Survey of Job Shop Scheduling Techniques

Albert Jones, Ph.D.
National Institute of Standards and Technology
Building 220, Room A127
Gaithersburg, MD 20899-0001
Phone: (301) 975-3554, Fax: (301) 258-9749
jonesa@cme.nist.gov

Luis C. Rabelo, Ph.D., Professor
Industrial and Manufacturing Engineering Department
California Polytechnic State University
San Luis Obispo, CA 93407
Phone: (216) 447-5203, Fax: (216) 447-5196
email: rabelo@brk.bfg.com

Scheduling has been defined as "the art of assigning resources to tasks in order to insure the termination of these tasks in a reasonable amount of time" (1). According to French (2), the general problem is to find a sequence, in which the jobs (e.g., a basic task) pass between the resources (e.g., machines), which is a feasible schedule, and optimal with respect to some performance criterion. Graves (3) introduced a functional classification scheme for scheduling problems. This scheme categorizes problems using the following dimensions:

1. Requirement generation,
2. Processing complexity,
3. Scheduling criteria,
4. Parameter variability,
5. Scheduling environment.

Based on requirements generation, a manufacturing shop can be classified as an open shop or a closed shop. An open shop is "build to order", and no inventory is stocked. In a closed shop the orders are filled from existing inventory.

Processing complexity refers to the number of processing steps and workstations associated with the production process. This dimension can be decomposed further as follows:

1. One stage, one processor
2. One stage, multiple processors,
3. Multistage, flow shop,
4. Multistage, job shop.

The one stage, one processor and one stage, multiple processors problems require one processing step that must be performed on a single resource or multiple resources respectively. In the multistage, flow shop problem each job consists of several tasks, which require processing by distinct resources; but there is a common route for all jobs. Finally, in the multistage, job shop situation, alternative resource sets and routes can be chosen, possibly for the same job, allowing the production of different part types.

The third dimension, scheduling criteria, states the desired objectives to be met. "They are numerous, complex, and often conflicting" (2). Some commonly used scheduling criteria include the following:

1. Minimize total tardiness,
2. Minimize the number of late jobs,
3. Maximize system/resource utilization,
4. Minimize in-process inventory,
5. Balance resource usage,
6. Maximize production rate.

The fourth dimension, parameters variability, indicates the degree of uncertainty of the various parameters of the scheduling problem. If the degree of uncertainty is insignificant (i.e., "the uncertainty in the various quantities is several orders of magnitude less than the quantities themselves") (2), the scheduling problem could be called deterministic. For example, the expected processing time is six hours, and the variance is one minute. Otherwise, the scheduling problem could be called stochastic. In this chapter, we consider both.

The last dimension, scheduling environment, defined the scheduling problem as static or dynamic. Scheduling problems in which the number of jobs to be considered and their ready times are available are called static. On the other hand, scheduling problems in which the number of jobs and related characteristics change over time are called dynamic. In this chapter, we are concerned primarily with a dynamic environment.

A large number of approaches to the modeling and solution of these scheduling problems have been reported in the Operations Research (OR) literature, with varying degrees of success. These approaches revolve around a series of technological advances that have occurred over the last 35 years. These include mathematical programming, dispatching rules, expert systems, neural networks, genetic algorithms, fuzzy logic, and inductive learning. In this chapter, we will focus on dynamic, job shop problems that are both deterministic and stochastic. We will take an evolutionary view and describe how these technologies have been applied to these problems. To do this, we discuss a few of the most important contributions in each of these technology areas and the most recent trends.

1.0 Mathematical techniques

Mathematical programming has been applied extensively to job shop scheduling problems. Problems have been formulated using integer programming (4) (5), mixed-integer programming (6) (7), and dynamic programming (8). Until recently, the use of these approaches has been limited because scheduling problems belong to the class of NP-complete problems. To overcome these deficiencies, a group of researchers began to decompose the scheduling problem into a number of subproblems, proposing a number of techniques to solve them. In addition, new solution techniques, more powerful heuristics, and the computational power of modern computers have enabled these approaches to be used on larger problems. Still, difficulties in the formulation of material flow constraints as mathematical inequalities and the development of generalized software solutions have limited the use of these approaches outside the classroom.

1.1 Decomposition strategies

Davis and Jones (9) proposed a methodology based on the decomposition of mathematical programming problems that used both Benders-type (10) and Dantzig/Wolfe-type (11) decompositions. The methodology was part of closed-loop, real-time, two-level hierarchical shop floor control system. The top-level scheduler (i.e., the supremal) specified the earliest start time and the latest finish time for each job. The lower level scheduling modules (i.e., the infimals) would refine these limit times for each job by detailed sequencing of all operations. A multicriteria objective function was specified that included tardiness, throughput, and process utilization costs. The decomposition was achieved by first reordering the constraints of the original problem to generate a block angular form, then transforming that block angular form into a hierarchical tree structure. In general, N subproblems would result plus a constraint set that contained partial members of each of the subproblems. The latter was termed the "coupling " constraints, and included precedence relations and material handling. The supremal unit explicitly considered the coupling constraints, while the infimal units considered their individual decoupled constraint sets. The authors pointed out that the inherent stochastic nature of job shops and the presence of multiple, but often conflicting, objectives made it difficult to express the coupling constraints using exact mathematical relationships. This made it almost impossible to develop a general solution methodology. To overcome this, a new real-time simulation methodology was proposed in (9) to solve the supremal and infimal problems.

Gershwin (12) used the notion of temporal decomposition to propose a mathematical programming framework for analysis of production planning and scheduling. This framework can be characterized as hierarchical and multi-layer. The problem formulations to control events at higher layers ignored the details of the variations of events occurring at lower layers. The problem formulations at the lower layers view the events at the higher layers as static, discrete events. Scheduling is actually carried out in bottom three layers so that the production requirements imposed by the planning layers can be met. First, a hedging point is found by solving a dynamic programming problem. This hedging point is the number of excess goods that should be produced to compensate for future equipment failures. This hedging point is used to formulate a linear programming problem to determine instantaneous production rates. These rates are then used to determine the actual schedule (which parts to make and when). A variety of approaches are under investigation to generate that schedule.

1.2 Enumerative techniques and Lagrangian relaxation

Two popular solution techniques for integer-programming problems are branch-and-bound and Lagrangian relaxation. Branch-and-bound is an enumerative technique (13) (14). Summarizing Morton and Pentico (15), "The basic idea of branching is to conceptualize the problem as a decision tree. Each decision choice point - a node - corresponds to a partial solution. From each node, there grow a number of new branches, one for each possible decision. This branching process continues until leaf nodes, that cannot branch any further, are reached. These leaf nodes are solutions to the scheduling problem". Although efficient bounding and pruning procedures have been developed to speed up the search, this is still a very computational intensive procedure for solving large scheduling problems. If the integer constraint is the main problem, then why not remove that constraint. A technique called Lagrangian relaxation, which has been used for more than 30 years, does just that (16). Lagrangian relaxation solves integer-programming problems by omitting specific integer-valued constraints and adding the corresponding costs (due to these omissions and/or relaxations) to the objective function. As with branch and bound, Lagrangian relaxation is computationally expensive for large scheduling problems.

1.3 Recent trends

Model-Based Optimization (MBO) is an optimization approach that uses mathematical expressions (e.g., constraints and inequalities) to model scheduling problems as mixed integer (non) linear programs (MINLP's) (17). A set of methods such as linear programming, branch-and-bound, and decomposition techniques are used to search the scenario space of solutions. Due to the advances in computer technologies, the computation times are becoming very practical. According to Subrahmanyam et al. (18) "For problems of moderate size, solutions of type D are given." Solutions of type D are optimal solutions of the maximum desirability possible within the constraints of operation. These approaches are being enhanced by the development of English-like "scheduling languages" and high-level graphical interfaces. The scheduling languages support the developing of the mathematical formulations with minimum intervention from the user.

2.0 Dispatching rules

Dispatching rules have been applied consistently to scheduling problems. They are procedures designed to provide good solutions to complex problems in real-time. The term dispatching rule, scheduling rule, sequencing rule, or heuristic are often used synonymously (19) (20) (21). Dispatching rules have been classified mainly according to the performance criteria for which they have been developed. Wu (22) categorized dispatching rules into several classes. Class 1 contains simple priority rules, which are based on information related to the jobs. Sub-classes are based on the particular piece of information used. Example classes include those based on processing times (such as SPT), due dates (such as EDD), slack (such as MINSLACK), and arrival times (such as FIFO). Class 2 consists of combinations of rules from class one. The particular rule that is implemented can now depend on the situation that exists on the shop floor. A typical example of a rule in this class is - use SPT until the queue length exceeds 5, then switch to FIFO. This prohibits jobs with large processing times from staying in the queue for long periods. Class 3 contains rules that are commonly referred to as Weight Priority Indexes. The idea is to use more than one piece of information about the jobs to determine the schedule. Pieces of information are assigned weights to reflect their relative importance. Usually, an objective function f(x) is defined. For example, f(x) = weight_1 * Processing Time of Job(x) + weight_2 * (Current Time - Due Date of Job(x)). Then, any time new sequence is needed, the function f(x) is evaluated for each job x in the queue. The jobs are ranked based on this evaluation.

During the last 30 years, the performance of a large number of these rules has been studied extensively using simulation techniques (23). These studies have been aimed at answering the question - If you want to optimize a particular performance criterion, which rule should you choose? Most of the early work concentrated on the shortest processing time rule (SPT). Conway and Maxwell (24) were the first to study the SPT rule and its variations. They found that, although some individual jobs could experience prohibitively long flow times, the SPT rule minimized the mean flow time for all jobs. They also showed that SPT was the best choice for optimizing the mean value of other basic measures such as waiting time and system utilization. Many similar investigations have been carried out to determine the dispatching rule which optimizes a wide range of job-related (such as due date and tardiness) and shop-related (such as throughput and utilization) performance measures. This problem of selecting the best dispatching rule for a given performance measure continues to be a very active area of research. However, the research has been expanded to include the possibility of switching rules to address an important problem: error recovery. Two early efforts to address error recovery were conducted by Bean and Birge (25) and Saleh (26). Both developed heuristic rules to smooth-out disruptions to the original schedule, thereby creating a match-up with that schedule. Bean and Birge based their heuristic on Turnpike Theory (27) to optimize a generalized cost function. Saleh showed that he could minimize duration of the disruption by switching the objective function from mean flow time to makespan. The technique he used to determine the actual schedule was based on disjunctive graphs (28).

3.0 Artificial intelligence (AI) techniques

Starting in the early 80s, a series of new technologies were applied to job shop scheduling problems. They fall under the general title of artificial intelligence (AI) techniques and include expert systems, knowledge-based systems, and several search techniques. Expert and knowledge-based systems were quite prevalent in the early and mid 80s. They have four main advantages. First, and perhaps most important, is that they use both quantitative and qualitative knowledge in the decision-making process. Second, they are capable of generating heuristics that are significantly more complex than the simple dispatching rules described above. The third is that the selection of the best heuristic can be based on information about the entire job shop including the current jobs, expected new jobs, and the current status of resources, material transporters, inventory, and personnel. Fourth, they capture complex relationships in elegant new data structures and contain special techniques for powerful manipulation of the information in these data structures There are, however, serious disadvantages. They can be time consuming to build and verify, as well as difficult to maintain and change. Moreover, since they generate only feasible solutions, it is rarely possible to tell how close that solution is to the optimal solution. Finally, since they are tied directly to the system they were built to manage, there is no such thing as a generic AI system.

3.1 Expert/knowledge-based systems

Expert and knowledge-based systems consist of two parts: a knowledge base, and inference engine to operate on that knowledge base. Formalizations of the “knowledge” that human experts use - into rules, procedures, heuristics, and other types of abstractions - are captured in the knowledge base. Three types of knowledge are usually included: procedural, declarative, and meta. Procedural knowledge is domain-specific problem solving knowledge. Declarative knowledge provides the input data defining the problem domain. Meta knowledge is knowledge about how to use the procedural and declarative knowledge to actually solve the problem. Several data structures have been utilized to represent the knowledge in the knowledge base including semantic nets, frames, scripts, predicate calculus, and production rules. The inference engine selects a strategy to apply to the knowledge bases to solve the problem at hand. It can be forward chaining (data driven) or backward chaining (goal driven).

ISIS (29) was the first major expert system aimed specifically at job shop scheduling problems. ISIS used a constraint-directed reasoning approach with three constraint categories: organizational goals, physical limitations and causal restrictions. Organizational goals considered objective functions based on due-date and work-in-progress. Physical limitations referred to situations where a resource had limited processing capability. Procedural constraints and resource requirements were typical examples of the third category. Several issues with respect to constraints were considered such as constraints in conflict, importance of a constraint, interactions of constraints, constraint generation and constraint obligation. ISIS used a three level, hierarchical, constraint-directed search. Orders were selected at level 1. A capacity analysis was performed at level 2 to determine the availability of the resources required by the order. Detailed scheduling was performed at level 3. ISIS also provided for the capability to interactively construct and alter schedules. In this capacity, ISIS utilized its constraint knowledge to maintain the consistency of the schedule and to identify scheduling decisions that would result in poorly satisfied constraints.

Wysk et al. (30) developed an integrated expert system/simulation scheduler called MPECS. The expert system used both forward and backward chaining to select a small set of potentially good rules from predefined set of dispatching rules and other heuristics in the knowledge base. These rules optimized a single performance measure, although that measure could change from one scheduling period to the next. The selected rules were then evaluated one at a time using a deterministic simulation of a laboratory manufacturing system. After all of the rules were evaluated, the best rule was implemented on the laboratory system. Data could be gathered about how the rule actually performed and used to update the knowledge base off-line. They were able to show that periodic rescheduling makes the system more responsive and adaptive to a changing environment. MPECS was important for several reasons. It was the first hybrid system to make decisions based on the actual feedback from the shop floor. It incorporated some learning into its knowledge base to improve future decisions. The same systems could be used to optimize several different performance measures. Finally, it utilized a new multi-step approach to shop floor scheduling.

Other examples of expert/knowledge-based scheduling systems developed include OPTIMUM-AIV (31), OPIS (Opportunistic Intelligent Scheduler) (32), and SONIA (33).

3.2 Distributed AI: agents

Due to the limited knowledge and the problem solving ability of a single expert or knowledge based system, these AI approaches have difficulty solving large scheduling problems as well. To address this, AI researchers have also begun to develop distributed scheduling system approaches (34). They have done this by an application of their well-known "divide and conquer" approach. This requires a problem decomposition technique, such as those described above, and the development of different expert/knowledge-based systems that can cooperate to solve the overall problem (35). The AI community's answer is the "agent" paradigm. An agent is a unique software process operating asynchronously with other agents. Agents are complete knowledge-based systems by themselves. The set of agents in a system may be heterogeneous with respect to long-term knowledge, solution-evaluation criteria, or goals, as well as languages, algorithms, hardware requirements. A multi-agent system is created by integrating agents selected from a "library" of agents.

For example, one such multi-agent system could involve two types of agents: tasks and resources. Each task agent might be responsible for scheduling a certain class of tasks such as material handling, machining, or inspection, on those resources capable of performing those tasks. This can be done using any performance measure related to tasks, such as minimize tardiness, and any solution technique. Each resource agent might be responsible for a single resource or a class of resources. Task agents must send their resource requests to the appropriate resource agent, along with the set of operations to be performed by that resource (36). Upon receipt of such a request, the resource agent must generate a new schedule using its own performance measures, such as maximize utilization, which includes this request. The resource agent will use the results to decide whether to accept this new request or not. To avoid the situation where no resource will accept a request, coordination mechanisms must be developed.

There are, now, no general guidelines for the design and implementation of this coordination. Therefore, the debates about centralized vs decentralized approaches to job shop scheduling go on. The agents formalism may provide an answer to these debates.

4.0 Artificial neural networks

Neural networks, also called connectionist or distributed parallel processing models, have been studied for many years in an attempt to mirror the learning and prediction abilities of human beings. Neural network models are distinguished by network topology, node characteristics, and training or learning rules. An example of a three-layer, feed-forward neural network is shown in Figure 1.

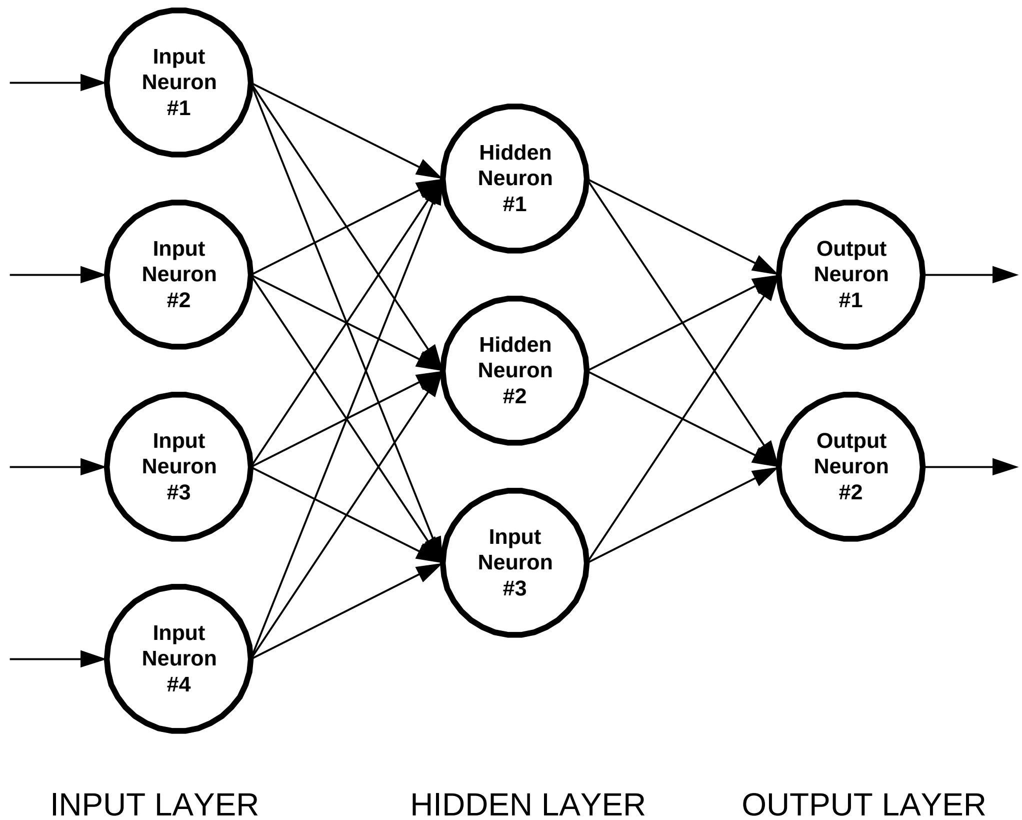

Figure 1. An example of a three-layer, feed-forward neural network

4.1 Supervised learning neural networks

Through exposure to "training patterns", supervised learning neural networks attempt to capture the desired relationships between inputs and the outputs. Back-propagation is the most popular and widely used supervised training procedure. Back-propagation (37) (38) applies the gradient-descent technique in the feed-forward network to change a collection of weights so that some cost function can be minimized. The cost function, which is only dependent on weights (**W**) and training patterns, is defined by:

$$\mathbf{C(W)} = \tfrac{1}{2}\Sigma\ (\mathbf{T}_{ij} - \mathbf{O}_{ij}) \qquad \textbf{(1)}$$

where the **T** is the target value, **O** is the output of the network, **i** represents the output nodes, and **j** represents the training patterns.

After the network propagates the input values to the output layer, the error between the desired output and actual output will be "back-propagated" to the previous layer. In the hidden layers, the error for each node is computed by the weighted sum of errors in the next layer's nodes. In a three-layered network, the next layer means the output layer. The activation function is usually a sigmoid function with the weights modified according to

$$\Delta \mathbf{W_{ij}} = \eta\ \mathbf{X_j\ (1- X_j)(T_j - X_j)\ X_i} \qquad \mathbf{(2)}$$

or

$$\Delta \mathbf{W_{ij}} = \eta\ \mathbf{X_j\ (1- X_j)\ (}\Sigma\ \delta_{\mathbf{k}}\mathbf{W_{jk}\ )\ X_i} \qquad \mathbf{(3)}$$

where $\mathbf{W_{jk}}$ is weight from node **i** to node (e.g., neuron) **j**, η is the learning rate, $\mathbf{X_j}$ is the output of node **j**, $\mathbf{T_j}$ is the target value of node **j**, and $\delta_{\mathbf{k}}$ is the error function of node **k**.

If **j** is in the output layer, Eq. (2) is used. If **j** is the hidden layers, Eq. (3) is used. The weights are updated to reduce the cost function at each step. The process continues until the error between the predicted and the actual outputs is smaller than some predetermined tolerance.

Rabelo (39) was the first to use back-propagation neural nets to solve job shop scheduling problems with several job types, exhibiting different arrival patterns, process plans, precedence sequences and batch sizes. Training examples were generated to train the neural network to select the correct characterization of the manufacturing environments suitable for various scheduling policies and the chosen performance criteria. In order to generate training samples, a performance simulation of the dispatching rules available for the manufacturing system was carried out. The neural networks were trained for problems involving 3, 4, 5, 8, 10, and 20 machines. To carry out this training, a special, input-feature space was developed. This space contained both job characteristics (such as types, number of jobs in each type, routings, due dates, and processing times) and shop characteristics (such as number of machines and their capacities). The output of the neural network represented the relative ranking of the available dispatching rules for that specific scheduling problem and the selected performance criteria. The neural networks were tested in numerous problems and their performance (in terms of minimizing Mean Tardiness) was always better than each single dispatching rule (25% to 50%).

4.2 Relaxation models

Neural networks based on relaxation models are defined by energy functions. They differ from "learning" schemes like backpropagation, which emphasize learning by mapping adjustments and utilize a transfer function to connect the input with the output. Relaxation models are pre-assembled systems that relax from input to output along a predefined energy contour. Hopfield neural networks (40) are a classical example of a relaxation model that has been used to solve some classic, textbook scheduling problems (41). Two-dimensional Hopfield networks were used to solve 4-job, 3-machine problems and 10-job, 10-machine problems (42). They were extended in (43) to 3 dimensions to represent jobs (i=1,...,I), machines j=1,...,J), and time (m=1,...,M). In each case, the objective was to minimize the makespan, total time to complete all jobs,which is defined as

$$\mathbf{E} = \sum_{\mathbf{j=1}} \sum_{\mathbf{i=1}} \sum_{\mathbf{m=1}} \mathbf{(v_{ijm})\ (m + T_{ij} - 1)} \qquad \mathbf{(4)}$$

where $\mathbf{v_{ijm}}$ is the output (1 or 0) of neuron **ijm**, and $\mathbf{T_{ij}}$ is the time required by $\mathbf{j^{th}}$ resource (e.g., machine) to complete the $\mathbf{i^{th}}$ job.

Due to a large number of variables involved in generating a feasible schedule, these approaches tend to be computationally inefficient and frequently generate infeasible solutions. Consequently, they have not been used to solve realistic scheduling problems.

4.3 Unsupervised neural networks (competition-based)

Competition-based neural networks are good at classifying or clustering input data. They can also be applied to scheduling problems. Figure 2 shows a two layer competitive network with two input and four output units. These unsupervised neural networks proceed on their own to find a useful correlation in the input data. A set of n I-dimensional data is presented to the network, which adaptively adjusts its weights. This input presentation process is repeated until the network reaches stability, that is each output unit gets activated only for a particular subset of the input patterns. Since the classes or clusters are not known in advance, the network must discover them. Variations of these neural networks have been used to solve scheduling problems. As an example, Bourret et al. (44) applied some of these principles to develop a neural network that was able to optimally schedule time periods of low-level satellites to one or several antennas. The neural network was able to take into account that each satellite has a given priority and several other operational constraints.

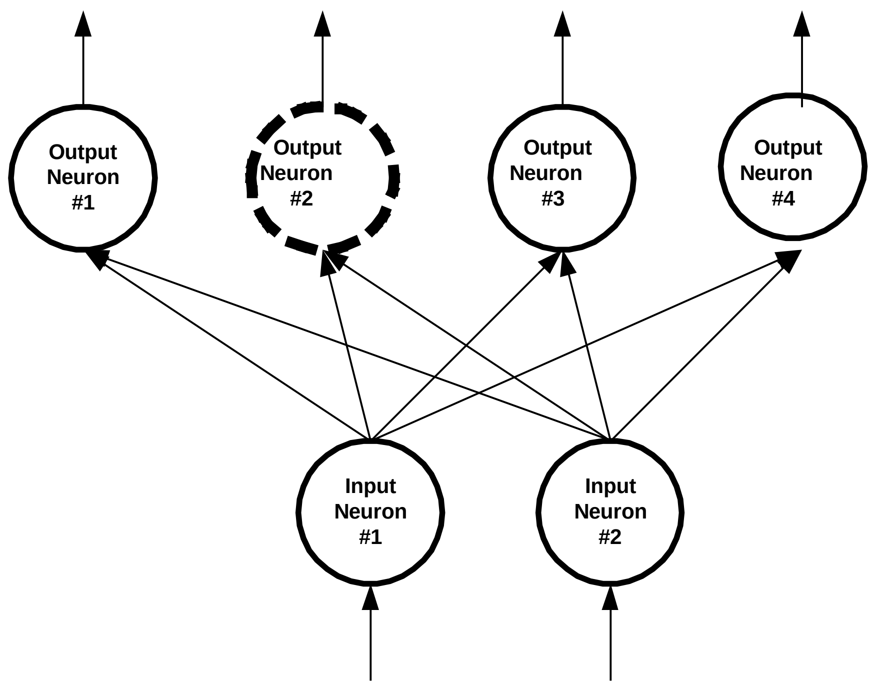

Figure 2. An example of an unsupervised neural network (based on competition) with two input neurons connected to four output neurons. The output neuron that responds most strongly to the input units values (in this case output neuron #2) is then trained to reinforce these connections.

4.4 Temporal reinforcement learning

It was said above that supervised learning neural networks attempt to capture the desired relationships between inputs and the outputs through exposure to training patterns. However, for some problems, the desired response may not always be available during the time of learning. When, the desired response is obtained, changes to the neural network are performed by assessing penalties for the actions previously decided by the neural network. As summarized by Tesauro (45), "In the simplest form of this paradigm, the learning system passively observes a temporal sequence of input states that eventually leads to a final reinforcement or reward signal (usually a scalar). The learning system's task in this case is to predict expected reward given an observation of an input state or sequence of input states. The system may also be set up so that it can generate control signals that influence the sequence of states." For scheduling, the learning task is to produce an scheduling action that will lead to minimize (or maximize) the performance measure (e.g., makespan, tardiness). Several procedures have been developed to train neural networks in a variety of generic cases. Rabelo et al. (46) utilized a procedure developed by Watkins (47), denominated Q-learning, to implement a scheduling system to solve dynamic scheduling problems. The scheduling system developed was able to follow trends in the shop floor and select a dispatching rule that provided the maximum reward according to performance measures based on tardiness and flow time. Zhang and Dietterich (48) utilized a procedure developed by Sutton (49) called TD(λ) to schedule payload processing of NASA's space shuttle program. The scheduling system implemented was able to outperform an iterative repair scheduler developed that combined heuristics with simulated annealing.

5. 0 Neighborhood search methods

Neighborhood search methods are very popular. Neighborhood search methods provide good solutions and offer possibilities to be enhanced when combined with other heuristics. One of the first neighborhood procedures was developed by Wilkerson and Irwin (50). This method iteratively added small changes ("perturbations") to an initial schedule, which is obtained by any heuristic. Conceptually similar to hill climbing, these techniques continue to perturb and evaluate schedules until there is no improvement in the objective function. When this happens, the procedure is ended. Popular techniques that belong to this family include Tabu search, simulated annealing, and genetic algorithms. Each of these has its own perturbation methods, stopping rules, and methods for avoiding local optimum.

5.1 Tabu search

The basic idea of Tabu search (51) (52) is to explore the search space of all feasible scheduling solutions by a sequence of moves. A move from one schedule to another schedule is made by evaluating all candidates and choosing the best available, just like gradient-based techniques. Some moves are classified as tabu (i.e., they are forbidden) because they either trap the search at a local optimum, or they lead to cycling (repeating part of the search). These moves are put onto something called the Tabu List, which is built up from the history of moves used during the search. These tabu moves force exploration of the search space until the old solution area (e.g., local optimum) is left behind. Another key element is that of freeing the search by a short term memory function that provides "strategic forgetting". Tabu search methods have been evolving to more advanced frameworks that includes longer term memory mechanisms. These advanced frameworks are sometimes referred as Adaptive Memory Programming (AMP) (53).

Tabu search methods have been applied successfully to scheduling problems and as solvers of mixed integer programming problems. Nowicki and Smutnicki (53) implemented tabu search methods for job shop and flow shop scheduling problems. Vaessens et al. (53) show that tabu search methods (in specific scheduling cases) are superior over other approaches such as simulated annealing, genetic algorithms, and neural networks. Skorin and Labourdette (53) have applied tabu search methods to schedule and manage optical telecommunications networks. Porto and Ribeiro (54) develop a scheduling system based on tabu search to schedule the loads in heterogeneous multiprocessor environments.

5.2 Simulated annealing

Simulated annealing is based on the analogy to the physical process of cooling and recrystalization of metals. The current state of the thermodynamic system is analogous to the current scheduling solution, the energy equation for the thermodynamic system is analogous to the objective function, and the ground state is analogous to the global optimum. In addition to the global energy J, there is a global temperature T, which is lowered as the iterations progress. Using this analogy, the technique randomly generates new schedules by sampling the probability distribution of the system (55):

$$\mathbf{p}_j \propto \exp(-T(\Delta J_{best} - \Delta J_j)/K) \qquad \textbf{(5)}$$

where $\mathbf{P_j}$ represents the probability of making move **j** from among the neighborhood choices. $\Delta\mathbf{J}_{\mathbf{best}}$ represents the improvement of the objective function for the best choice, and $\Delta\mathbf{J_j}$ represents the improvement for choice **j**. **K** is a normalization factor. Since increases of energy can be accepted, the algorithm is able to escape local minima.

Simulated annealing has been applied effectively to scheduling problems. Vakharia and Chang (56) developed a scheduling system based on simulated annealing for manufacturing cells. Jeffcoat and Bulfin (57) applied simulated annealing to a resource-constrained scheduling problem. Their computational results indicated that the simulated annealing procedure provided the best results in comparison with other neighborhood search procedures.

5.3 Genetic algorithms

Genetic algorithms (GA) are an optimization methodology based on a direct analogy to Darwinian natural selection and mutations in biological reproduction. In principle, genetic algorithms encode a parallel search through concept space, with each process attempting coarse-grain hill climbing (59). Instances of a concept correspond to individuals of a species. Induced changes and recombinations of these concepts are tested against an evaluation function to see which ones will survive to the next generation. The use of genetic algorithms requires five components:

1. A way of encoding solutions to the problem **-** fixed length string of symbols.
2. An evaluation function that returns a rating for each solution.
3. A way of initializing the population of solutions.
4. Operators that may be applied to parents when they reproduce to alter their genetic composition such as crossover (i.e., exchanging a randomly selected segment between parents), mutation (i.e., gene modification), and other domain specific operators.
5. Parameter setting for the algorithm, the operators, and so forth.

A number of approaches have been utilized in the application of genetic algorithms (GA) to scheduling problems (60) (61) (62):

1. Genetic algorithms with blind recombination operators have been utilized in job shop scheduling. Their emphasis on relative ordering schema, absolute ordering schema, cycles, and edges in the offsprings will arise differences in such blind recombination operators.
2. Sequencing problems have been also addressed by the mapping of their constraints to a Boolean satisfiability problem using partial payoff schemes. This scheme has produced good results for very simple problems.
3. Heuristic genetic algorithms have been applied to job shop scheduling. In these genetic schemes, problem specific heuristics are incorporated in the recombination operators (such as optimization operators based).

Table 1. The sequencing problem

Queue Position	*Job Type*	*Arrival Time*	*Processing Time*	*Due Date*
1	6	789	Normal(8,0.4)	890
2	6	805	Normal(8,0.4)	911
3	5	809	Normal(10,0.6)	910
4	1	826	Normal(4,0.2)	886
5	2	830	Normal(6,0.3)	905
6	7	832	Normal(15,0.75)	1009
7	6	847	Normal(8,0.4)	956
8	3	848	Normal(5,0.2)	919
9	1	855	Normal(4,0.2)	919
10	4	860	Normal(3,0.1)	920

Table 2. Set-Up Times Dependencies

		Previous Job Type						
		1	***2***	***3***	***4***	***5***	***6***	***7***
Current	***1***	0	1	2	2	3	2	2
Job	***2***	1	0	2	3	4	3	2
Type	***3***	2	2	0	3	4	3	2
	4	1	2	2	0	4	4	2
	5	1	2	2	3	0	3	2
	6	1	2	2	3	3	0	3
	7	1	2	2	2	2	2	0

The following example illustrates the application of both the blind recombination and partially mapped crossover (PMX) operators (61) to a simple sequencing problem with 7 job types. Each job-type has its own arrival time, due date, and processing time distributions. The objective is to determine a sequence that minimizes Maximum Tardiness for the 10-job problem described in Table 1. The set-up time for these jobs is sequence dependent as shown in Table 2.

The simple genetic algorithm procedure is used to resequence the 10 jobs in the queue.

1. Randomly generate n feasible sequences e.g., n=50.
2. Compute the tardiness for each sequence and rank from lowest to highest.
3. Choose the m best sequences, m < n, e.g., m=25.
4. Use blind recombination operator to produce a new set of n feasible sequences.
5. Randomly select pairs of sequences. Apply PMX operator.
6. Randomly select a pair of jobs in each offspring and mutate (switch) them.

This process continues until no improvement in the objective function can be found.

The PMX operator generates offspring by randomly selecting a swapping interval between two crossover points and switching the jobs. These offspring will inherit the elements from the interval of one of the parents. If the resulting sequence is infeasible (has one or more duplicate jobs) the infeasibility must be removed through a technique known as mapping and exchanging. For example, consider two sequences (A and B):

Position	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
A (Job Numbers)	**9**	**8**	**4**	**5**	**6**	**7**	**1**	**3**	**2**	**10**
B (Job Numbers)	**8**	**7**	**1**	**2**	**3**	**10**	**9**	**5**	**4**	**6**

If the swapping interval 4 to 6 was selected, then the application of PMX would yield two infeasible sequences.

Position	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
A'(Job Numbers)	**9**	**8**	**4**	**2**	**3**	**10**	**1**	**3**	**2**	**10**
B'(Job Numbers)	**8**	**7**	**1**	**5**	**6**	**7**	**9**	**5**	**4**	**6**

Now, we apply the mapping and exchanging technique to remove duplicate jobs. This yields the following two feasible offsprings.

Position	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
A''(Job Numbers)	**9**	**8**	**4**	**2**	**3**	**10**	**1**	**6**	**5**	**7**
B''(Job Numbers)	**8**	**10**	**1**	**5**	**6**	**7**	**9**	**2**	**4**	**3**

These offsprings do not necessarily replace their parents in the population; they are simply placed according to their ranking. After 17 iterations, approximately: 17 * 50 = 850 sequences were generated and tested (this took less than 100 milliseconds in a PC 486 @ 33MHz). The optimal sequence was found to be

8	**5**	**4**	**1**	**2**	**3**	**9**	**10**	**6**	**7**

with a Maximum Tardiness of 2.

Starkweather et al. (63) were the first to use genetic algorithms to solve a dual -criteria job shop scheduling problem in a real production facility. Those criteria were the minimization of average inventory in the plant and the minimization of the average waiting time for an order to be selected. These criteria are negatively correlated (The larger the inventory, the shorter the wait; the smaller the inventory, the longer the wait.). To represent the production/shipping optimization problem, a symbolic coding was used for each member (chromosome) of the population. In this scheme, customer orders are represented by discrete integers. Therefore, each member of the population is a permutation of customer orders. The Genetic Algorithm used to solve this problem was based on a modification to the blind recombinant operator described above. This recombination operator emphasizes information about the relative order of the elements in the permutation, because this impacts both inventory and waiting time. A single evaluation function (a weighted sum of the two criteria) was utilized to rank each member of the population. That ranking was based on an on-line simulation of the plant operations. This approach generated schedules that produced inventory levels and waiting times that were acceptable to the plant manager. In addition, the integration of the genetic algorithm with the on-line simulation made it possible to react to system dynamics.

These applications have emphasized the utilization of genetic algorithms as a "solo" technique. This has limited the level of complexity of the problems solved and their success. Recent research publications have demonstrated the sensitivity of genetic algorithms to the initial population. When the initial population is generated randomly, genetic algorithms are shown to be less efficient that the annealing-type algorithms, but better than the heuristic methods alone. However, if the initial population is generated by a heuristic, the genetic algorithms become as good as, or better than the annealing-type algorithms. In addition, integration with other search procedures (e.g., tabu search) has enhanced the capabilities of both. This result is not surprising, as it is consistent with results from non-linear optimization. Simply stated, if you begin the search close to the optimal solution you are much more likely to get the optimum than if you begin the search far away.

6.0 Fuzzy logic

Fuzzy set theory has been utilized to develop hybrid scheduling approaches. Fuzzy set theory can be useful in modeling and solving job shop scheduling problems with uncertain processing times, constraints, and set up times. These uncertainties can be represented by fuzzy numbers that are described by using the concept of an interval of confidence. These approaches usually are integrated with other methodologies (e.g., search procedures, constraint relaxation). For example, Slany (64) stresses the imprecision of straight-forward methods presented in the mathematical approaches and introduces a method known as fuzzy constraint relaxation, which is integrated with a knowledge-based scheduling system. His system was applied to a steel manufacturing plant. Grabot and Geneste (65) use fuzzy logic principles to combine dispatching rules for multi-criteria problems. On the other hand, Krucky (66) addresses the problem of minimizing setup times of a medium-to-high product mix production line using fuzzy logic. The heuristic, fuzzy logic based algorithm described helps determine how to minimize setup time by clustering assemblies into families of products that share the same setup by balancing a product's placement time between multiple-high-speed placement process steps. Tsujimura et al. (67) presented a hybrid system, which uses fuzzy set theory to model the processing times of a flow shop scheduling facility. Triangular Fuzzy Numbers (TFNs) are used to represent these processing times. Each job is defined by two TFNs, a lower bound and an upper bound. A branch and bound procedure is utilized to minimize makespan.

7.0 Reactive Scheduling

Reactive scheduling is generally defined as the ability to revise or repair a complete schedule that has been "overtaken" by events on the shop floor (68) (69). Such events include rush orders, excessive delays, and broken resources. There are two approaches: reactive repair and the proactive adjustment. In the former, the scheduling system waits until an event has occurred before it attempts to recover from that event. The match-up techniques described in section 3 fall into this category. The other approach requires a capability to monitor the system continuously, predict the future evolution of the system, do contingency planning for likely events, and generate new schedules, all during the execution time of the current schedule. The work of Wysk, (30) and Davis (9) fall into this category. Approaches that are more recent utilize artificial intelligence and knowledge-based methodologies (32). Still most of the AI approaches propose a quasi-deterministic view of the system, i.e., a stochastic system featuring implicit and/or explicit causal rules. The problem formulation used does not recognize the physical environment of the shop floor domain where interference not only leads to readjustment of schedules but also imposes physical actions to minimize them.

8.0 Learning in Scheduling

The first step in developing a knowledge base is knowledge acquisition. This in itself is a two step process: get the knowledge from knowledge sources and store that knowledge in digital form. Much work has been done in the area of knowledge acquisition, such as protocol analysis, interactive editing, and so on (70). Knowledge sources may be human experts, simulation data, experimental data, databases, text, etc. In scheduling problems, the knowledge sources are likely to be human experts or simulation data. To extract knowledge from these two sources, the machine learning technique that learns from examples (data) becomes a promising tool. Inductive learning is a state classification process. If we view the state space as a hyperplane, the training data (consisting of conditions and decisions) can be represented as points on the hyperplane. The inductive learning algorithm seeks to draw lines on the hyperplane based on the training data to divide the plane into several areas within which the same decision (conclusion) will be made.

One algorithm that has been implemented in inductive aids and expert system shells is that developed by Ross Quinlan, called Iterative Dichotomister 3 or ID3 (71). ID3 uses examples to induce production rules (e.g. IF ... THEN ...), which form a simple decision tree. Decision trees are one way to represent knowledge for the purpose of classification. The nodes in a decision tree correspond to attributes of the objects to be classified, and the arcs are alternative values for these attributes. The end nodes of the tree (leaves) indicate classes to which groups of objects belong. Each example is described by attributes and a resulting decision. To determine a good attribute to partition the objects into classes, entropy is employed to measure the information content of each attribute, and then rules are derived through a repetitive decomposition process that minimizes the overall entropy. The entropy value of attribute A_k can be defined as

$$\mathbf{H(A_k) = \sum_{j=1}^{M_k} P(a_{kj}) \{- \sum_{i=1}^{N} P(c_i|a_{kj}) \log 2\, P(c_i|a_{kj})\}} \qquad \mathbf{(6)}$$

where $\mathbf{H(A_k)}$ is the entropy value of attribute $\mathbf{A_k}$, $\mathbf{P(a_{kj})}$ is the probability of attribute k being at its j^{th} value, $\mathbf{P(c_i|a_{kj})}$ is the probability that the class value is $\mathbf{c_i}$ when attribute **k** is at its $\mathbf{j^{th}}$ value, $\mathbf{M_k}$ is the total number of values for attribute $\mathbf{A_k}$, and **N** is the total number of different classes (outcomes).

The attribute with the minimum entropy value will be selected as a node in the decision tree to partition the objects. The arcs out of this node represent different values of this attribute. If all the objects in an arc belong to one class, the partition process stops. Otherwise, another attribute will be identified using entropy values to further partition the objects that belong to this arc. This partition process continues until all the objects in an arc are in the same class. Before applying this algorithm, all attributes that have continuous values need to be transformed to discrete values.

In the context of job shop scheduling, the attributes represent system status and the classes represent the dispatching rules. Very often, the attribute values are continuous. In 1988, (72) proposed a trace-driven knowledge acquisition (TDKA) methodology to deal with continuous data and to avoid the problems occurring in verbally interviewing human experts. TDKA learns scheduling knowledge from expert schedulers without a dialogue with them. There are three steps in this approach. In Step 1, an interactive simulator is developed to mimic the system of interest. The expert will interact with this simulator and make decisions. The entire decision making process will be recorded in the simulator and can be repeated for later analysis. The series of system information and the corresponding decision collected is called a "trace." Step 2 analyzes the "trace" and forms classification rules to partition the trace into groups. The partition process stops when most of the cases in each group use the same dispatching rule (error rate is below the threshold defined by the knowledge engineer). Then, the decision rules are formed. The last step is to verify the generated rules. The resulting rule base is used to schedule jobs in the simulator. If it performs as well as or better than the expert, the process stops. Otherwise, the threshold value is increased, and the process returns to Step 2.

As the job shop operates over time, it is important to be able to modify the knowledge contained in these rule bases. Chiu's work (73) is specifically designed to look at knowledge modification for job shop scheduling problems. He developed a framework of dynamic scheduling schemes that explores routing flexibility and handles uncertainties. He proposed a learning-based methodology to extract scheduling knowledge for dispatching jobs to machines. The proposed methodology includes three modules: discrete-event simulation, instance generation, and incremental induction. First, a simulation module is developed to implement the dynamic scheduling scheme, to generate training examples, and to evaluate the methodology. Second, in an instance-generation module, the searching of good training examples is successfully fulfilled by the genetic algorithm. Finally, in an incremental-induction module, a tolerance-based incremental learning algorithm is proposed to allow continuous learning and facilitate knowledge modification. This algorithm uses entropy values to select attributes to partition the examples where the attribute values are continuous. The tolerance is used to maintain the stability of the existing knowledge while the new example is introduced. The decision tree will not be reconstructed unless there is enough momentum from the new data, that is, the change of the entropy value become significant. The experimental results showed that the tolerance-based incremental learning algorithm cannot only reduce the frequency of modifications, but also enhances the generalization ability of the resulting decision tree in a distributed job shop environment.

9.0 Summary and conclusions

Since scheduling problems fall into the class of NP-complete problems, they are among the most difficult to formulate and solve. Operations research analysts and engineers have been pursuing solutions to these problems for more than 30 years, with varying degrees of success. Through an evolutionary approach and a number of key examples, this paper has summarized the major strategies used to solving scheduling problems. These strategies have been based on a number of technologies ranging from mathematical programming to genetic algorithms. In addition, new trends were described.

While they are difficult to solve, scheduling problems are among the most important because they impact the ability of manufacturers to meet customer demands and make a profit. They also impact the ability of autonomous systems to optimize their operations, the deployment of intelligent systems, and the optimizations of communications systems. For this reason, operations research analysts and engineers will continue this pursuit well into the next century.

Bibliography

1. M. Dempster, J. Lenstra, and R. Kan, Deterministic and stochastic scheduling: introduction. *Proceedings of the NATO Advanced Study and Research Institute on Theoretical Approaches to Scheduling Problems*, D. Reidel Publishing Company: 3-14, 1981.

2. S. French, *Sequencing and Scheduling*. New York: Halsted Press, 1982.

3. S. Graves, A Review of Production Scheduling. *Operations Research*, **29**: 646-675, 1981.

4. E. Balas, An additive algorithm for solving linear programs with zero-one variables. *Operations Research*, **13**: 517-546, 1965.

5. E. Balas, Discrete programming by the filter method. *Operations Research*, **15**: 915-957, 1967.

6. E. Balas, Machine sequencing via disjunctive graphs: An implicit enumeration algorithm. *Operations Research*, **17**: 1-10, 1969.

7. E. Balas, Machine sequencing: disjunctive graphs and degree-constrained subgraphs. *Naval Research Logistics Quarterly*, **17**: 941-957, 1970.

8. V. Srinivasan, A hybrid algorithm for the one machine sequencing problem to minimize total tardiness. *Naval Research Logistics Quarterly*, **18**: 317-327, 1971.

9. W. Davis and A. Jones, A real-time production scheduler for a stochastic manufacturing environment. *International Journal of Computer Integrated Manufacturing*, **1** (2): 101-112, 1988.

10. J. Benders, Partitioning procedures for solving mixed-variables mathematical programming problems. *Numersche Mathematik*, **4** (3): 238-252, 1960.

11. G. Dantzig and P. Wolfe, Decomposition principles for linear programs. *Naval Research Logistics Quarterly,* **8** (1): 101-111, 1960.

12. S. Gershwin, Hierarchical flow control: a framework for scheduling and planning discrete events in manufacturing systems. *Proceedings of IEEE Special Issue on Discrete Event Systems*, **77**: 195-209, 1989.

13. N. Agin, Optimum seeking with branch and bound. *Management Science*, **13**, 176-185, 1966.

14. E. Lawler and D. Wood, Branch and bound methods: a survey. *Operations Research*, **14**, 699-719, 1966.

15. E. Morton and D. Pentico, *Heuristic Scheduling Systems*, New York: John Wiley & Sons, 1993.

16. J. Shapiro, A survey of Lagrangian techniques for discrete optimization. *Annals of Discrete Mathematics*, **5**: 113-138, 1979.

17. M. Zentner, J. Pekny, G. Reklaitis, and N. Gupta, Practical considerations in using model-based optimization for the scheduling and planning of batch/semicontinuous processes. *J. Proc. Cont.*, **4** (4): 259-280, 1994.

18. S. Subrahmanyam, M. Zentner, and J. Pekny, Making the most out of corporate information assets: the next generation of process scheduling, planning, and design tool. *Proceedings of the Process Industry Technical Conference: Looking Toward the 21st Century*, June 26-27, Erie, Pennsylvania, 1996.

19. S. Panwalker and W. Iskander, A survey of scheduling rules. *Operations Research*, **25** (1): 45-61, 1977.

20. J. Blackstone, D. Phillips, and G. Hogg, A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research*, **20** (1): 27-45, 1982.

21. K. Baker, *Introduction to Sequencing and Scheduling*. New York: John Wiley & Sons, 1974.

22. D. Wu, *An Expert Systems Approach for the Control and Scheduling of Flexible Manufacturing Systems*. Ph.D. Dissertation, Pennsylvania State University, 1987.

23. M. Montazer and L. Van Wassenhove, Analysis of scheduling rules for an FMS. *International Journal of Production Research*, **28**: 785-802, 1990.

24. R. Conway and W. Maxwell, *Theory of Scheduling*. Reading, Massachusetts: Addison-Wesley, 1967.

25. J. Bean and J. Birge, Match-up real-time scheduling. NBS Special Publication, **724**: 197-212, 1986.

26. A. Saleh, *Real-Time Control of a Flexible Manufacturing Cell*. Ph.D. Dissertation, Lehigh University, 1988.

27. L. McKenzie, Turnpike theory. *Econometrics*, **44**: 841-864, 1976.

28. J. Adams, E. Balas and D. Zawack, The shifting bottleneck procedure for job shop scheduling. *Management Science*: **34** (3), 391-401, 1988.

29. M. Fox, *Constraint-Directed Search: A case study of Job Shop Scheduling*. Ph.D. Dissertation, Carnegie-Mellon University, 1983.
30. R. Wysk, D. Wu and R. Yang, A multi-pass expert control system (MPECS) for flexible manufacturing systems", NBS Special Publication, **724**: 251-278, 1986.
31. M. Aarup, M. Arentoft, Y. Parrod, I. Stokes, H. Vadon and J. Stader, OPTIMUM-AIV: A Knowledge-based planning system for spacecraft AIV. In *Intelligent Scheduling*, M. Zweben and M. Fox (eds.), San Francisco: Morgan Kaufman, 451-469, 1995.
32. S. Smith, OPIS: A methodology and architecture for reactive scheduling. In *Intelligent Scheduling*, M. Zweben and M. Fox (eds.), San Francisco: Morgan Kaufman, 29-66, 1995.
33. C. Le Pape, Scheduling as intelligent control of decision-making and constraint propagation . In *Intelligent Scheduling*, M. Zweben and M. Fox (eds.), San Francisco: Morgan Kaufman, 67-98, 1995.
34. H. Parunak, B. Irish, J. Kindrick, and P. Lozo, Fractal actors for distributed manufacturing control. *Proceedings of the Second IEEE Conference on Artificial Intelligence Applications*, 653-660, 1985.
35. M. Zhang and C. Zhang, The consensus of uncertainties in distributed expert systems. *Proceedings of the First International Conference on Multi-Agent Systems*, Cambridge, Massachusetts: MIT Press, 1995.
36. T. Daouas, K. Ghedira and J. Muller, Distributed flow shop scheduling problem versus local optimization. *Proceedings of the First International Conference on Multi-Agent Systems*, Cambridge, Massachusetts: MIT Press, 1995.
37. D. Rumelhart, J. McClelland and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, ***1**: Foundations*. Cambridge, Massachusetts: MIT Press, 1986.
38. P. Werbos, Neurocontrol and supervised learning: An overview and evaluation. *In Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, D.A. White and D.A. Sofge (eds.), New York: Van Nostrand Reinhold Publication, 65-89, 1995.
39. L. Rabelo, *A Hybrid Artificial Neural Networks and Knowledge-Based Expert Systems Approach to Flexible Manufacturing System Scheduling*. PhD. Dissertation, University of Missouri-Rolla, 1990.
40. J. Hopfield and D. Tank, Neural computation of decisions in optimization problems. *Biological Cybernetics*, **52**: 141-152, 1985.
41. Y. Foo and Y. Takefuji, Stochastic neural networks for solving job-shop scheduling: Part 2. Architecture and simulations. *Proceedings of the IEEE International Conference on Neural Networks*, published by IEEE TAB: II283-II290, 1988.
42. D. Zhou, V. Cherkassky, T. Baldwin and D. Hong, Scaling neural networks for job shop scheduling. *Proceedings of the International Conference on Neural Networks*, **3**: 889-894, 1990.
43. Z. Lo and B. Bavarian, Scheduling with neural networks for flexible manufacturing systems. *Proceedings of the IEEE International Conference on Robotics and Automation*, Sacramento, California, 818-823, 1991.
44. P. Bourret, S. Goodall, and M. Samuelides, Optimal scheduling by competitive activation: application to the satellite-antennae scheduling problem. *Proceedings of the International Joint Conference on Neural Networks*, 1989.
45. G. Tesauro, Practical issues in temporal difference learning. *Machine Learning*, **8**: 257-277, 1992.
46. L. Rabelo, M. Sahinoglu and X. Avula, Flexible manufacturing systems scheduling using Q-Learning. *Proceedings of the World Congress on Neural Networks*, San Diego, California: I378-I385, 1994.
47. C. Watkins, *Learning from Delayed Rewards*. PhD. Thesis, King's College, Cambridge, 1989.
48. W. Zhang and T. Dietterich, High-performance job-shop scheduling with a time-delay TD(λ) network. In *Advances in Neural Informantion Processing Systems*, **8**, D. Touretzky, M. Mozer, and M. Hasselmo (eds.), Cambridge, Massachusetts: MIT Press: 1025-1030, 1996.
49. R. Sutton, Learning to predict by the methods of temporal differences. *Machine Learning*, **3**: 9-44, 1988.
50. L. Wilkerson and J. Irwin, An improved algorithm for scheduling independent tasks. *AIIE Transactions*, **3**: 239-245, 1971.
51. F. Glover, Tabu search - Part I. *ORSA Journal on Computing*, **1** (3): 190-206, 1989.
52. F. Glover, Tabu search - Part II. *ORSA Journal on Computing*, **2** (1): 4-32, 1990.
53. F. Glover, Tabu search and adaptive memory programming - advances, applications and challenges. To appear in*: Interfaces in Computer Science and Operations Research*. B. Helgason and K. Kennington (eds.), The Netherlands: Kluwer Academic Publishers, 1996.

54. S. Porto and C. Ribeiro. A Tabu Search Approach to Task Scheduling on Heterogeneous Processors Under Precedence Constraints. *International Journal of High-Speed Computing*, **7**: 45-71, 1995.

55. S. Kirkpatrick, C.D. Gelatt Jr, and M.P. Vecchi, Optimization by simulated annealing. *Science*, **220** (4598): 671-680, 1983.

56. A. Vakharia and Y. Chang, A simulated annealing approach to scheduling a manufacturing cell. *Naval Research Logistics*, **37**: 559-577, 1990.

57. D. Jeffcoat and R. Bulfin, Simulated annealing for resource-constrained scheduling. *European Journal of Operational Research*, **70:** 43-51, 1993.

59. D. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*. Menlo Park: California: Addison-Wesley, 1988.

60. L. Davis, Job shop scheduling with genetic algorithms. *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, Carnegie Mellon University, 136-140, 1985.

61. D. Goldberg and R. Lingle R., 1985, Alleles, loci, and the traveling salesman problem. *Proceedings of the of the International Conference on Genetic Algorithms and Their Applications*, Carnegie Mellon University, 162-164, 1985.

62. T. Starkweather, D. Whitley, K. Mathias and S. McDaniel, Sequence scheduling with genetic algorithms. *Proceedings of the US/German Conference on New Directions for OR in Manufacturing*, 130-148, 1992.

63. T. Starkweather, D. Whitley and B. Cookson, A Genetic Algorithm for scheduling with resource consumption., *Proceedings of the Joint German/US Conference on Operations Research in Production Planning and Control*, 567-583, 1993.

64. W. Slany, *Scheduling as a fuzzy multiple criteria optimization problem*. CD-Technical Report 94/62, Technical University of Vienna, 1994.

65. B. Grabot and L. Geneste, Dispatching rules in scheduling: a fuzzy approach. *International Journal of Production Research*, **32** (4): 903-915, 1994.

66. J. Krucky, Fuzzy family setup assignment and machine balancing. *Hewlett-Packard Journal*, June: 51-64, 1994.

67. Y. Tsujimura, S. Park, S. Chang, and M. Gen, An effective method for solving flow shop scheduling problems with fuzzy processing times. *Computers and Industrial Engineering*, **25**: 239-242, 1993.

68. M. Zweben, B. Daun, E. Davis, and M. Deale, Scheduling and rescheduling with iterative repair. In *Intelligent Scheduling*, M. Zweben and M. Fox (eds.), San Francisco: Morgan Kaufman, 241-256, 1995.

69. K. Kempf, Intelligently scheduling semiconductor wafer fabrication. In *Intelligent Scheduling*, M. Zweben and M. Fox (eds.), San Francisco: Morgan Kaufman, 517-544, 1995.

70. M. Shaw, S. Park, and N. Raman, Intelligent scheduling with machine learning capabilities: The induction of scheduling knowledge. *IEE Transactions on Design and Manufacturing*, **24**, 156-168, 1992.

71. J. Quinlan, Induction of decision trees. *Machine Learning*, **1**: 81-106, 1986.

72. Y. Yih, Trace-driven knowledge acquisition (TDKA) for rule-based real-time scheduling systems. *Journal of Intelligent Manufacturing*, **1** (4): 217-230, 1990.

73. C. Chiu, *A Learning-Based Methodology for Dynamic Scheduling in Distributed Manufacturing Systems*. Ph.D. Dissertation, Purdue University, 1994.