementation of these techniques, see Smith (1992) and Noronha and Sarma (1991). For a simple example of a constraint-based search technique, see Pinedo (1995).

3. Recent developments in scheduling practice

A number of emerging new applications of scheduling have gained significant attention of researchers and practitioners during recent years. Examples of these emerging areas are flexible-resource scheduling (Daniels and Mazzola (1993, 1994), Ozdamar and Ulusoy (1995), Daniels, Hoopes and Mazzola (1996, 1997), Alidaee and Ahmadian (1997), Alidaee and Kochenberger (1997), and Armstrong, Gu and Lei (1997a, 1997b)), scheduling variable-speed machines (Trick (1994)), scheduling with finite capacity input and output buffers (Hall, Posner and Potts (1993, 1994, 1997), Nawijn, Kern and Bass (1994)), scheduling of machine and material handling operations (Egbelu (1987), Matsuo, Shang and Sullivan (1991), Hall, Kamoun and Sriskandarajah (1993), Lei, Armstrong and Gu (1993, 1995), Blazewicz, Drozdowski

and Weglarz (1994), Hall, Kamoun and Wan (1994), and Crama (1995)), and integrating scheduling with batching and lot-sizing (Potts and Van Wassenhove (1992)).

To limit the length of this survey, we confine our scope to the recent work in machine scheduling with material handling operations. This area differs from classical machine scheduling in the sense that two types of resources are now involved: machines and material handling transporters. Either resource could become a bottleneck if not properly scheduled. Transportation operations that move jobs between machines are non-instantaneous, and the transportation duration depends on the sequence in which material movement is executed.

It is known that in any manufacturing system, material handling is expensive. Material handling can take a significant portion, at times as high as 80%, of the total cost (Tompkins and White (1984)). Attempts to reduce this cost have tremendously increased the need for new and effective methodologies for the scheduling of machine and material handling operations. To meet this need, the following issues must be addressed simultaneously:

(a) *Sequencing* that specifies the order in which jobs are processed at machining centers;

(b) *Scheduling* that makes time-phased routing and dispatching of transporters for job pick-up and delivery; and

(c) *Facility layout and flowpath design* that makes efficient operations possible.

Due to the combinatorial nature of the problems, finding an optimal solution that addresses all these issues at the same time is very difficult. Most studies reported in the literature consider at most two of these issues. In this section, we shall review recent work on the first two issues. To deal with these issues, we need to expand the notation $\alpha|\beta|\gamma$ to $\alpha(K)|\beta|\gamma$, where K denotes the number of transporters in a system. In addition, we use J to denote the total number of job types, n the total number of jobs to be processed, n_{mps} the number of jobs in a minimal part set (MPS), Π_{min} the objective of minimizing the production cycle time of an MPS in a repetitive process, *tw* a manufacturing environment where the starting time of each material handling operation must be confined within a *time window*, and *nwt* the constraint that jobs are not allowed to wait in process.

A general model to review this work can be formulated as follows. We are given a set of n jobs to be processed, and m machining centers. All jobs are ready at the time zero (i.e., we do not consider dynamic arrivals), each with its own route and processing specifications. The deliveries of jobs between machining centers are performed by K, $K \geq 1$, identical transporters. These transporters travel on a shared network where traffic collisions must be avoided. All the operations by the transporters, including loading, unloading and moving jobs between machining centers, are non-instantaneous and non-preemptive. Neither a machine nor a transporter can hold more than one job at any time. The problem is to find a simultaneous feasible schedule

for job sequencing and time-phased dispatching and routing of transporters so that a given objective is optimized. Recent work related to this model can be divided into:

(1) Robotic cell scheduling;

(2) Scheduling of Automated Guided Vehicles (AGVs); and

(3) Cyclic scheduling of hoists subject to time-window constraints.

These types of problems differ mainly in the structure of their constraints. Among these, the robotic cell scheduling problem has the fewest constraints, and is also the one for which most analytical results are available. In most related studies, the cell is of a flowline type with several flexible machines and a single material handling robot. The size of in-process buffers is either zero or finite. This makes the sequence of robot moves and the order in which jobs are introduced into the cell determine, to a great extent, the cell performance (e.g., the resulting makespan and machine utilization). The main concern of robotic cell scheduling is to identify the optimal job input sequence and the robot operation sequence with respect to certain objective functions. On the other hand, AGV scheduling typically deals with an automated job shop with non-zero buffers at machining centers and multiple AGVs traveling on a shared network. A major constraint that must be satisfied by any AGV schedule is to avoid traffic collisions of AGVs during their operations. The AGV scheduling problem is mainly concerned with how to schedule the moves of AGVs in a traffic network so that traffic collisions are eliminated and the risk of machine blocking (i.e., machine deadlock) is minimized. Among the three, the problem of cyclic scheduling of hoists with time window constraints is perhaps the most restrictive. It typically deals with the scheduling of multiple hoists in a flexible flowshop. The most distinct feature of the hoist scheduling problem is that the job processing time at each machine is strictly limited by a lower and an upper bound (i.e., the time window or *tw* constraints). This means that any hoist schedule that causes a hoist not to pick up a job within the time window is infeasible. In addition, the hoist scheduling problem is also subject to the *nwt* and *collision-free* constraints. The general versions of these problems are all NP-hard in the strong sense. Most of these, especially those encountered in real systems, are so complicated that they preclude a formal mathematical formulation.

The main purpose of this section is not to make a complete coverage of all the work published on these topics, but rather to review some of the recent developments in approaches for the related scheduling problems.

3.1. The robotic cell scheduling problem

The problem of robotic cell scheduling typically arises in cellular manufacturing systems consisting of a series of cells. Each cell is equipped with a single material handling robot and several flexible machines that produce MPSs repetitively. To reduce the holding cost, buffers between machines are limited (either zero or finite

size). This makes the cell performance (e.g., makespan, machine utilization, or tardiness) depend, to a great extent, on the sequence of robot moves and the order in which jobs are loaded into the cell. One common objective considered in the literature is to minimize the steady-state cycle time, or $Fm(1)|J>1|\Pi_{min}$. An extensive review of the work in this area up to 1991 can be found in Sethi et al. (1992). Here, we will cover mainly the results developed after 1992.

3.1.1. The no-buffer case

A major stream of work in this area considers the case with no buffer between machines. Under this assumption, several elegant polynomial algorithms are available for 2-machine cells ($m = 2$). Sethi et al. (1992) find a strong polynomial procedure that generates the optimal job sequence for $F2(1)|J>1|\Pi_{min}$, assuming that a fixed one-unit cycle is used (a one-unit cycle is a particular sequence of robot moves in which each machine is loaded and unloaded exactly once per cycle). This result is achieved by showing that the problem of finding the optimal job input sequence can be reduced to a solvable case of the traveling salesman problem. They also show that for any given m-machine cell, there are only $m!$ one-unit cycles. Hall, Posner and Potts (1993, 1994) and Hall, Kamoun and Wan (1994) further extend the results by Sethi et al. (1992). They indicate that the optimal cycle time is not necessarily attained by repeating the same one-unit cycle, even when $m = 2$. When $m = 2$, there are a total of two one-unit cycles, S_1 and S_2, and therefore four combination cycles based on S_1 and S_2 for each job to be processed. They then introduce an $O(n_{mps}^4)$ algorithm, *MinCycle*, that jointly optimizes the robot moves and job sequence (Hall, Kamoun and Sriskandarajah (1996a)). This polynomial algorithm finds the optimal schedule by evaluating alternative cycle combinations and their associated optimal job sequences. To handle the instances with large number of jobs effectively, Hall, Kamoun and Sriskandarajah (1996b) also introduce a simple but effective heuristic, *QuickCycle*. This heuristic differs from MinCycle as it considers only a subset of (promising) candidate combinations of cycles. Empirical results show that this Quick-Cycle routinely generates solutions very close to optimal while the required number of computation steps is bounded by $O(n_{mps}^2)$.

With respect to $F2(1)|J>1|C_{max}$, Kise, Shioyama and Ibaraki (1991) propose an $O(n^3)$ procedure that solves the problem based on the known Gilmore and Gomory algorithm (Gilmore and Gomory (1964)). For the same problem, however, if the transportation time between machines is job dependent (Stern and Vitner (1990)), then the problem is equivalent to an asymmetric traveling salesman problem and is NP-hard in the strong sense (Ganesharajah, Hall and Sriskandarajah (1995)).

However, both $Fm(1)|J>1|C_{max}$ and $Fm(1)|J>1|\Pi_{min}$, even with no buffer and a fixed one-unit cycle, are NP-hard when $m \geq 3$. A formal proof of this together with several interesting results are presented in Hall, Kamoun and Wan (1994). For the three-machine cell ($m = 3$) case, they show that the optimal job sequence can be found by an efficient algorithm in four of the six available cycles. In the other two cycles,

the job sequencing problem becomes intractable. For these difficult problems, they then develop heuristics based on the Gilmore and Gomery algorithm. In general, Sriskandarajah, Hall and Kamoun (1994) show that the job sequencing problem is polynomial solvable for $2m - 2$ of the $m!$ available cycles and unary NP-hard for the remaining cycles, for any $m \geq 2$.

On the other hand, if we restrict ourselves to the cases of either $Fm(1)|J = 1|C_{max}$ or $Fm(1)|J = 1|\Pi_{min}$, that is, all jobs are identical and thus the job sequencing problem vanishes, then the problem becomes polynomial solvable. Sethi et al. (1992) propose a simple decision rule that finds the optimal schedule for robot moves when there are only three machines ($m = 3$) in the system. Crama and Klundert (1994) further extend this result and prove that the robotic cell scheduling problem with identical jobs and an arbitrary large m can be solved in a time that is strongly polynomial in m. This result is achieved by showing that the set of pyramidal permutations necessarily contains an optimal solution of the problem. Based on this observation, they then propose an efficient dynamic programming approach that solves the special case optimally in $O(m^3)$ time.

3.1.2. The finite buffer case

The robotic cell scheduling problem with finite, but non-zero, input and output buffers at machine centers and zero transportation times is known to be NP-hard. King, Hodgson and Chafee (1993) consider the 2-machine scheduling problem, $F2(1)|J > 1|C_{max}$, where each machine has a finite input buffer, but no output buffer (i.e., a machine becomes blocked if a completed job is not removed). They assume a fixed job input sequence, and then determine the sequence of robot moves that minimizes the makespan by using a branch-and-bound procedure. This procedure solves a large number of longest-path subproblems during the search. Also assuming that the input sequence of a given set of jobs is fixed, Jeng, Lin and Wen (1993) propose a branch-and-bound procedure to find the optimal schedule of robot moves in a robotic cell with multiple parallel machines. The objective is to minimize the total flow time, or $\sum C_j$. To speed up the search, a heuristic based on the SPT rule is used to derive an initial sequence and an upper bound for the search.

3.1.3. Related research

There are several interesting studies closely related to this area. Matsuo, Shang and Sullivan (1991) study the problem of scheduling a single transporter (crane) in a flexible automated flowshop. The objective is to maximize the production rate subject to job flowtime limit. They show that, if there is only a single product being produced, then cyclic schedules provide a near-optimal solution with a relative error that is asymptotically zero when the number of cycles approaches infinity. For the general case with parallel identical machines and multiple products, they find that if the job input sequence is fixed, then a schedule that minimizes the cycle time can be

derived in $O(n_{mps}J + J^3)$. This schedule is derived by solving an associated maximum cost circular network flow problem. Heuristics for constructing job sequences are also proposed. Rao and Jackson (1993) consider cyclic schedules for re-entrant flowshops in which transportation delays are fixed and robots are modeled as regular machines.

A special case of robotic cell scheduling occurs when all machines are identical and working in parallel. A thorough analysis of the complexity of the various associated scheduling problems, with respect to different objectives (e.g., C_{max}, L_{max}, $\sum T_j$, etc.), is given in Hall, Potts and Sriskandarajah (1994).

3.2. Scheduling of automated guided vehicles (AGVs)

The AGV scheduling problem typically occurs in the process of flexible manufacturing. Such a flexible manufacturing system (FMS) usually consists of numerically controlled machining centers, each with limited input and output buffers, interconnected by a material flow network. It can be viewed as an automated job shop, or $Jm(K)|J > 1|\gamma$, with non-instantaneous material delivery, and is capable of processing a wide variety of different jobs of small lot sizes to meet specific customer needs. During the manufacturing process, AGVs circulate on a network of guidepaths connecting machine centers, and transport tools and jobs among the centers. Because of the level of sophistication and complexity of an FMS, any improper dispatching of AGVs will necessarily lead to congestion, collision, and lengthy delays in manufacturing processes (Sinriech and Tanchoco (1992), Riopel and Langevin (1991), Bozer and Srinivasan (1991)).

AGV flowpaths that are commonly used in practice include: the *unidirectional* ($\rightarrow$) and the *bi-directional* ($\leftrightarrow$) flowpath. In general, bi-directional networks often result in a higher control and implementation cost, but have a greater potential to improve productivity, require fewer AGVs, and reduce AGV travel time. There are also two kinds of commonly used network configurations: a *single-loop* and a *multi-loop* network. With a single-loop configuration, all machines are accessible via the loop, and the avoidance of AGV collisions is relatively easy to address. Consequently, most analytical studies that find optimal solutions to the special cases of the AGV scheduling problem assume the single-loop layout. With a multi-loop network, AGV collision and the risk of machine blocking (which occurs when the output buffer of a machine is full) become major concerns in scheduling, especially on a bi-directional network. For a more detailed coverage, classification and operational issues of AGV systems, we refer to the work of Egbelu and Roy (1988), Chu, Egbelu and Wu (1995), and Ganesharajah, Hall and Sriskandarajah (1995).

3.2.1. Analytical approaches to AGV scheduling

Most analytical approaches that guarantee the optimal AGV schedule with respect to certain objective functions are limited to special cases.

Undirected flowpath case

With the assumption of unidirectional flowpath, Blazewicz et al. (1991) show that, if the deadlines of deliveries to machines are fixed, then the existence of a feasible schedule with a given fleet size on a single-loop network can be determined in $O(n \log(n))$, where n is the total number of delivery operations to be performed. With a similar application background, Blazewicz et al. (1994) study the problem of AGV scheduling on a two-loop network with a common stretch on which AGVs can switch between loops to obtain higher routing flexibility. They analyze conditions for collision-free routing and then propose an efficient dynamic programming approach to search for a feasible schedule with a fixed delivery plan (i.e., with fixed deadlines for deliveries at each machine). The proposed approach is based on a transformation of the original problem to a scheduling problem on parallel processors with deadlines where each job has only one of two distinct processing times. Their proposed approach is also able to generate a schedule that minimizes the maximum tardiness when no feasible delivery plan exists with the given fleet size.

Jaikumar and Solomon (1992) consider an AGV system on a unidirectional network modeled from a real system. They assume that each loop has a safety zone so that AGV interference can be eliminated, and that all jobs are returned to a central warehouse between successive machining steps. They then solve the minimum fleet size and AGV scheduling problem by identifying the minimum number of paths that cover all the nodes on an associated acyclic time-space network. The minimum number of paths is found by solving a related maximal flow problem implied by Dilworth's theorem (Shapiro (1979)). The resulting algorithm is strongly polynomial.

Ganesharajah, Hall and Sriskandarajah (1996) study objectives of minimizing cycle time, fleet size and AGV utilization under three different AGV dispatching policies. They distinguish polynomially solvable sequencing problems, based on the policy, from others that are NP-hard. The relative performance of the policies with respect to the three objectives is discussed.

Many results developed for the robotic cell scheduling problems can be applied to the cases with single loop and zero buffer. For example, when $m = 2$, then the resulting $F2(1)|J > 1|C_{max}$ AGV scheduling problem can be solved in strongly polynomial time by applying the Gilmore and Gomory algorithm (Kise, Shioyama and Ibaraki (1991)). A related study and algorithm can be found in Agnetis, Pacciarelli and Rossi (1994).

Bi-directional flowpath case

Several analytical approaches also consider the cases with a bi-directional network. Kim and Tanchoco (1991) propose a strongly polynomial procedure to find the minimum-delay path for sending an AGV from a source location to a particular machine center without disrupting the scheduled moves of other AGVs. The procedure has a complexity of $O(K^4m^2)$, where K is the fleet size and m is the number of nodes (which can be either a machining center or a pick up station) on the network.

The procedure is developed based on Dijkstra's algorithm. Krishnamurthy, Batta and Karwan (1993) propose a column generation based heuristic approach for bi-directional AGV dispatching. The master problem consists of the makespan and vehicle interference constraints, and contains columns that are routes iteratively generated for each vehicle. The subproblems are constrained shortest path problems, with time-dependent costs on the edges.

Langevin, Lauzon and Riopel (1994) propose a dynamic programming approach to solve the two-AGV scheduling problem with an objective of minimizing the makespan. For each task to be scheduled, a vehicle is selected based on the makespan criterion, and an associated shortest-path problem is solved to design the route of that vehicle. If a new route results in conflicts, then one route is fixed while the other is modified by using the time-window constrained shortest-path algorithm by Desrochers and Soumis (1988). With a similar objective function, a heuristic scheduling algorithm is discussed in the work by Blair, Charnsethikul and Vasques (1987).

3.2.2. Heuristic rules for AGV and machine scheduling

Due to the combinatorial nature of AGV scheduling in general FMS settings, a major focus of researchers in this area has been the design of effective heuristics.

Most heuristic rules prioritize either machines to be served by an AGV or the AGVs to serve a machine, with respect to a given objective function. A seminal work in this area is Egbelu and Tanchoco (1984). Related surveys can be found in Kusiak and Cyrus (1985), Cheng (1987), Co and Tanchoco (1991), Ganesharajah, Hall and Sriskandarajah (1995), and Klein and Kim (1996). Recent studies in this area extend simple scheduling rules to easy-to-use dispatching algorithms (for example, see Egbelu (1987), Slomp, Gaalman and Nawijn (1988)), and to methods for generating machine and AGV schedules simultaneously (Bilge and Ulusoy (1995)).

In general, AGV dispatching rules can be classified into either *work center-initiated* rules or *vehicle-initiated* rules. With a work center-initiated rule, a work center selects an AGV for a delivery operation whenever it finishes an operation. With a vehicle-initiated rule, an AGV selects a pick up when it becomes idle. Egbelu and Tanchoco (1984) show that in a busy system, the vehicle-initiated rules are more effective than the work center-initiated rules. The vehicle-initiated rules can be further classified into *pull-based* and *push-based*. With a pull-based dispatching policy, a vehicle selects a work center with the highest need for job replenishment. Then, a job from the set of candidate jobs which can be sent to that work center is selected. With a push-based policy, a vehicle first selects a job to move and then a work center to which the job should be sent.

Taghaboni and Tanchoco (1988) propose a heuristic procedure to plan conflict-free vehicle routes. Each time a vehicle is selected to make a delivery, all pre-established routes for other vehicles are fixed, and a feasible route for the selected AGV is designed. The intersection conflict is solved on a first-come-first-serve basis. Sabuncuoglu and Hommertzheim (1989, 1992b) test various machine and AGV

scheduling rules against different scheduling criteria via simulation experiments. Sabuncuoglu and Hommertzheim (1992a) also propose a hierarchical approach for real-time on-line AGV scheduling problems. The proposed algorithm uses the information concerning job status to schedule one AGV at a time whenever a scheduling decision is needed or the status of the system is changed. The idea behind their approach is that a job should not be moved if it will have to wait for the next machine on its route. Yim and Linn (1993) use a Petri net based simulation model to investigate the effect of different dispatching rules on the FMS performance. The evaluation results show that there is no significant difference in terms of output rate between push and pull based policies when an FMS system is busy. Bilge and Ulusoy (1995) propose an iterative heuristic procedure to find a feasible schedule for both machine and AGV operations. At each iteration, a new machine schedule is constructed, which is then used to define the time windows for AGV trips.

There is also an emerging line of work on AGV scheduling via artificial intelligence and expert systems techniques. A review of this new line of work can be found in Kusiak (1989). Chung and Fischer (1995) propose an approach based on a Hopfield neural network with simulated annealing. The resulting procedure finds the shortest flow path for an AGV system on a specified routing structure and is able to avoid potential collisions between AGVs.

3.3. The hoist scheduling problem

The hoist scheduling problem can be considered as a special class of $Jm(K)|J > 1|\Pi_{min}$ problems with *tw* and *nwt* constraints. Its most distinctive feature is that the job processing time at each machine is not fixed, but a decision variable whose value must be selected from a given range. In the literature, this is called the *interval processing time*. This interval processing time makes most of the results discussed in sections 3.1 and 3.2 inapplicable. Hoist scheduling problems are often found in electroplating and chemical industries. In the United States, there are more than 4,000 electroplating manufacturers that produce various industrial connectors, printed circuit boards, and switches used in telecommunications hardware.

A typical electroplating line consists of a large number of chemical tanks (machines) in which jobs are submerged. Each job is a barrel carrying identical parts to be plated. Different job types may require a different route and treatment process. Because of the nature of the chemical process, the processing time (i.e., submerging duration) at each tank must be strictly controlled within given lower and upper bounds (i.e., the interval processing time or *tw* constraints). To reduce the damage of oxidization, it is also required that a job, after being removed from a tank, must be directly sent to the next tank on its route and then submerged so that the time exposed to air is minimized. This is referred to as the *nwt* constraints.

Material handling transporters used in such a process are typically computer-programmed hoists, moving along a shared track. To perform a material handling

operation, the assigned hoist will travel to the processing tank, wait if necessary, lift the job up at the scheduled starting time, travel to the next tank on the route and then drop the job into the chemicals there. After that, the hoist is available for another scheduled operation. Due to the size of the job, both a tank and a hoist can hold only one job at a time. Therefore, any schedule that causes either jobs to be "jammed-up" in the same tank or multiple jobs to compete for the same hoist at the same time is not acceptable (i.e., infeasible). Also, due to the single-track constraint, any schedule that requires two or more hoists to cross each other to perform their assignments (i.e., the traffic collision) must be eliminated. A common objective of hoist scheduling in practice is to minimize the cycle time of a repetitive process for producing a given MPS.

3.3.1. $Fm(K)|J = 1, nwt, tw|\Pi_{min}$

When all the jobs are identical (i.e., each MPS consists of a single job), the entire manufacturing process becomes a cyclic no-wait flowline. This special case typically occurs in the mass production of a single product such as computer disks.

Even when $K = 1$, the problem $Fm(1)|J = 1, nwt, tw|\Pi_{min}$ has been proved to be NP-hard in the strong sense (Lei and Wang (1989)). An early analytical approach to this problem is reported by Phillips and Unger (1976), who formulate and then solve the problem as a mixed integer program. Shapiro and Nuttle (1988) propose a branch-and-bound procedure that solves a large number of LP subproblems. They then report the optimal schedules for five industry benchmark problems (Shapiro (1987)). Armstrong, Gu and Lei (1991, 1994) also propose a branch-and-bound procedure. Instead of solving LP subproblems at each node on the search tree, they solve relaxations of LPs. The procedure is able to approach the optimal schedule quickly in terms of the same set of benchmark problems. Different bounding schemes are also discussed in Levner and Ptuskin (1989), Mikhalevich, Beletsky and Monastyrev (1989), Hertz, Mottet and Rochat (1993), Song, Zabinsky and Storch (1993), Lei and Wang (1994), Armstrong, Gu and Lei (1995), and Chu and Proth (1996). A recent work on $Fm(1)|J = 1, nwt, tw|\Pi_{min}$, by Chen, Chu and Proth (1995), is also based on branch and bound. They transform the subproblems encountered during the search process into so-called "cycle time evaluation problems on bi-valued graphs", and then propose a polynomial algorithm for solving each of these subproblems. Another recent branch-and-bound procedure for solving $Fm(1)|J = 1, nwt, tw|\Pi_{min}$ can be found in Ng (1996).

A special case of $Fm(1)|J = 1, nwt, tw|\Pi_{min}$ occurs when the job processing time in each tank is fixed. This special case is shown by Levner and Kats (1995) to be solvable in $O(m^3\log(m))$ time. Another special case of $Fm(1)|J = 1, nwt, tw|\Pi_{min}$ occurs when the unit-cycle is fixed (i.e., the sequence of hoist operations to be performed in each cycle is fixed). In this case, Lei (1993a) shows that the optimal integer schedule that minimizes the cycle time can be derived in $O(m^2\log(m)\log(B))$, where the parameter B stands for the interval between a lower and upper bound on Π_{min}.

When multiple transporters are involved ($K > 1$), the scheduling process becomes more complicated due to the additional single-track or collision-free constraints. Even a formal mathematical programming formulation for this case is difficult to construct since an explicit expression of the single-track constraints depends on the traffic regulation rules in use. Known approaches for this problem are all heuristic-based. Lei and Wang (1990) propose a local optimization approach for the $Fm(2)|J = 1, nwt, tw|\Pi_{min}$ problem. The idea behind this approach is to partition a given line into two non-overlapping zones. Each transporter is then assigned exclusively to a particular zone. Since the movements of each transporter are confined to its own territory, traffic collisions are eliminated. The optimal partition that leads to the minimum system cycle time (i.e., the minimum cycle time that is feasible for both transporters) is then searched for. Manier, Varnier and Baptiste (1994) further generalize this zoned approach to allow single-transporter zones to overlap to accommodate the need for a bi-directional job flow. A set of inequalities that eliminate transporter collisions in overlap sections of zones are developed for the scheduling purpose. This extended approach, however, does not guarantee an optimal solution.

One special case of multiple-hoist scheduling occurs when job processing times are fixed and the single-track constraint is relaxed, or $Fm(K)|J = 1, nwt|\Pi_{min}$. In this case, Lei (1993b) shows that, for any given fleet size, the optimal hoist schedule that minimizes the cycle time can be found by a pseudopolynomial algorithm that solves a sequence of associated assignment problems.

3.3.2. $Fm(K)|J > 1, nwt, tw|\Pi_{min}$

The computational effort required to solve $Fm(K)|J > 1, nwt, tw|\Pi_{min}$ increases tremendously even in the case of $K = 1$, due to the need for both job sequencing and hoist scheduling. Given an MPS of n_{mps} jobs, each requiring processing in m tanks, the total number of candidate cyclic schedules is $O(K^{n_{mps}m} n_{mps}!(n_{mps}m)!)$. This estimation is based on $n_{mps}!$ candidate job sequences, $n_{mps} \cdot m!$ candidate (unit) hoist cycles, and $K^{n_{mps}m}$ candidate assignments of hoists to delivery operations.

All available approaches to this problem are based on either heuristic dispatching rules or expert systems. Representative work using expert systems for the scheduling of hoist movements can be found in Yih (1990), and Yih and Thesen (1991).

To summarize the algorithmic studies on these transporter scheduling problems, tables 2 and 3 in the Appendix present the main results in these areas.

4. Conclusion

In this survey, we have attempted to review some of the recent developments in the theory, the heuristic search methods, and the practice of deterministic scheduling.

In terms of new results in scheduling theory, we have briefly reviewed the scheduling literature with the 1-job-on-r-machine pattern, primarily in the parallel machine environment with r being a positive integer. As discussed in section 1, changing r

from 1 to a positive integer usually increases the complexity of most problems. Note that there is no clear relationship between the complexity of the *nonfix* and the *fix* models. However, both *fix* and *nonfix* models are actually special cases of the *set* model where, as mentioned before, *set* means that a job can choose a set of alternatives where each alternative contains several dedicated machines. Hence, a *set* problem is NP-hard if either the corresponding *fix* problem or *nonfix* problem is NP-hard. Very little work has been done on set problems.

Most of the nonpreemptive problems discussed are NP-hard in the strong sense. Hence, heuristic algorithms are of interest. For the preemptive problems, most polynomial algorithms are based on linear integer programming techniques. Special cases with $p_j = 1$ have also been studied extensively. In either *nonfix* or *fix* models, if we assume $p_j = 1$, then the maximum possible number of job types that can fit on m machines in each unit time interval is a function of m. Hence, for the problem with fixed m, it is a fixed number. Thus, we can use linear integer programming, dynamic programming, or network representations with shortest-path algorithms to solve most problems optimally with complexity that is polynomial in n.

Several open questions were mentioned in section 1. Since most problems are NP-hard, either branch-and-bound techniques, dynamic programming, or heuristic algorithms with an error bound analysis are of interest. Due to a lack of research concerning these approaches, this is a research direction of interest. The nonpreemptive case with different job release times and different machine available time windows is an area for future research.

We have also reviewed briefly deterministic machine scheduling with availability constraints. The literature so far has focused on both the resumable and the nonresumable case. It will be of interest to consider a semi-resumable case where some extra setup time may be required when a job restarts. Extension of the existing models to more complicated job shop and open shop problems is also an interesting research direction. Furthermore, combining machine availability constraints with human resource constraints is another area of study.

In section 2, we presented a fairly complete review of the various search algorithms developed for scheduling during the past decade. These new developments can be classified into neighbourhood search techniques and constraint-guided heuristic search techniques. The former intends to make local improvement of an existing solution, while the latter merely has as goal to find a good feasible solution. In terms of neighbourhood search techniques, we have reviewed the three most commonly used approaches, namely simulated annealing, tabu search, and genetic algorithms.

A number of comparative studies have been conducted to compare these methods with one another and with other more classical mathematical programming type of approaches. There are several important parameters that have to be compared. The most important one is the quality of the solution, i.e., which approach leads to a better answer. Another important aspect is the amount of CPU time that is required to arrive at an acceptable solution. Also, a ratio of these two factors could be of interest. A

third characteristic on which an approach can be judged is the development time or the implementation time. Neighbourhood search approaches usually need less development time than the more classical mathematical programming approaches.

It has to be kept in mind that there are a number of issues that affect the outcome of each comparative study, namely

- the initial solution,
- the setting of parameters,
- the language and manner in which the procedure is coded,
- the platform on which the study is conducted.

Della Croce, Taddei and Volta (1992) make a comparison of various techniques, including the Shifting Bottleneck technique (see Adams, Balas and Zawack (1988)), tabu search and genetic algorithms. In their comparison, genetic algorithms appear to be the least effective technique among the three neighbourhood search techniques. According to the latest computational studies by Aarts, Van Laarhoven and Lenstra (1994) and Morton and Ramnath (1995), tabu search is the most efficient one of the neighbourhood search techniques. Studies comparing neighbourhood search techniques with constraint-guided heuristic search techniques have not yet been conducted.

In section 3, we discussed scheduling applications. There have been a large number of new applications of scheduling in practice; only the area of scheduling of material handling transporters is reviewed here. The three types of transporter scheduling problems (robot, hoist and AGVs) being reviewed share similarities in many aspects. For example, the single-AGV scheduling problem defined on a single-loop layout with no buffer is identical to the single-robot robotic cell scheduling problem. Also, the single-hoist scheduling problem with numerical processing times can be viewed as a special version of the robotic cell scheduling problem, and the multiple-hoist problem can be considered as a special class of AGV scheduling problems. This means that results developed for one may benefit the other.

One important line of work, from a practical point of view, which has not yet received enough attention is the development of effective approaches that address machine and transporter scheduling, as well as facility layout problems in an integrated manner. By machine scheduling, we refer to time-phased scheduling of machine operations, which includes the job sequencing. Clearly, the performance of a system depends on how these issues are addressed and how the operations of machines and transporters are coordinated. While deriving a “global optimal” solution to address all these issues may not be practical, “easy-to-use” heuristic algorithms that will terminate the search within a reasonable time without sacrificing too much in the solution quality is of great interest. With the advances in computing hardware (speed and memory), more and more real scheduling systems can afford a search for the best solution within a reasonably large subspace. Yet, not enough analytical results that guide this type of search have been reported for the scheduling problems reviewed in

section 3. The methodologies reviewed in section 2 of this survey would benefit this purpose.

Another important area is transporter scheduling with dynamic job arrivals. A number of researchers have focussed on this type of scheduling problem. Scheduling in a dynamic environment is more difficult as we can not foresee future jobs. There are many questions waiting to be addressed, e.g., where is the optimal home position of a transporter after a delivery? How to coordinate the machine and material handling operations to minimize the machine, transporter, and job waiting time? How to construct collision-free schedules when jobs arrive dynamically? Several studies on these issues can be found in the work by Bartholdi and Platzman (1989), Kim and Tanchoco (1991), Egbelu (1993), Chang and Egbelu (1995), and Ozdamar and Ulusoy (1995).

Appendix

Table 1

Summary of complexity classification of 1-job-on-r-machine problems.
P: Polynomial; NP: NP-hard in the ordinary sense;
NP!: NP-hard in the strong sense, and $R \leq 2^m - 1$;
NP(?): NP-hard yet open question for the strong sense.

Problem	Complexity	Reference
$P2/nonfix/C_{max}$	NP	Du and Leung (1989)
$P3/nonfix/C_{max}$	NP	Du and Leung (1989)
$P4/nonfix/C_{max}$	NP(?)	Du and Leung (1989)
$P5/nonfix/C_{max}$	NP!	Du and Leung (1989)
$Pm/nonfix, p_j = 1/C_{max}$	$O(n)$	Blazewicz et al. (1986)
$Pm/prmp, nonfix/C_{max}$	P	Blazewicz et al. (1986)
$P2/nonfix/\sum w_j C_j$	NP!	Lee and Cai (1996)
$P2/nonfix, p_j = 1/\sum w_j C_j$	$O(n \log n)$	Lee and Cai (1996)
$P2/nonfix/L_{max}$	NP!	Lee and Cai (1996)
$P2/fix/C_{max}$	$O(n)$	trivial
$P3/fix/C_{max}$	NP!	Blazewicz et al. (1992)
$P2/\text{fix}, r_j/C_{max}$	NP!	Hoogeveen et al. (1994)
$Pm/fix, r_j, p_j = 1/C_{max}$	$O(R2^R n^{R+1})$	Brucker (1995)
$Pm/prmp, fix/C_{max}$	P	Hoogeveen et al. (1994)
$P/fix, p_j = 1/\text{Cmax}$	NP!	Hoogeveen et al. (1994)
$P2/fix/\sum C_j$	NP!	Cai et al. (1996)
$P2/prmp, fix/\sum C_j$	$O(n \log n)$	Cai et al. (1996)
$Pm/fix, r_j, p_j = 1/\sum C_j$	$O(R2^R n^{R+1})$	Brucker (1995)
$Pm/fix, p_j = 1/\sum w_j C_j$	$O(R2^R n^{R+1})$	Brucker (1995)
$P/fix, p_j = 1/\sum C_j$	NP!	Hoogeveen et al. (1994)
$P2/chain, fix, p_j = 1/\sum C_j$	NP!	Hoogeveen (1994)
$P2/fix/L_{max}$	NP!	Section 1.1.2
$Pm/fix, p_j = 1/\sum T_j$	$O(R2^R n^{R+1})$	Brucker (1995)
$Pm/fix, p_j = 1/\sum w_j U_j$	$O(R2^R n^{R+1})$	Brucker (1995)

Table 2

Problem classification, complexity and analytical approaches to the transporter scheduling problems with zero in-process buffers.

Problem	Complexity	Approaches	References
$F2(1)\|J>1\|C_{max}$	$O(n^3)$	Based on Gilmore and Gomory algorithm	Kise, Shioyama and Ibaraki (1991)
$F2(1)\|J>1\|C_{max}$ with job dependent transportation time	NP!		Ganesharajah, Hall and Sriskandarajah (1995)
$F2(1)\|J>1\|\Pi_{min}$	$O(n^3)$	Based on Gilmore and Gomory algorithm	Sethi et al. (1992)
$Fm(1)\|J>1\|\Pi_{min}$	NP!	Heuristics based on Gilmore and Gomory algorithm	Hall, Kamoun and Wan (1994)
$Fm(1)\|J=1\|\Pi_{min}$	$O(m^3)$	Dynamic Programming	Crama and Klundert (1994)
$Fm(1)\|J=1, nwt\|\Pi_{min}$ with numerical processing times	$O(m^3\log(m))$	Interval cutting	Levner and Kats (1996)
$Fm(1)\|J=1, nwt, tw\|\Pi_{min}$	NP!	Branch and Bound	Shapiro and Nuttle (1988) Levner and Ptuskin(1989) Mikhalevich et al. (1989) Song et al. (1993) Hertz et al. (1993) Armstrong et al. (1994) Lei and Wang (1994) Chen, Chu and Proth (1995) Ng (1996)
$Fm(1)\|J=1, nwt, tw\|\Pi_{min}$ with cycle fixed	$O(m^2\log(m)\log(D))$ where D is a data-dependent parameter	Binary search	Lei (1993a)
$Fm(2)\|J=1, nwt, tw\|\Pi_{min}$	NP!	Local optimization based on zoned partition	Lei and Wang (1991)
$Fm(K)\|J=1, nwt\|\Pi_{min}$ with numerical processing times and relaxed single-track constraint	Open	Pseudopolynomial algorithm	Lei (1993b)

Table 3

Known heuristic rules for multiple transporter scheduling.
(*Denotes rules that perform competitively under certain conditions/objectives.)

Single-attribute rules	(*Transporter-status based*)	
	Nearest transporter*	Egbelu and Tanchoco (1984) Mahadevan and Narendran (1990) Klein and Kim (1996)
	Farthest transporter	Egbelu and Tanchoco (1984)
	Longest idle transporter	Egbelu and Tanchoco (1984)
	Most idle/least utilized transporter	Cheng (1987) Mahadevan and Narendran (1990)
	First available transporter*	Cheng (1987) Mahadevan and Narendran (1990)
	Least idle transporter	Cheng (1987)
	Shortest travel time	Egbelu and Tanchoco (1984)
	Longest travel time	Egbelu and Tanchoco (1984)
	(*Machine-status based*)	
	Maximum queue size*	Klein and Kim (1996) Russell and Tanchoco (1984)
	Minimum remaining outgoing queue space	Egbelu and Tanchoco (1984)
	Longest waiting job*	Russell and Tanchoco (1984) Klein and Kim (1996)
	Longest waiting time since the last job arrival	Yim and Linn (1993)
	Maximum remaining incoming queue space	Yim and Linn (1993)
Multiple-attribute rules	Additive weighting factor* (*choose the alternative with the maximum weighted sum of attribute values*)	Hodgson et al. (1987) Klein and Kim (1996)
	Fuzzy multi-attribute decision method (*choose the alternative with the highest outcome under fuzzy ratings*)	Klein and Kim (1996)
	Modified additive weighting factor* (*same as the Additive Weighting Factor, except that normalized attribute values are now used*)	Klein and Kim (1996)
	Max-Max method (*choose the alternative whose maximum objective value is the maximum of all*)	Klein and Kim (1996)
	Multi level/module hierarchy scheduling model* (*choose the alternative based on the information of shop, cell, machines and transporters*)	Akturk and Yilmaz (1996) Sabuncuoglu and Hommertzheim (1992)
	Longest waiting job and minimum remaining outgoing queue space	Yim and Linn (1993)

References

E.H.L. Aarts, P.J.M. van Laarhoven and J.K. Lenstra, Job shop scheduling by local search, INFORMS Journal of Computing 8(1996)302–317.

E.H.L. Aarts, P.J.M. van Laarhoven, J.K. Lenstra and N.L.J. Ulder, A computational study of local search algorithms for job shop scheduling, ORSA Journal of Computing 6(1994)118–125.

J. Adams, E. Balas and D. Zawack, The shifting bottleneck procedure for job shop scheduling, Management Science 34(1988)391–401.

B. Adenso-Dias, Restricted neighbourhood in the tabu-search for the flow shop problem, European Journal of Operational Research 62(1992)27–37.

I. Adiri, J. Bruno, E. Frostig and A.H.G. Rinnooy Kan, Single machine flow-time scheduling with a single breakdown, Acta Informatica 26(1989)679–696.

A. Agnetis, D. Pacciarelli and F. Rossi, Batch scheduling in two-machine cell with swapping devices, Working Paper No. 28.94, Department of Informatica e Sistemistica, University of Rome, 1994.

M.S. Akturk and H. Yilmaz, Scheduling of AGVs in a decision making hierarchy, International Journal of Production Research 34(1996)577–591.

B. Alidaee and A. Ahmadian, Scheduling on a single processor with variable speed, Information Processing Letters (1997), to appear.

B. Alidaee and G.A. Kochenberger, A framework for machine scheduling problems with controllable processing times, Journal of Production and Operations Management (1997), to appear.

G. Amiri and S.F. Smith, A comparative analysis of constraint based scheduling strategies, Technical Report, The Robotics Institute, Carnegie Mellon University, 1992.

E.J. Anderson, C.A. Glass and C.N. Potts, Local search in combinatorial optimization: Applications in machine scheduling, Preprint Series No. OR56, Faculty of Mathematical Studies, University of Southampton, Southampton, England, 1995.

I. Arizono, A. Yamamoto and H. Ohta, Scheduling for minimizing total actual flow time by neural networks, International Journal of Production Research 30(1992)503–511.

R. Armstrong, S. Gu, and L. Lei, Minimizing the cycle time of a just-in-time manufacturing process with a single transporter, Manufacturing Research and Technology 12(1991)237–246.

R. Armstrong, S. Gu, and L. Lei, A bounding scheme for deriving the minimal cycle time of a single-transporter *N*-stage process with time-window constraints, European Journal of Operational Research 75(1994)130–140.

R. Armstrong, S. Gu and L. Lei, Greedy algorithm to determine the number of transporters in a cyclic electroplating process, IIE Transactions 27(1995)1–9.

R. Armstrong, S. Gu and L. Lei, An efficient algorithm for a class of two-resource allocation problems, ORSA Journal on Computing (1997), to appear.

R. Armstrong, S. Gu and L. Lei, Solving a class of two-resource allocation problems by equivalent load method, Journal of the Operational Research Society (1997), to appear.

H. Aytug, S. Bhattacharyya, G.J. Koehler and J.L. Snowdon, A review of machine learning in scheduling, IEEE Transactions on Engineering Management 41(1994)165–171.

K.R. Baker, *Elements of Sequencing and Scheduling*, Amos Tuck School, Dartmouth College, Hanover, NH, 1995.

K.R. Baker and G.D. Scudder, Sequencing with earliness and tardiness penalties: A review, Operations Research 38(1990)22–36.

J.W. Barnes and M. Laguna, Solving the multiple machine weighted flow time problem using tabu-search, IIE Transactions 25(1993)121–128.

J.W. Barnes and M. Laguna, A tabu search experience in production scheduling, Annals of Operations Research 41(1993)141–156.

J.W. Barnes, M. Laguna and F. Glover, An overview of tabu-search approaches to production scheduling problems, in: *Intelligent Scheduling Systems*, Brown and Scherer, eds., Kluwer Academic, 1995, pp. 101–128.

J. Bartholdi, III and L. Platzman, Decentralized control of automated guided vehicles on a simple loop, IIE Transactions 21(1989)76–81.
J. Bean, Genetics and random keys for sequencing and optimization, ORSA Journal of Computing 6 (1994)154–160.
L. Bianco, J. Blazewicz, P. Dell'Olmo and M. Drozdowski, Preemptive multiprocessor task scheduling with release times and time windows, Annals of Operations Research (1997), this volume.
L. Bianco, J. Blazewicz, P. Dell'Olmo and M. Drozdowski, Preemptive scheduling of multiprocessor tasks on the dedicated processors system subject to minimal lateness, Information Processing Letters 46(1993)109–113.
L. Bianco, J. Blazewicz, P. Dell'Olmo and M. Drozdowski, Scheduling multiprocessor tasks on a dynamic configuration of dedicated processors, Annals of Operations Research 58(1995)493–517.
C. Bierwirth, A generalized permutation approach to job shop scheduling with genetic algorithms, OR Spektrum 17(1995)87–92.
U. Bilge and G. Ulusoy, A time window approach to simultaneous scheduling of machines and material handling system in an FMS, Operations Research 43(1995)1058–1070.
E.L. Blair, P. Charnsethikul and A. Vasques, Optimal routing of driverless vehicles in a flexible material handling system, Material Flow 4(1987)73–83.
J. Blazewicz, R. Burkard, G. Finke and G. Woeginger, Vehicle scheduling in two cycle flexible manufacturing systems, Mathematics and Computer Modeling 20(1994)19–31.
J. Blazewicz, P. Dell'Olmo, M.Drozdowski and M.G. Speranza, Scheduling multiprocessor tasks on three dedicated processors, Information Processing Letters 41(1992)275–280.
J. Blazewicz, M.Drabowski and J. Weglarz, Scheduling multiprocessor tasks to minimize schedule length, IEEE Transactions on Computers C-35(1986)389–393.
J. Blazewicz, M.Drozdowski, G. Schmidt and D. de Werra, Scheduling independent two-processor tasks on a uniform duo-processor system, Discrete Applied Mathemaatics 28(1990)11–20.
J. Blazewicz, M.Drozdowski, G.Schmidt and D. de Werra, Scheduling independent multiprocessor tasks on a uniform *k*-processor system, Parallel Computing 20(1994)15–28.
J. Blazewicz, M.Drozdowski and J.Weglarz, Scheduling multiprocessor tasks – a survey, International Journal of Microcomputer Applications 13(1994)89–97.
J. Blazewicz, K. Ecker, G. Schmidt and J. Weglarz, *Scheduling in Computer and Manufacturing Systems*, 2nd ed., Springer, New York, 1994.
J. Blazewicz, K. Ecker, E. Pesch, G. Schmidt and J. Weglarz, *Scheduling Computer and Manufacturing Processes*, Springer, Berlin/New York, 1996.
J. Blazewicz, H.A. Eiselt, G. Finke, G. Laporte and J. Weglarz, Scheduling tasks and vehicles in a flexible manufacturing system, The International Journal of Flexible Manufacturing Systems 4(1991)5–16.
J. Blazewicz, J. Weglarz and W. Drabowski, Scheduling independent 2-processor tasks to minimize scheduling length, Information Processing Letters 18(1984)267–273.
Y.A. Bozer and M.M. Srinivasan, Tandem configurations for automated guided vehicle systems and the analysis of single vehicle loops, IIE Transactions 23(1991)72–82.
D.E. Brown and W.T. Scherer (eds.), *Intelligent Scheduling Systems*, Kluwer Academic, Boston, MA, 1995.
P. Brucker, J. Hurnik and F. Werner, Improving local search heuristics for some scheduling problems: Part I, Discrete Applied Mathematics (1996), to appear.
P. Brucker, J. Hurnik and F. Werner, Improving local search heuristics for some scheduling problems: Part II, Discrete Applied Mathematics (1996), to appear.
P. Brucker, *Scheduling Algorithms*, Springer, New York, 1995.
P. Brucker and A. Krämer, Shop scheduling problems with multiprocessor tasks on dedicated processors, Annals of Operations Research 57(1995)13–27.
P. Brucker and A. Krämer, Polynomial algorithms for resource-constrained and multiprocessor task scheduling problems, European Journal of Operational Research 90(1996)214–226.

X. Cai, C.-Y. Lee and C.L. Li, Scheduling multiprocessor tasks with prespecified processor allocations, (1996), submitted.

S.H. Chang and P.J. Egbelu, Dynamic relative positioning of AGVs in a loop layout to minimize the mean system response time, Working Paper, Department of Industrial and Manufacturing Engineering, Pennsylvania State University, 1995.

H. Chen, C. Chu and J.M. Proth, Cyclic hoist scheduling based on graph theory, in: *INRIA/IEEE Symposium on Emerging Tech. and Factory Automation*, 1995, pp. 451–459.

C.-C. Cheng and S.F. Smith, Applying constraint satisfaction techniques to job shop scheduling, Annals of Operations Research (1997), this volume.

T.C.E. Cheng, A simulation study of automated guided vehicle dispatching, Robotics and Computer-Integrated Manufacturing 3(1987)335–338.

H.K. Chu, P.J. Egbelu and C. Wu, Advisor: A computer-aided material handling equipment selection system, International Journal of Production Research 33(1995)3311–3329.

C. Chu and J.M. Proth, Single machine scheduling with chain structured precedence constraints and separation time windows, IEEE Trans. on Robotics and Automation (1996), to appear.

Y. Chung and G.W. Fisher, A neural algorithm for finding the shortest flow path for an automated guided vehicle system, IIE Transactions 27(1995)773–783.

C.G. Co and J.M.A. Tanchoco, A review of research on AGV vehicle management, Engineering Costs and Production Economics 21(1991)35–42.

Y. Crama, Combinatorial optimzation models for production scheduling in automated manufacturing systems, *EURO 20th Anniversary Volume*, 1996, to appear.

Y. Crama and J. Klundert, Cyclic scheduling of identical parts in a robotic cell, Operations Research (1996), to appear.

H.A.J. Crauwels, C.N. Potts and L.N. van Wassenhove, Local search heuristics for single machine scheduling with batching to minimize total weighted completion time, Annals of Operations Research (1997), this volume.

R.L. Daniels, B.J. Hoopes and J.B. Mazzola, Scheduling parallel manufacturing cells with resource flexibility, Management Science 42(1996)1260–1276.

R.L. Daniels, B.J. Hoopes and J.B. Mazzola, An analysis of heuristics for the parallel machine flexible-resource scheduling problem, Annals of Operations Research (1997), this volume.

R.L. Daniels and J.B. Mazzola, A tabu-search heuristic for the flexible-resource flow shop scheduling problem, Annals of Operations Research 41(1993)207–230.

R.L. Daniels and J.B. Mazzola, Flow shop scheduling with resource flexibility, Operations Research 42(1994)504–522.

S. Dauzère-Pérès and J. Paulli, A global tabu search procedure for the general multi-processor job-shop scheduling problem, Annals of Operations Research (1997), this volume.

F. Della Croce, R. Tadei and G. Volta, A genetic algorithm for the job shop problem, Computers and Operations Research 22(1995)15–24.

M. Dell'Amico and M. Trubian, Applying tabu-search to the job shop scheduling problem, Annals of Operations Research 41(191993)231–252.

M. Desrochers and F. Soumis, A generalized permanent labelling algorithm for the shortest path with time windows, INFOR 26(1988)193–214.

G. Dobson and U. Karmarkar, Simultaneous resource scheduling to minimize weighted flow times, Operations Research 37(1989)592–600.

G. Dobson and I. Khosla, Simultaneous resource scheduling with batching to minimize weighted flow times, IIE Transactions 27(1995)587–598.

U. Dorndorf and E. Pesch, Evolution based learning in a job shop scheduling environment, Computers and Operations Research 22(1995)25–40.

Z. Du and J.Y.-T. Leung, Complexity of scheduling parallel task systems, SIAM J. Discr. Math. 2(1989) 473–487.

P.J. Egbelu and J.M.A. Tanchoco, Characterization of automated guided vehicle dispatching rules, International Journal of Production Research 22(1984)359–374.
P.J. Egbelu, Pull versus push strategy for automated guided vehicle load movement in a batch manufacturing system, Journal of Manufacturing Systems 6(1987)271–180.
P.J. Egbelu and N. Roy, Material flow control in AGV/unit load based production lines, International Journal of Production Research 26(1988)81–94.
P.J. Egbelu, Positioning of automated guided vehicles in a loop layout to improve response time, European Journal of Operational Research 71(1993)32–44.
T. Ganesharajah, N. Hall and C. Sriskandarajah, Design and operational issues in AGV-served manufacturing systems, Working Paper, University of Toronto, 1995.
T. Ganesharajah, N. Hall and C. Sriskandarajah, AGV-served manufacturing systems: Scheduling and design in loop layouts, Working Paper, University of Toronto, 1996.
M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, New York, 1979.
P.C. Gilmore and R.E. Gomory, Sequencing a one-variable machine: A solvable case of the traveling salesman problem, Operations Research 12(1964)655–679.
C.A. Glass, C.N. Potts and P. Shade, Unrelated parallel machine scheduling using local search, Mathematical and Computer Modelling 20(1994)41–52.
F. Glover, Tabu-search, Part I, ORSA Journal of Computing 1(1989)190–206.
F. Glover, Tabu-search, Part II, ORSA Journal of Computing 2(1990)4–32.
F. Glover, Tabu thresholding: Improved search by nonmonotonic trajectories, ORSA Journal of Computing 7(1995)426–442.
F. Glover, E. Taillard and D. de Werra, A user's guide to tabu search, Annals of Operations Research 41(1993)3–28.
S.C. Graves, A review on production scheduling, Operations Research 29(1981)646–676.
C. Gupta, Y.P. Gupta and A. Kumar, Minimizing flow time variance in a single machine using genetic algorithms, European Journal of Operational Research 70(1993) 289–303.
N.G. Hall, H. Kamoun and C. Sriskandarajah, Scheduling robotic cells: Large cells, Working Paper, The Ohio State University, 1993.
N.G. Hall, H. Kamoun and C. Sriskandarajah, Scheduling in robotic cells: Classification, two machine cells, identical parts, Operations Research (1996a), to appear.
N.G. Hall, H. Kamoun and C. Sriskandarajah, Scheduling in robotic cells: Heuristics and cell design, (revised for) Operations Research (1996b).
N.G. Hall, H.K. Kamoun and H. Wan, Scheduling large robotic cells, Working Paper, College of Business, The Ohio State University, 1994.
N.G. Hall, M.E. Posner and C.N. Potts, Scheduling with finite output buffers, (revised for) Operations Research (1993).
N.G. Hall, M.E. Posner and C.N. Potts, Scheduling with finite capacity input buffers, (revised for) Operations Research (1994).
N.G. Hall, M.E. Posner and C.N. Potts, Preemptive scheduling with finite capacity input buffers, Annals of Operations Research (1997), this volume.
N.G. Hall, C. Potts and C. Sriskandarajah, Parallel machine scheduling with a common server, Working Paper, College of Business, The Ohio State University, 1994.
N.G. Hall and C. Sriskandarajah, A survey of machine sceduling problems with blocking and no-wait in process, Operations Research 44(1996)510–525.
J.W. Herrmann, C.-Y. Lee and J. Hinchman, Global job shop scheduling with a genetic algorithm, Production and Operations Management 4(1995)30–45.
J. Herrmann, C.-Y. Lee and J. Snowdon, A classification of static scheduling problems, in: *Complexity in Numerical Optimization*, P.M. Pardalos, ed., World Scientific, 1993, pp. 203–253.
A. Hertz, Y. Mottet and Y. Rochat, On a scheduling problem in a robotized analytical system, Working Paper No. ORWP 92/17, Ecole Polytechnique Fédérale, Lausanne, 1993.

T.J. Hodgson, R.E. King, S.K. Monteith and S.R. Shultz, Developing control rules for an AGVS using Markov decision processes, Material Flow 4(1987)85–96.
J.H. Holland, Genetic algorithms and the optimal allocation of trials, SIAM Journal on Computing 2(1973)88–105.
J.H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, MI, 1975.
J.A. Hoogeveen, S.L. van de Velde and B. Deltman, Complexity of scheduling multiprocessor tasks with prespecified allocations, Discrete Applied Mathematics 55(1994)259–272.
R. Hubscher and F. Glover, Applying tabu-search with influential diversification to multiprocessor scheduling, Computers and Operations Research 21(1994)877–884.
J. Hurink, B. Jurisch and M. Thole, Tabu search for the job shop scheduling problem with multi-purpose machines, OR Spektrum 15(1994)205–215.
R. Jaikumar and M.M. Solomon, Dynamic scheduling of automated guided vehicles for a certain class of systems, Journal of Manufacturing Systems 9(1992)315–323.
W. Jeng, J. Lin and U. Wen, Algorithms for sequencing robot activities in a robot-centered parallel-processor workcell, Computers and Operations Research 20(1993)185–197.
M. Kaspi and B. Montreuil, On the scheduling of identical parallel processes with arbitrary initial processor available times, Research Report 88-12, School of Industrial Engineering, Purdue University, 1988.
C.W. Kim and J.M.A. Tanchoco, Conflict-free shortest-time bidirectional AGV routing, International Journal of Production Research 29(1991)2377–2391.
R.E. King, T.J. Hodgson and F.W. Chafee, Robot task scheduling in a flexible manufacturing cell, IIE Transactions 25(1993)80–87.
S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, Optimization by simulated annealing, Science 220(1983) 671–680.
H. Kise, T. Shioyama and T. Ibaraki, Automated two machine flowshop scheduling: A solvable case, IIE Transactions 23(1991)10–16.
C.M. Klein and J. Kim, AGV dispatching, International Journal of Production Research 34(1996) 95–110.
A. Krämer, Scheduling multiprocessor tasks on dedicated processors, Ph.D. Thesis, FB Mathematik/ Informatik, Universität Osnabrück, 1995.
H. Krawczyk and M. Kubale, An approximation algorithm for diagnostic test scheduling in multicomputer systems, IEEE Transactions on Computers 34(1985)869–872.
N. Krishnamurthy, R. Batta and M. Karwan, Developing conflict-free routes for automated guided vehicles, Operations Research 41(1993)1077–1090.
M. Kubale, The complexity of scheduling independent two-processor tasks on dedicated processors, Information Processing Letters 24(1987)141–147.
A. Kusiak and P. Cyrus, Routing and scheduling of automated guided vehicles, in: *Proceedings of the 8th International Conference on Production Research*, 1985, pp. 247–251.
A. Kusiak, Scheduling automated manufacturing systems: Knowledge-based approach, in: *Proceedings of the 3rd ORSA/TIMS Conference on Flexible Manufacturing Systems: Operations Research Models and Applications*, K.E. Stecke and R. Suri, eds., 1989, pp. 377–382.
M. Laguna, J.W. Barnes and F. Glover, Tabu search methods for a single machine scheduling problem, Journal of Intelligent Manufacturing 2(1991)63–74.
M. Laguna, J.W. Barnes and F. Glover, Intelligent scheduling with tabu-search: An application to jobs with linear delay penalties and sequence-dependent setup costs and times, Journal of Applied Intelligence 3(1993)159–172.
M. Laguna and F. Glover, Integrating target analysis and tabu-search for improved scheduling systems, Expert Systems with Applications 6(1993)287–297.
M. Laguna and J.L. Gonzalez Velarde, A search heuristic for Just-In-Time scheduling in parallel machines, Journal of Intelligent Manufacturing 2(1991)253–260.

A. Langevin, D. Lauzon and D. Riopel, Dispatching, routing and scheduling of two automated guided vehicles in a flexible manufacturing system, GERAD Working Paper, Ecole Polytechnique de Montréal, 1994.

E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D. Shmoys, Sequencing and scheduling: Algorithms and complexity, in: *Handbook in Operations Research and Management Science*, S.S. Graves, A.H.G. Rinnooy Kan and P. Zipkin, eds., Vol. 4, North-Holland, New York, 1993, pp. 445–522.

G. Lawton, Genetic algorithms for schedule optimization, AI Expert (May 1992)23–27.

C.-Y. Lee, Parallel machines scheduling with non-simultaneous machine available time, Discrete Applied Mathematics 30(1991)53–61.

C.-Y. Lee, Machine scheduling with an availability constraint, Journal of Global Optimization, special issue on Optimization on Scheduling Application, 9(1996a)395–416.

C.-Y. Lee, Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint, Operations Research Letters (1996b), to appear.

C.-Y. Lee and X. Cai, Scheduling multiprocessor tasks without prespecified processor allocations, 1996, submitted.

C.-Y. Lee and S.D. Liman, Single machine flow-time scheduling with scheduled maintenance, Acta Informatica 29(1992)375–382.

C.-Y. Lee and S.D. Liman, Capacitated two-parallel machines scheduling to minimize sum of job completion times, Discrete Applied Mathematics 41(1993)211–222.

C.-Y. Lee, S. Piramuthu and Y.K. Tsai, Job shop scheduling with a genetic algorithm and machine learning, Int. J. Production Research (1995), to appear.

C.-Y. Lee and G. Vairaktarakis, Complexity of single machine hierarchy scheduling: A survey, in: *Complexity in Numerical Optimization*, P.M. Pardalos, ed., World Scientific, New Jersey, 1993, pp. 269–298.

Y.H. Lee, K. Bhaskanan and M. Pinedo, A heuristic to minimize total weighted tardiness with sequence dependent setups, IIE Scheduling and Logistics (1997), to appear.

L. Lei, An $O(N^2 \log N \log B)$ algorithm for determining the optimal integer cyclic transportation schedule with a given route, Computers and Operations Research 20(1993a)807–816.

L. Lei, Adaptive control of feasible cycle times in a processing line with multiple transporters, in: *Proceedings of the International Conference on Pacific Region Management*, 1993b, pp. 123–131.

L. Lei, R. Armstrong and S. Gu, Minimizing the fleet size for a cyclic transportation schedule with dependent time windows and single-track constraints, Operations Research Letters 14(1993)91–98.

L. Lei, R. Armstrong and S. Gu, Determining the number of transporters for a cyclic transportation schedule, Applications of Management Science: Engineering 9(1996), to appear.

L. Lei and T.J. Wang, A proof on the NP-completeness of the hoist scheduling problems, Working Paper No. 89-06, Graduate School of Management, Rutgers University, 1989.

L. Lei and T.J. Wang, The minimum common-cycle algorithm for cyclic scheduling of two hoists with time window constraints, Management Science 37(1991)1629–1639.

L. Lei and T.J. Wang, On the optimal cyclic schedules of single hoist electroplating processes, IIE Transactions 26(2)(1994)25–33.

V.J. Leon and R. Balakrishnan, Strength and adaptibility of problem-space based neighbourhoods for resource constrained scheduling, OR Spektrum 17(1995)173–182.

E. Levner and V. Kats, A polynomial algorithm for a cyclic robotic flowshop problem, (revised for) European Journal of Operational Research (1996).

E. Levner and A.S. Ptuskin, The construction of cyclic schedules for fuzzy durations of operations, Engineering Cybernetics: Soviet J. of Computer and Systems Science 3(1989)10–14.

C.-L. Li, X. Cai and C.-Y. Lee, Machine scheduling with multiple-job-on-one-machine pattern (1996), submitted.

S. Liman, Scheduling with capacities and due-dates, Ph.D. Dissertation, Industrial and Systems Engineering Department, University of Florida, 1991.

B. Mahadevan and T.T. Narendren, Design of automated guided vehicle-based material handling system for a flexible manufacturing system, International Journal of Production Research 29(1990) 1611–1622.
M. Manier, C. Varnier and P. Baptiste, A constraint logic programming approach to a cyclic multi-hoists scheduling problem, in: *Proceedings of the 4th International Conference of Project Management and Scheduling*, 1994.
H. Matsuo, J. Shang and R.S. Sullivan, A crane scheduling problem in a computer-integrated manufacturing environment, Management Science 37(1991)587–606.
H. Matsuo, C.J. Suh and R.S. Sullivan, A controlled search simulated annealing method for the single machine weighted tardiness problem, Working Paper 87-12-2, Department of Management, University of Texas at Austin, TX, 1987.
H. Matsuo, C.J. Suh and R.S. Sullivan, A controlled search simulated annealing method for the general job shop scheduling problem, Working Paper 03-44-88, Department of Management, University of Texas at Austin, TX, 1988.
E. Mayrand, P. Lefrancois, O. Kettani and M.-H. Jobin, A genetic search algorithm to optimize job sequencing under a technological constraint in a rolling-mill facility, OR Spektrum 17(1995)183–192.
V.S. Mikhalevich, S.A. Beletsky and A.P. Monastyrev, Optimization of multi-stage cyclic service of a production line by a transmanipulator, Cybernetics 24(1988/1989)500–511.
J. Mittenthal, M. Raghavachari and A.I. Rana, A hybrid simulated annealing approach for single machine scheduling problems with non-regular penalty functions, Computers and Operations Research 20 (1993)103–111.
T.E. Morton and D.W. Pentico, *Heuristic Scheduling Systems*, Wiley, New York, 1993.
T.E. Morton and P. Ramnath, Guide forward search in tardiness scheduling of large one machine problems, in: *Intelligent Scheduling Systems*, Brown and Scherer, eds., Kluwer Academic, 1995.
G. Mosheiov, Minimizing the sum of job completion times on capacitated parallel machines, Math. Comput. Modelling 20(1994)91–99.
W.C. Ng, A branch and bound algorithm for hoist scheduling of a circuit board production line, The International Journal of Flexible Manufacturing Systems 8(1996)45–65.
S.J. Noronha and V.V.S. Sarma, Knowledge-based approaches for scheduling problems: A survey, IEEE Transactions on Knowledge and Data Engineering 3(1991)160–171.
E. Nowicki and C. Smutnicki, A fast taboo search algorithm for the job shop problem, Preprint 8/93, Instytut Cybernetyki Technicznej, Politechniki Wroclawskiej, Wroclaw, 1993.
E. Nowicki and C. Smutnicki, A fast taboo search algorithm for the flow shop problem, Preprint 8/94, Instytut Cybernetyki Technicznej, Politechniki Wroclawskiej, Wroclaw, 1994.
W.M. Nowijn, W. Kern and S. Bass, Scheduling jobs with release times on a machine with finite storage, European Journal of Operational Research 74(1994)120–127.
F.A. Ogbu and D.K. Smith, The application of the simulated annealing algorithm to the solution of the flow shop problem, Computers and Operations Research 17(1990)243–253.
L. Ozdamar and G. Ulusoy, A survey on the resource-constrained project scheduling problem, IIE Transactions 27(1995)574–586.
E. Pesch, *Learning in Automated Manufacturing – A Local Search Approach*, Physica-Verlag, Heidelberg, 1994.
L.W. Phillips and P.S. Unger, Mathematical programming solution of a hoist scheduling program, AIIE Transactions 8(1976)219–225.
M.L. Pinedo, *Scheduling: Theory, Algorithms and Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1995.
M.L. Pinedo and M. Singer, A shifting bottleneck heuristic for minimizing the total weighted tardiness in job shops, Technical Report IEOR 96-12, Columbia University, New York, 1996.
J. Plehn, Preemptive scheduling of independent jobs with release times and deadlines on a hypercube, Information Processing Letters C-34(1990)161–166.
C.N. Potts and L.N. Van Wassenhove, Integrating scheduling with batching and lot-sizing: A review of algorithms and complexity, Journal of Operational Research Society 43(1992)395–406.

U.S. Rao and P.L. Jackson, Subproblems in identical jobs cyclic scheduling: Properties, complexity and solution approaches, Working Paper, Cornell University, 1993.
C.R. Reeves, Improving the efficiency of tabu search for machine sequencing problems, Journal of the Operational Research Society 44(1993)375–382.
C.R. Reeves, A genetic algorithm for flow shop sequencing, Computers and Operations Research 22 (1995)5–13.
D. Riopel and A. Langevin, Optimizing the location of material transfer stations within layout analysis, International Journal of Production Economics 22(1991)169–176.
R. Russell and T.M.A. Tanchoco, An evaluation of vehicle dispatching rules and their effect on shop performance, Material Flow 1(1984)271–280.
I. Sabuncuoglu and D.L. Hommertzheim, An investigation of machine and AGVscheduling rules in an FMS, in: *Proceedings of the 3rd ORSA/TIMS Conference on Flexible Manufacturing Systems: Operations Research Models and Applications*, K.E. Stecke and R. Suri, eds., 1989, pp. 261–266.
I. Sabuncuoglu and D.L. Hommertzheim, Dynamic dispatching algorithm for scheduling machines and automated guided vehicles in a flexible manufacturing system, International Journal of Production Research 30(1992a)1059–1079.
I. Sabuncuoglu and D.L. Hommertzheim, Experimental investigation of machine and AGV scheduling rules against the mean-flow time criterion, International Journal of Production Research 30(1992b) 1617–1635.
S. Sahni, Approximation algorithms for the 0–1 knapsack problem, Journal of the Association for Computing Machinery 20(1975)115–124.
G. Schmidt, Scheduling on semi-identical processors, ZOR – Zeitschrift für Operations Research 28 (1984)153–162.
G. Schmidt, Scheduling independent tasks with deadlines on semi-identical processors, Journal of the Operational Research Society 39(1988)271–277.
S.P. Sethi, C. Sriskandarajah, G. Sorger, J. Blazewicz and W. Kubiak, Sequencing of parts and robot moves in a robotic cell, International Journal of Flexible Manufacturing Systems 4(1992)331–358.
G.W. Shapiro, Hoist scheduling for a PCB electroplating facility, unpublished M.S. Thesis, North Carolina University, 1985.
G.W. Shapiro and H.W. Nuttle, Hoist scheduling for a PCB electroplating facility, IIE Transactions 20(1988)157–167.
J. Shapiro, *Mathematical Programming: Structures and Algorithms*, Wiley, 1979.
J. Slomp, G.J.G. Gaalman and W.M. Nawijn, Quasi on-line scheduling procedures for flexible manufacturing systems, International Journal of Production Research 26(1988)585–598.
D. Sinriech and J.M.A. Tanchoco, The centroid projection method for locating pick-up and delivery stations in single-loop AGV systems, Journal of Manufacturing Systems 11(1992)297–307.
S.F. Smith, Knowledge-based production management: Approaches, results and prospects, Production Planning and Control 3(1992)350–380.
S.F. Smith, private communication, 1996.
S.F. Smith, N. Muscettola, D.C. Matthys, P.S. Ow and Y. Potvin, OPIS: An opportunistic factory scheduling system, in: *Proceedings of the 3rd International Conference on Industrial and Expert Systems* (*IEA/AIE 90*), Charleston, SC, 1990.
W. Song, Z.B. Zabinsky and R.L. Storch, An algorithm for scheduling a chemical processing tank line, Production Planning and Control 4(1993)323–332.
C. Sriskandarajah, N.G. Hall and H. Kamoun, Scheduling in robotic cells: Complexity and steady state analysis, Working Paper, The Ohio State University, 1994.
K.E. Stecke, Formulation and solution of nonlinear integer production planning problems for flexible manufacturing systems, Management Science 29(1983)273–288.
H.I. Stern and G. Vitner, Scheduling parts in a combined production-transportation work cell, Journal of the Operational Research Society 41(1990)625–632.

R.H. Storer, S.D. Wu and R. Vaccari, New search spaces for sequencing problems with application to job shop scheduling, Management Science 38(1992)1495–1509.
R.H. Storer, S.D. Wu and R. Vaccari, Problem and heuristic space search strategies for job shop scheduling, ORSA Journal of Computing 7(1995)453–467.
F. Taghaboni and J.M.A. Tanchoco, A LISP-based controller for free-ranging automated guided vehicle systems, International Journal of Production Research 26(1988)173–188.
E. Taillard, Parallel taboo search techniques for the job shop scheduling problem, ORSA Journal of Computing 6(1994)108–117.
V.S. Tanaev, V.S. Gordon and Y.M. Shafransky, *Scheduling Theory. Single-Stage Systems*, Kluwer Academic, The Netherlands, 1994.
V.S. Tanaev, Y.N. Sotskov and V.A. Strusevich, *Scheduling Theory. Multi-Stage Systems*, Kluwer Academic, The Netherlands, 1994.
J.A. Tompkin and J.A. White, *Facilities Planning*, Wiley, New York, 1984.
M.A. Trick, Scheduling multiple variable-speed machines, Operations Research 42(1994)234–248.
D. Trietsch and K.R. Baker, Basic techniques for lot streaming, Operations Research 41(1993)1065–1076.
A.J. Vakharia and Y.-L. Chang, A simulated annealing approach to scheduling a manufacturing cell, Naval Research Logistics 37(1990)559–577.
P.J.M. Van Laarhoven, E.H.L. Aarts and J.K. Lenstra, Job shop scheduling by simulated annealing, Operations Research 40(1992)113–125.
R.J.M. Vaessens, E.H.L. Aarts and J.K. Lenstra, Job shop scheduling by local search, INFORMS Journal of Computing 8(1996)302–317.
B. Veltmann, B.J. Lageweg and J.K. Lenstra, Multiprocessor scheduling with communication delays, Parallel Computing 16(1990)173–182.
Y. Yih, Trace-driven knowledge acquisition (TDKA) for rule-based real-time scheduling systems, Journal of Intelligent Manufacturing 1(1990)217–230.
Y. Yih and A. Thesen, Semi-Markov decision models for real-time scheduling, International Journal of Production Research 29(1991)2331–2346.
D. Yim and R. Linn, Push and pull rules for dispatching automated quided vehicles in a flexible manufacturing system, International Journal of Production Research 31(1993)43–57.
S. Webster and K.R. Baker, Scheduling groups of jobs on a single machine, Operations Research 43 (1996)693–702.