CHAPTER 2
BACKGROUND

This chapter of the dissertation performs two functions: it elaborates on the topics in the introduction and provides a review of the relevant literature. The discussion in this chapter is both wide-ranging and descriptive. In Chapters 3, 4, and 5, discussions of previous research will be problem-specific and less explicit.

The first section will describe in more detail the particular operations associated with the testing of semiconductor devices. Remaining sections will discuss papers on semiconductor manufacturing, job shop scheduling techniques, and flow shop scheduling problems. Also included in this chapter are sections on look-ahead and look-behind scheduling, the scheduling of the burn-in operation, class scheduling, some one-machine problems, smart-and-lucky searches, problem and heuristic space, and NP-completeness.

2.1 Semiconductor Test Operations

The manufacturing of semiconductors consists of many complex steps. As described in the introduction, this includes wafer fabrication, probe, assembly, and test. This research is concerned with this last facility. Although the routes of lots through the test process vary significantly over the many different types of products, general trends can be described. Figure 2.1 shows what these routes look like for seventeen representative products of a test area. The numbers on each arc are the number of products that follow that path.

Presented here is a description of the operations in a typical product route. The terms semiconductor and device describe the same thing.

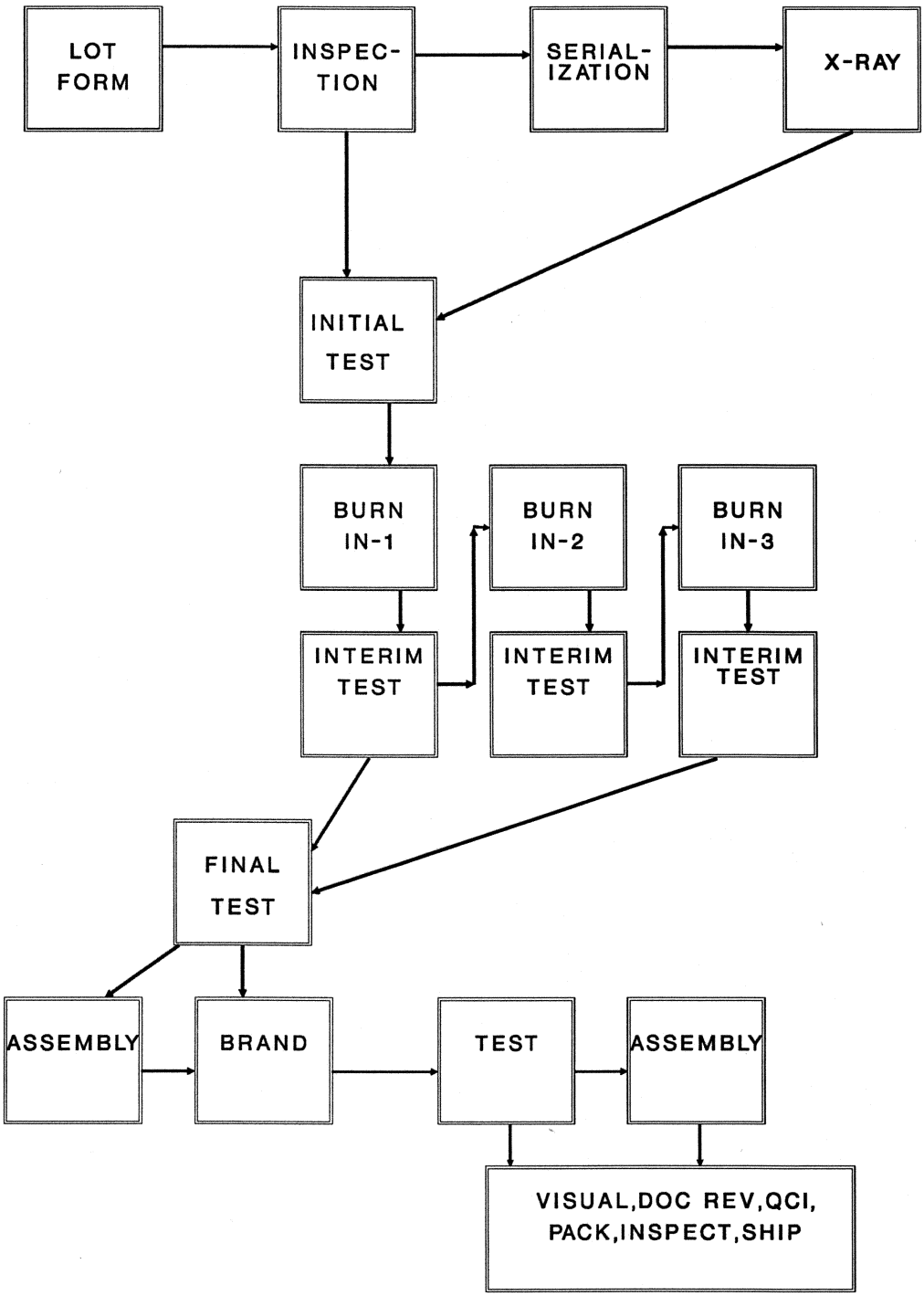

Figure 2.1. Product Routes in the Test Area.

The lot arrives at the lotform area with an assembly traveler that describes the history of the lot up to that point. A lot is started through the test floor when it is needed to meet back-ordered demand or when the area planner includes it on the weekly release plan. The test traveler for this product is printed, and engineering information is extracted from the database and attached to the traveler. The tubes of devices are placed into a box, and the box is moved to a waiting area.

The lot is then serialized. Done in the brand area, serialization is the process of giving each device a unique serial number. This process is labor-intensive as someone must stamp each device with a branding machine.

The lot then undergoes X-ray testing. This operation includes six tasks: loading the devices into racks, shooting the X-rays, unloading the devices, developing the X-rays, reading the X-rays for internal defects in the package assembly, and culling the rejects. The lot may experience significant waiting before and after the reading of the X-rays. Three people handle the lot: one who loads and shoots, one who reads, and one who culls.

Electrical testing, the next step, is an important, machine-intensive operation. During an electrical test, the lot is processed by a handler machine that positions each device over a contact with the tester, a computer that performs a number of tests by subjecting the device to a number of different electrical inputs. After the test of that semiconductor is done, the device is fed to an output bin. The test floor contains a number of different types of testers and different handlers. Since each product requires a different combination of handler and tester, significant setup time can be incurred if the operator must find and install a handler to process the lot.

After the initial test, the lot must be burned-in. The devices in the lot are loaded onto boards that hold from 10 to 100 devices each. These boards are placed into ovens, where the devices are subjected to thermal stresses and electrical inputs in order to cause infant mortality. The burn-in period lasts a minimum of 24 hours, although no penalty is associated with keeping the devices in the oven for longer than this time. After the boards are removed from the oven, the devices are unloaded from the boards.

After the burn-in period, the lot returns to a tester and undergoes interim test. Interim test consists of testing at room, low, and high temperatures. For the low temperature test, a bottle of liquid nitrogen must be attached to the handler and the handler must chill the testing environment to the desired temperature. In a test at high temperature, the handler warms the test chamber to the desired temperature.

After the last burn-in and interim test, a final test is conducted. The lot then goes to the assembly facility, where fine and gross leak testing of the package seal is performed. During this time, the lot is out of the test facility's area of control.

The lot returns and moves to brand, where each device is stamped with the company logo. The devices are placed into pans and then sit through a three-hour bake in a kiln. After this step, testing is performed to verify that no damage was done to the devices during brand.

After another trip to assembly, the lot undergoes a visual test, where each device is inspected under magnification for obvious defects in the packaging. The lot enters document review next. At this time, all of the paperwork associated with the lot must be reviewed for completeness.

The lot then undergoes customer source inspection, where a representative of the customer personally supervises the testing of a sample of the lot. In the shipping operation, the devices are placed in boxes and the boxes are addressed, ready for delivery to the customer.

2.2 Semiconductor Scheduling

The manufacturing of semiconductors has received much attention form production planning and scheduling researchers. This section reviews a number of papers that address the issue of semiconductor production planning and scheduling directly or indirectly. As in Uzsoy, Lee, and Martin-Vega (1992a, 1993), the papers are classified into the following topics: production planning; shop floor scheduling: dispatching rules and work release, deterministic scheduling, batch processing, control-theoretic approaches, knowledge-based approaches, and simulation; and performance evaluation: queueing models, and simulation. This review is

intended to display the numerous methods that have been used in the research of production and scheduling issues of semiconductor manufacturing.

2.2.1 Production Planning

A number of authors have addressed the large-scale problem of production planning for the semiconductor industry. The most common approach has been a hierarchical decomposition of the problem.

Leachman (1986) describes a corporate-level production planning system. This system divides the manufacturing process into the stages of wafer fab, probe, assembly and test, linked by inventories. Its model may include multiple facilities. The production processes in a plant are treated as a single entity, and problem complexity is reduced by the aggregation of products that are in the same process at each stage.

Information from corporate databases is used in a linear program with dynamic production functions that capture the relationships between the processes. This yields a production plan. Through a set of linear programs, the aggregate plan is then decomposed into a capacity-feasible weekly start schedule for various facilities. In a later work, this system was implemented by Harris Semiconductor.

Hadavi and Voigt (1987) describe the planning system for a Siemens development wafer wafer fab. In their approach, they create different levels of abstraction for different levels of planning and localize rescheduling while making minimal resource constraints.

The system architecture is a hierarchy of constraint sets that represent different time windows. Quarterly requirements are divided into months, weeks, and days. Arriving orders start feasibility analysis (can it be done?) and then a scheduling heuristic (when will it be done?) that tries to satisfies the constraint sets (based on the hierarchy). Scheduled orders are sequenced by an algorithm that maximizes throughput. A rule-based expert system recovers from disturbances by local rescheduling.

Harrison, Holloway, and Patell (1989) discuss the production planning question and present a case study of National Semiconductor Corporation. They list the following programs that can improve customer service: management information systems; logic and algorithms used to generate delivery quotations and to schedule production; and performance evaluations and incentive systems based on measures of delivery performance. They place the emphasis on operational decision-making, from booking orders to scheduling lots on machines.

The authors also consider performance measurement, which should give the upper management information about the different groups, all of whom are acting to improve their performance on these scales. These measures should encourage mangers to act optimally for the company.

The authors make the following comments about scheduling. Marketing wants commitments to be inviolate; production planners thus want control over factory loading. Managers view scheduling as a MIS problem that provides reliable routine execution. The company must get the right information to the right people and then coerce them to do the right thing.

Golovin (1986) describes various attempts to solve the problem of production planning and factory scheduling. Mathematically, an integer programming problem of the factory scheduling problem considers all costs. The solution philosophies are diverse and involve certain tradeoffs. Mathematical optimization gives the best solution but is computationally costly in data and time and ignores local conditions. Dispatching rules are concerned only with the present. They may make poor scheduling decisions by ignoring global conditions, but they work quickly and cheaply. Just-In-Time maintains a balance of work in the factory and attempts to maintain a high level of quality.

The authors find most promising hierarchical systems that decompose the problem into sets of decisions made at different times: capacity planning, release planning, and lot dispatching. This approach may yield suboptimal policies but it gives control of decisions to the users who are responsible for the results. Scheduling is done under the assumption that sufficient capacity

exists. Dealing with uncertainty in yields and equipment calls for keeping safety stock of standard product and buffers in front of each machine to prevent starvation.

2.2.2 Shop Floor Control

Methods used to control the shop floor vary widely in their approaches to solving the problems encountered there. The basic areas include lot release policies, dispatching rules, deterministic scheduling, control-theoretic approaches, knowledge-based approaches, and simulation.

The production of detailed shop schedules for planning and shop floor control has also been considered by researchers looking for some way to go beyond materials requirements planning, which does not consider capacity constraints, and to search for schedules better than those found by dispatching rules.

Vollmann, Berry, and Whybark (Chapter 5, 1988) discuss three simple ways in which shop floor control can be done: Gantt charts (often created backwards from the job due dates), priority sequencing (dispatching) rules, and finite loading schemes, which create a schedule for the time horizon by simulating the operation of the shop.

Bai and Gershwin (1989) initiate a discussion of all important phenomena in semiconductor fabrication and characterization of all events and scheduling objectives and factory types. They start with the observation that two types of events exist: controllable and uncontrollable. Controllable events are those that the person in charge of scheduling the shop can start. The state of the system is a complete description of the production variables, such as the status of each machine and each worker. The variables in the state are chosen by the scheduler for his purposes, and he may ignore certain inconsequential quantities. Events change the state of the system, and the current state limits the options of the scheduler, who must make a decision based upon the state.

The authors note that schedules in the real world are subject to disruption by uncontrollable events and that schedulers have three weapons to reduce the effects of disruption: real-time

scheduling systems, prediction, and inherently-robust schedules. The first is the most powerful, but the computational effort makes combinatorial optimization impractical; thus, scheduling heuristics are necessary.

The paper includes a summary of the semiconductor manufacturing processes and a detailed description of wafer fabrication. This description includes an explanation of the different machines, workers, and events in the fabrication process. The uncontrollable events are classified into those that are predictable and unpredictable. The latter include machine failure and defective wafers and are the ones that make life difficult for schedulers. The authors also include the constraints on scheduling and the objectives of scheduling, which are the minimization of WIP, throughput, variability in throughput, and costs and the meeting of demand.

Lot release policies and dispatching rules. A number of researchers have realized that the release of work into the shop has large ramifications on the performance of that shop. Dispatching rules are in use in many shops, and some work has been devoted to developing better rules.

Wein (1988) considers the problem of minimizing cycle time in a semiconductor wafer fabrication. His approach is to study the input regulation policy. He studies four alternatives: no control (Poisson arrivals); uniform start policy (constant release rate); constant WIP (closed loop); and workload regulating. This last rule focuses on the heavily-utilized workstations and uses a Brownian network model to approximate a multiclass queueing model with dynamic control capability. A lot is released when the work in the system for a bottleneck machine falls below a certain level. Bottleneck machines use various dispatching rules while other machines use FIFO. The workload regulation rule was found through simulation studies to reduce the mean and variance of cycle time, and the effects of dispatching rules were less significant and varied by system and input type. The study uses data gathered at Hewlett-Packard Technology Research Center in Palo Alto.

Glassey and Resende (1988 a,b) examine a wafer wafer fab with a single bottleneck workstation, single product, and constant demand. Their approach is to control input regulation

by considering global information. The measure of performance is the cycle-time versus throughput curve. Their answer is to release jobs so that they arrive at the bottleneck just in time to avoid starving that machine (starvation avoidance). This requires the determination of the virtual inventory, the amount of work for the bottleneck machine that is there now or will be soon (within the lead time it takes a released job to reach the machine). If this inventory is less than the lead time, release a job. This is similar to a safety-stock inventory policy.

Simulation studies reveal that starvation avoidance results in lower cycle-time vs. throughput curves than other release rules: uniform, fixed-WIP, and workload regulation. Solorzano (1989) describes the implementation at Harris Semiconductor.

Glassey and Petrakian (1989) consider the problem of minimizing the queue of a bottleneck machine in a shop using starvation avoidance input regulation. They do so by using queue predictions and dispatching rules that give higher priority to lots that should encounter a shorter queue on their next visit to the bottleneck. This extra queue prediction computation is intensive, although certain simplifying assumptions are made. Object-oriented simulation studies show good behavior (better than dispatching rules such as SPT and FIFO) in both a one-product wafer fab and a two-product wafer fab.

Wein and Chevalier (1992) take a broader view of the job-shop scheduling problem by considering three dynamic decisions: assigning due-dates to arriving jobs, releasing jobs from backlog to shop floor, sequencing jobs at each workstation. Their objective is to minimize WIP and due-date lead time, subject to an upper bound constraint on fraction tardy. They take a two-step approach: (1) release and sequence jobs to minimize WIP subject to throughput rate; and (2) set due dates that minimize due-date lead time.

The authors propose three principles: (1) while maintaining fraction tardy, average due date lead times can be reduced by dynamically basing due-dates on the status of the backlog and shop floor, the type of arriving job, and the job release and sequencing policies used; (2) without affecting the throughput, WIP can be reduced by regulating the amount of work on the shop floor

for bottleneck stations; and (3) better due-date performance can be achieved by focusing on efficient system performance and ignoring due dates when sequencing.

The proposed job release policy is to inject a customer into the shop whenever the workload at the bottlenecks is at a certain level, determining the customer by a workload balancing input heuristic. Priority sequencing uses dynamic reduced costs from an LP. In step two, due dates are set using rough approximations that follow the spirit of principle 1. Simulation experiments considered a two-machine, two-product shop. The proposed policies beat standard policies.

Lee *et al.* (1993) describe a decision support system for shop-floor control in a test facility. The system uses the Short-Interval Scheduler (SIS) module of COMETS to perform dispatching. The main contribution of the work is the development and implementation of a mechanism that considers sequence-dependent setups while making despatching decisions. This is done by classifying the setups and assigning each operation a setup code representing the setup configuration (determined by handler, temperature, and package type). This allows the operator to select operations with desirable setup characteristics in addition to the due date or operation type allowed by COMETS. This system has been implemented in a test facility and has been running successfully for over two years.

Deterministic scheduling. Under certain conditions, or using simplifying assumptions, controlling the shop floor can be represented by a scheduling problem that can be solved deterministically. The solution to this problem gives a schedule that can be used on the shop floor.

The papers by Uzsoy, Lee, and Martin-Vega (1992b) and Uzsoy *et al.* (1991a, 1991b) consider the scheduling of a semiconductor test facility and the associated single-machine problems. This work addresses back-end and due-date issues. Their first approach is the use of the shifting bottleneck algorithm, which iteratively schedules workcenters based on some measure of criticality.

In another result, they use dynamic programming algorithms and heuristics to minimize maximum lateness and number tardy on a single-machine problem with sequence-dependent setup times. Finally, they minimize maximum lateness on a single tester with a branch and bound algorithm and local search improvements to heuristics. They also describe the prototype implementation of an approximation scheme for an entire test facility (Harris Semiconductor), which may be a practical alternative to dispatching rules.

Graves *et al.* (1983) study the problem of scheduling a production facility with re-entrant product flows for identical products. Their objective is to minimize average cycle time while meeting a target production rate. They develop cyclic schedules that process each operation of a job once every cycle, where the length of the cycle is the reciprocal of the production rate. Thus, in each cycle, one job is started and one job is finished. The scheduling of tasks on machines in the cycle is done with a greedy heuristic that maintains feasibility. The machines in the problem may be multi-channel or batch facilities. The authors compare their rule to the simple FIFO dispatching rule.

Kubiak, Lou, and Wang (1990) consider a re-entrant job shop with a hub machine, the machine to which jobs return repeatedly. They wish to minimize total completion time under the following assumptions: (1) the shortest operation on the bottleneck is longer than any other operation (allowing the single-machine simplification); and (2) the jobs possess a hereditary order, where a smaller total processing time implies a shorter processing time for any operation on hub machine. An optimal well-ordered schedule sequences jobs at the same operation for the hub machine by SPT. They develop a dynamic program and present a heuristic that develops clustered schedules, where groups of jobs scheduled together, finishing one cluster before moving to the next.

Lee, Uzsoy, and Martin-Vega (1992) study burn-in as a single-machine problem. They assume that each lot of devices has been loaded onto a number of boards, and each of these boards forms a job for a burn-in oven. If a batch of jobs is placed into the oven, the entire batch must remain in the oven until all of the jobs have been processed long enough. Thus, the

processing time of the batch equals the maximum processing time of the jobs in the batch. The authors examine the performance measures of maximum tardiness, number of tardy jobs, maximum lateness, and makespan. In this mathematical paper, the authors present dynamic programming algorithms to solve batch scheduling problems with release dates and parallel batch scheduling problems.

Bitran and Tirupati (1988 a,b) consider the problem of epitaxial wafer manufacturing. They identify the bottleneck as the epitaxial growth operation, which takes place in a number of different reactors. Their model is a single-stage parallel unrelated machine problem, with the objectives of makespan and total tardiness. They develop static scheduling heuristics that create schedules in two phases: product group priorities (by workload and due date measures) and then job priorities (by due date) within each group.

They also address the planning problem of assigning reactors to product groups in order to decompose the problem into independent shops. This approach reduces the complexity of the problem and results in the following observation: the choice of heuristic to solve the problem should be guided by the homogeneity of product set and the objective function. An implemented scheduling system provides shop floor schedule for each reactor, job status, lead-time estimates, and reactor load profile. Periodic resolving of model after planning preprocessing creates uniform reactor loads and homogeneous product mixes.

Control-theoretic approaches. In an attempt to find policies that perform better than standard rules, some researchers have created control-theoretic models. The solutions to these models are then used to manage the shop floor.

Bai, Srivatsan, and Gershwin (1990) consider a hierarchical production planning and scheduling system for a semiconductor wafer fabrication facility. They attempt to meet throughput goals while treating random disruptions explicitly. They integrate the scheduler with the system data base. Events in the wafer wafer fab are classified by their frequency and whether they are controllable (starting a lot vs. machine breakdown). Planning hierarchy is organized by these frequencies. Each level has events with the same magnitude of frequency, capacity

constraints, and objectives passed to lower levels. When something with a low frequency happens, the target frequencies of higher-frequency events are recalculated. The real-time scheduling is derived from control theory and treats random events as part of the system. This decomposition allows small but very detailed models. The approach is implemented at the MIT Integrated Circuit Lab. Recent results on this work are covered in Bai and Gershwin (1992).

Bai and Gershwin (1990) cover previous work on scheduling single-part, multiple-part, and re-entrant flow systems using a real-time feedback control algorithm. For single-part systems, the linear programming problem for the feedback control law divides the surplus space into regions where the optimal production rate is constant. There also exists a spot called the hedging point where the surplus is enough to compensate for any disruptions. The feedback controller attempts to drive the system to this point.

The authors measure the system's performance in keeping the amount of surplus close to zero with low buffer inventories. They also attempt to improve behavior by separating the machines from each other to reduce effect of machine failure. Solving a nonlinear program yields an optimal hedging point (minimizes buffer inventory and buffer sizes). The current buffer inventories and the hedging point are then used to calculate the desired production rate for the system. Loading times for machines are calculated heuristically to be close to the optimal production rate.

For multiple-part systems, machines are divided into single-part sub-systems with the same failure and repair rates and capacities proportional to the demand for that part. In re-entrant systems, machines are again divided by part and also by operation, with appropriate capacities calculated.

In Lou and Kager (1989), the authors consider VLSI wafer fabrication with the goal of reducing WIP while following target production and observing machine capacity. Their approach is to use lot release and lot dispatching control rules based on flow rate control, a stochastic optimal control problem. Assuming a continuous flow, the authors divide the shop into workstations and determine a production rate for each according to control rules that consider the

inventories across the floor and predetermined hedging points. The advantages of this model include reducing the dimension (by ignoring machines) and providing dynamic feedback control (by responding to surpluses, down machines). The authors compare their policy to an event-driven simulation using uniform-loading by costs of inventory at all stages, and claim that the flow-rate control reduces costs by 50%.

Gong and Matsuo (1990a) examine a multiperiod production system with random yield, with the objective of minimizing fluctuation of WIP inventory, leading to more predictable system performance. Their approach is to consider control rules for starting product into each stage and raw material. They report that intuitive control rules can be destabilizing. Their new control rule - Minimum Weighted Variance - is the optimal control policy for a stochastic dynamic program with a quadratic objective function that penalizes WIP deviation from targets and infinite time horizon. Rule performs well in systems close to capacity.

Gong and Matsuo (1990b) also consider a problem with multiple products, limited workcenter capacity, rework, and re-entry. Again, they wish to minimize weighted WIP variance, where the weights in the objective are determined by a nonlinear program that minimizes total expected WIP. With sufficient conditions on the variance of yield and rework fractions, the authors find an optimal control policy for the associated dynamic program. An important conclusion is that the development of stable control policies in uncertain environments is challenging and depends upon the yield and rework distributions.

In Ou and Wein (1991), the wafer fabrication system has a single bottleneck, multiple processes with re-entry, and by-products. The authors use for their model a single-server queue with job classes that correspond to different operations. The model's objective is to minimize total cost: the sum of holding costs for WIP (jobs waiting for re-entry) and finished goods and backorder costs. Their approach is to develop a control problem approximation involving Brownian motion. The optimal control policy can be interpreted as the optimal schedule. The authors compare their policy to two state-dependent heuristic policies with a simulation and find that their policy reduces costs.

Knowledge-based approaches. Many researchers have attempted to give managers better control of the shop floor with decision support and expert systems, which incorporate knowledge that can be used to set control policies or schedule machines. Some expert systems can even perform scheduling events themselves.

Adachi, Moodie, and Talavage (1988) consider a production system with re-entrant product flows. They use a simulation model to examine the effect of management decision variables on system performance. The variables are lot size, dispatching rules, start rate, frequency of urgent jobs, and frequency of machine breakdowns. The measures are product mix, throughput, cycle time, and WIP.

The authors report that the start rate was a more important factor than the dispatching rules. The lot sizes also had significant effects. The dispatching rules were important policies only in over-capacity shops. The authors then develop a pattern recognition DSS that, through an iterative procedure, helps the user find a system that matches the user-defined goals. A prototype is implemented on the printed circuit board fabrication line at NEC Corporation.

In Adachi, Moodie, and Talavage (1989), the authors extend their work on production systems with re-entrant product flows. This time, the decision support system includes a rule-based component and simulation model. The simulation model comes from the previous work and evaluates the effect of control variables on measures of inventory, product mix, cycle time, throughput. The results are used to obtain regression coefficients, which are stored in a database for the rule-based component, which organizes the control variables into the hierarchy of start rate, lot size, and priority rule. This DSS was implemented for a printed circuit board fabrication line, resulting in superior control policies to the previous pattern recognition DSS.

Adelsberger and Kanet (1991) report on a new tool in computer-aided manufacturing scheduling: the leitstand, a decision support system with a computer-aided graphical interface. Leitstands include the following components: the graphics component is a electronic Gantt chart; a schedule editor allows a user to easily change an existing production schedule; the data base manager incorporates data from production planning system, engineering, and shop floor and

uses specific knowledge; an evaluation component measures the schedules on objective functions and creates reports; and an advanced automatic schedule generator produces feasible schedules.

Savell, Perez, and Koh (1989) describe scheduling semiconductor wafer production with an expert system that takes advantage of the modularity of workcenters and sophisticated data analysis. The wafer fab is divided into into thirty cells. The expert system, which is an off-the-shelf PC package with some external routines for data manipulation, creates daily schedules. It includes two modules: (1) the priority assignment module uses information from CAM about the operational slack of each lot and knowledge about special lots; and (2) the equipment scheduling module for each cell uses specific knowledge about the cell and the lot priorities to create a schedule for the lots in the cell and those arriving at the cell within a certain time frame. The expert system is implemented in two cells (P-Diffusion and Aluminum Deposition) at Harris Semiconductor.

Sullivan and Fordyce (1990) report on IBM Burlington's Logistics Management System for wafer fabrication. The main function is the shop floor dispatching of lots. It replaces slack lead times with information to handle the coupling of strategic and operational decisions.

The LMS includes real-time lot and machine status and proactive intervention, with an expert system that has knowledge about generating alerts in certain conditions and responding to some of these by making dispatch decisions that consider five conflicting objectives: lot priority, on-time delivery, flow requirements, increasing throughput, meeting engineering specifications. The LMS is implemented in various areas and realizes the importance of accurate data and continual updating to reflect changing environment.

Fordyce *et al.* (1992) discuss the current version of the Logistics Management System (LMS) in place at the IBM Burlington Semiconductor manufacturing site. The emphasis of this paper is on the last of the four tiers of the scheduling decision hierarchy. This is dispatch or short-interval scheduling for periods from one hour to two weeks. This tier contains the decisions concerning the actual manufacturing flow, including the scheduling of an operation. The LMS contains a dispatcher/short-interval-scheduler, that creates zones of control around the bottleneck

points, which may a sequence of operations that lots must visit repeatedly. Kanbans within these zones monitor the WIP level for the different passes through the zone. The dispatching is done by something called a Judge. The Judge receives information from goal advocates in order to make its decision, and the different advocates have different goals, including meeting due dates, meeting the daily plan, maintaining low WIP (this is the goal of the kanbans), and increasing machine utilization (this includes minimizing setups).

Hadavi *et al.* (1991) present a distributed architecture for real-time scheduling and describe its implementation in a wafer fabrication factory. The system (ReDS) works to meet management objectives of meeting due dates, reducing WIP and finished goods inventory, reducing cycle times, and maximizing machine utilization. The system abstracts constraints and time into a tree with nodes that correspond to an interval of time; it also abstracts an order into an "essence function" that describes its critical resources. The release policy uses "continuity indices" to reduce cycle times. The modules in the system include a preprocessor to abstract the constraints and orders and a feasibility analysis to release orders. Also in the system are a detailed scheduling module that uses least commitment planning in determining a daily or shift schedule. A sequencer then dispatches the operations scheduled for a time period by using a dynamic sequencing rule that responds in real time to a changing floor.

Rao and Lingaraj (1988) review a number of expert systems for production and operations management decision making. They classify the systems along two dimensions: strategic decisions versus tactical decisions and operations orientation versus resources orientation. Scheduling systems are classified as tactical, operations-oriented applications. They review a number of systems in scheduling, capacity planning, facility layout, process & product design, quality control, aggregate planning, inventory control, and maintenance & reliability. They conclude that expert systems should combine technological knowledge and logistical data in order to be effective.

ISIS (Fox and Smith, 1984) is an expert system for scheduling that uses a hierarchical approach to scheduling: job selection, capacity analysis, resource analysis, and reservation selection. It uses decisions made at each level as constraints for the lower levels.

Opportunistic scheduling is a knowledge-based approach introduced in the OPIS system (Smith, Fox, and Ow, 1986, and Ow and Smith, 1988). In this system, the authors extend the job-oriented scheduler of ISIS with a machine-oriented scheduling procedure. After identifying an initial bottleneck, the search for a good schedule proceeds by directing activity towards the bottleneck subproblems of the job shop scheduling problem. The characteristics of this system include alternative problem decompositions, multiple scheduling heuristics, and multiple problem abstraction.

Sadeh (1991) discusses a system called MICRO-BOSS, another opportunistic scheduler that identifies an initial bottleneck and revises its strategy as new bottlenecks emerge during the construction of the schedule. This system is a more flexible procedure, however, for it can revise its scheduling strategy after each operation. Thus, it avoids having to scheduling large subproblems for a machine or a job.

Bensana, Bel, and Dubois (1988) describe a system called OPAL, a job shop scheduling software that combines three types of knowledge: theoretical knowledge about scheduling problems, empirical knowledge about scheduling heuristics, and practical knowledge of the scheduling environment. They claim that combining artificial intelligence techniques with operations research should be an effective approach to scheduling problems.

OPT and other approaches. OPT (Optimized Production Technology) is a proprietary scheduling system that focuses on the scheduling of the bottleneck resource. The system has been reviewed by a number of authors, including Jacobs (1984), Meleton (1986), Lundrigan (1986), and Vollman (1986). OPT uses a forward finite-loading scheduling procedure to schedule the identified bottleneck. The remaining, non-critical operations are scheduled using an backward infinite-loading procedure. According to Morton (1992), the advantages of OPT are its good solutions to large scheduling problems and its focus on the bottlenecks. Disadvantages

include the inability to do reactive scheduling without resolving the whole problem. Details about OPT can be found in the above articles and in Vollmann, Berry and Whybark (Chapter 20, 1988).

Faaland and Schmitt (1993) use a cost-based system to enhance the planning function of MRP in an assembly shop. They developed a system that can create a detailed schedule of jobs, workers, and work centers for a maker of aircraft audio, power, and light systems. They include inventory holding costs associated with finishing early, opportunity costs associated with late delivery, and payroll costs, since the model includes the cross-training of workers.

Morton (Chapter 16, 1992) describes an early version of bottleneck dynamics called SCHED-STAR (initially reported in Morton *et al.*, 1988). This system iterated lead times and prices over the bottleneck and considered an objective function that included revenue, tardiness, direct completion costs, and holding costs. A number of release and dispatch heuristics were studied on a variety of shop situations, and the authors conclude that bottleneck dynamics (and iterated pricing and lead times) leads to better schedules.

Simulation scheduling. Primarily, simulation models are used to predict the performance of certain policies. However, once the policies are set, the simulation model can be used to create a feasible schedule for the shop floor.

Atherton (1988) states that simulation can be used on all levels of planning. For short-term scheduling, models may perform short-interval and shop-floor scheduling. This form of scheduling considers factory capacity (dispatching rules don't). A validated simulation model, provided with current information on the system status, can use rules to determine what will happen in the short term, providing a shop floor schedule. If the projected lot completions do not meet production requirements, further simulations may be necessary to find a schedule that is close to the goal.

Leachman and Sohoni attack the problem of semiconductor manufacturing using an automated scheduling system and teamwork-fostering management. For every shift, a target is set that considers the real floor. The entire staff meets to identify problems and propose

solutions. The targets are generated by the scheduling system: a simulation model of the wafer fab in BLOCS that is linked to the WIP tracking system, providing it with real-time data. The simulation considers availability but assumes perfect execution. Thus, it is fair and accurate. To make the schedule, it uses logic accepted by the staff, including least slack, FIFO, and starvation avoidance rules and furnace waiting cost analysis.

The authors discuss how a shift proceeds: meetings before and after the shift with explicit incentives for meeting the target. They also review the organizational effectiveness of their approach in terms of motivation, satisfaction, communication and coordination, problem solving capability, acceptance of change.

Najmi and Lozinski (1989) discuss the implementation of the system in two wafer fabs at NCR, Inc. The simulation model is written in BLOCS, an object-oriented language where each physical object or collection of data is a object (composed of data and procedures), and the objects communicate to each other with messages that represent the interaction of objects.

2.2.3 Performance Evaluation

The complexity of semiconductor fabrication facilities makes direct analytical evaluation of them difficult at best. In many cases, researchers use queueing models or simulation models in order to gain some insight on how the system performs in the current or some proposed environment.

Burman *et al.* (1986) discuss the ways that OR tools and techniques are used to analyze IC manufacturing lines: simulation, queueing analysis, and deterministic capacity models. The authors describe a simulation study for direct step on wafer printers in photolithography. The study determined the effect of lot size, number of products, and setup time on capacity and WIP. It also used large-scale simulations to obtain arrival distributions to this area. A deterministic capacity model of a clean room uses the mean number of steps and the mean process time. The analysis took much less computing time than a simulation. The model was useful for estimating a room's capacity and determining the minimal number of machines required for a proposed

program. They conclude that simulation models have the greatest flexibility and best success but take considerable time to develop and run. The other techniques are quicker and, if not quite accurate, do provide estimates that may be verified in simulation.

Queueing models. Queueing models attempt to model the shop floor as a set of servers with service time distributions and possible stochastic routings. The system yields a set of equations that describe its behavior. Because the success of solving these equations is usually undermined by the complexity of the shop being modelled, researchers are forced to make simplifying assumptions.

Chen *et al.* (1988) develop a simple queueing network model to predict performance measures in an research and development wafer fab. The Hewlett-Packard Technology Research Center Silicon Wafer fab serves as the model. The model is a classic job shop, and the performance measures are throughput and cycle time. Machine breakdown rates added to actual service time to create an effective service time. The naive queueing network model includes different types of customers that have different routing distributions. Straightforward formulas from previous researchers yield performance measures. The authors report that the model yielded performance measures that were within 10% of the actual numbers.

Wein (1991) investigates a wafer wafer fab with time-dependent stochastic defects, using a simple queueing theoretic model to determine the relationship of yield and cycle time to throughput. The model is a single-server queueing system with exponential arrival and service times. The author derives a closed form relationship between the mean cycle time and the throughput of good die. In standard models, as the start rate is increased, the throughput increases and asymptotically approaches some upper bound as cycle times increase to infinity. In this model, the cycle times increase without bound but throughput reaches some maximum and then decreases as the increased cycle time leads to a higher defect rate (and thus less yield).

Simulation. Simulation models remain as a popular approach to measuring the performance of manufacturing system, and many packages have been developed. The state of the

art is probably the BLOCS package developed at California-Berkeley and partially described in Najmi and Lozinski (1989). The following papers report on some uses of simulation.

In two papers, Dayhoff and Atherton (1986 a,b) develop simulation models of wafer fabs. They model wafer fab as a queueing network with components for each lot and process flow. Models are used to analyze system performance, measured by its signature: the plots of cycle time, inventory level, and throughput versus start rate. The signature provides a way to gain information from the large quantity of data provided by a simulation by displaying the relationships among different variables. As a specific example, they model a bipolar wafer fab.

They compare the signatures for a number of dispatch schemes for a bipolar wafer fab where products visit the masking station seven times. In the wafer fab, there exist three levels of dispatching: (1) among lots of one product waiting for one process, (2) among process steps of the same product, (3) among products. This paper works at level two; the first level is done by FIFO, third by product priority. The study compares the following dispatch schemes for masking: earlier steps first (a form of longest remaining processing time), later steps first (a form of shortest remaining processing time), round-robin (changing priority). Qualitative analysis shows that earlier steps first failed at higher start rates by building inventory and that the later steps first was better by getting higher throughput with no increase in inventory.

Dayhoff and Atherton (1987) provide definitions of the elements of wafer fabrication and their interactions. They identify the important parameters that govern wafer wafer fab and result from wafer fab operations. The elements defined include work station, process flow, products, wafer lot, process step, batch, batching down, service, dispatch system, rework, and wafer fab graph. They discuss the following types of simulation results: average wait times, queue lengths, inventory, cycle time, equipment utilization, yield, throughput.

Atherton and Pool (1989) discuss the ACHILLES simulation model in a wafer wafer fab of Silicon Systems, Inc. The describe the model and its initialization, calibration, and validation by comparison to actual measures of factory performance. The discuss the use of the model to predict the effect of reducing inventory levels on cycle-times.

Spence and Welter (1987) discuss a project by Stanford University and Motorola Inc. that attempts to improve the performance of a semiconductor wafer fabrication line. The goal is to increase the throughput of the line without sacrificing cycle time. The analysis was done with a Monte Carlo simulation to determine capacity, defined as throughput rate, cycle time, and WIP. The performance was measured on the steady-state cycle-time vs throughput trade-off curve, instead of using transient signatures (c.f. Dayhoff and Atherton). The photolithography cell was studied for its analytical difficulty, the build up of WIP in this cell, and the proposal of system changes. The authors report that adding resources (operators, aligner equipment) reduced cycle time at higher throughput. Reducing reworks, setup times, and the time to wait for repair reduced cycle times at all levels. Larger lot sizes were preferable at higher throughputs.

2.2.4 Summary

From this review, it can be seen that many different approaches have been applied to the problems of production planning and scheduling. However, most of this work addresses the questions raised in wafer fabrication. The primary exceptions are Uzsoy, Lee, and Martin-Vega (1992b), Uzsoy *et al.* (1991a, 1991b), Lee, Uzsoy, and Martin-Vega (1992), and Lee *et al.* (1993) which do explicitly consider the problems of semiconductor test. However, this work does not make use of the new, sophisticated heuristic searches and is not concerned with the ideas of class scheduling and look-ahead and look-behind scheduling.

Although Savell, Perez, and Koh (1989) do develop a system that is look-behind, it is implemented in an off-line system in a wafer fabrication cell. The research in this dissertation, however, investigates the application of both look-behind and look-ahead rules into the current real-time dispatching system of a semiconductor test area.

2.3 Job Shop Scheduling

Before beginning the analysis of scheduling problems, it is useful to review the notation to be used and the basics of scheduling problems. Because even the simplest job shop scheduling problems are NP-complete, the literature consists of different heuristics.

Job shop scheduling, as one of the most difficult scheduling problems, has attracted a lot of attention from researchers. Techniques such as the shifting bottleneck algorithm (Adams, Balas, and Zawack, 1988) or bottleneck dynamics (described in Morton, 1992) concentrate on solving the problem at one machine at a time. Other researchers have studied how well different dispatching rules perform in minimizing makespan and other objective functions. Panwalkar and Iskander (1977) present a list of over 100 rules. Recent studies include Fry, Philipoom, and Blackstone (1988) and Vepsalainen and Morton (1988). Various scheduling systems for shop floor control are reviewed in Section 2.2. More sophisticated look-ahead and look-behind rules have also been introduced; see Section 2.5 for a discussion of these ideas. In this section we review scheduling notation, the shifting bottleneck algorithm, and dispatching rules.

2.3.1 Scheduling Notation

In the general job shop scheduling problem, there exists a set of jobs J_j, $j = 1, ..., n$, and set of machine M_i, $i = 1, ..., m$. Each job J_j has n_j operations (or tasks) O_{ij}, $i = 1, ..., n_j$, where $O_{ij} = k$ if the ith operation of J_j is to be processed on machine M_k. In a flow shop, all $n_j = m$ and $O_{ij} = i$ for all i. In general, each operation has a processing time $p_{ij} \geq 0$. A job can have release dates r_j and due dates d_j and deadlines D_j.

A feasible schedule σ is a plan that determines when each operation is processed on each machine subject to the following constraints: the operations of each job must be performed in order, and no machine can process more than one operation at at time.

For a schedule, certain performance measures can be associated with a job J_j: the completion time C_j, the lateness $L_j = C_j - d_j$, the tardiness $T_j = \max \{L_j, 0\}$, and whether or not

the job is tardy, $U_j = 1$ if $T_j > 0$ and 0 otherwise. Objective functions that measure the performance of the schedule include the makespan, $C_{max} = \max \{C_j\}$, the total flowtime ΣC_j, and the number of tardy jobs ΣU_j.

The scheduling problem is to find a feasible schedule that minimizes the objective function. For the regular performance measures mentioned above, the set of schedules that need to be considered is the set of active schedules, where no operation can be started earlier without causing another operation to be delayed. In this set, there exists a one-to-one correspondence between a schedule and its representation by a sequence of operations for each machine. Thus, a sequence that orders the jobs on each machine can be considered a solution to the problem.

Standard scheduling problems can be classified in a concise way by using the following three-element description: $x / y / z$. The field x describes the machine environment as one-machine, parallel-machine, or shop. The second field (y) describes any constraints or special characteristics of the problem. The third field (z) describes the objective function. Consider the following examples:

$1 / r_j / \Sigma U_j$ is a one-machine problem where the jobs have release dates and the objective is to minimize the maximum lateness.

$1 / D_j / \Sigma C_j$ is the one-machine problem where the jobs have deadlines and the objective is to minimize the total flowtime.

F2 $/ / C_{max}$ is the two-machine flowshop problem of minimizing the makespan.

J $/ / C_{max}$ is the general job shop scheduling problem of minimizing makespan.

2.3.2 Shifting Bottleneck

Adams, Balas, and Zawack (1988) introduce the Shifting Bottleneck procedure to minimize the makespan of job shop scheduling. This algorithm sequences the machines in a job shop successively by identifying the machine that is a bottleneck among the machines not yet sequenced. After the scheduling of the new bottleneck, all of the previously-considered machines

2.3.3 Dispatching Rules

Because of the complexity of job shop scheduling, algorithms to find the optimal solution for any arbitrary objective function do not exist. Thus, researchers have studied and schedulers have used dispatching rules to order the jobs waiting for processing at a machine. Most of the research in this area has examined the performance of various dispatching rules, which sequence the jobs that are waiting for a machine according to some statistic that is a function of the jobs' characteristics.

When a machine becomes available, it chooses from among the jobs in its queue by using a rule that sorts the jobs by some criteria. Common dispatching rules employ processing times and due dates in simple rules and complex combinations.

These dispatching rules are sometimes extensions from simple one-machine problems. For instance, the Shortest Processing Time (SPT) algorithm is known to minimize the total flowtime of jobs processed on one machine. The SPT dispatching rule sorts jobs waiting for a machine by the amount of processing time they require on the machine. Also, the Earliest Due Date (EDD) algorithm is known to minimize the maximum lateness of a set of jobs being processed on one machine. The EDD dispatching rule is used in job shop scheduling in an attempt to reduce maximum lateness. While the list of known dispatching rules includes over 100 items, only a handful are commonly used. And most of the rest are combinations of the most common rules.

Day and Hottenstein (1970) present a review of sequencing research, in which they discuss Jackson's Decomposition Principle (1967), which assumes that the arrival times for each job arriving from outside the system are exponentially distributed, the processing times at each machine are exponentially distributed, the jobs are routed to a machine by a fixed probability transition matrix, and the priority rule at each machine is First-Come-First-Served (FCFS).

They also discuss several due date assignment schemes presented by Conway (1965). These are CON (constant from the order to the due date), RAN (random: due date chosen by customer and accepted by salesman), TWK (Total Work Content: the allowable shop time is

proportional to the sum of the processing times of the operations of the job), and NOP (Number of Operations: the allowable shop time is proportional to the number of operations).

The authors also review the previous research on dispatching rules by Conway (1965), who compared SI to 31 other rules and found that it dominated the set of rules tested. It is simpler and easier to implement. However, the biggest objection to SI is the following fact: if the mean time which jobs spend in the system is small, individual jobs (those with long operations) may be intolerably delayed. Thus the variance of the lateness distribution is the basic disadvantage of the SI rule. Conway tried three methods to reduce the variance: (i) Alternating the SI rule with a low variance rule (with respect to flowtime) to periodically clean out the shop, (ii) forcibly truncating the SI rule by imposing a limit on the delay that individual jobs will tolerate; and (iii) dividing jobs into two classes, preferred and regular, as the primary criteria, leaving SI as a secondary criteria.

The authors then move to the COVERT dispatching rule. Buffa (1968) retained the performance of the SI rule and tended to minimize the extreme completion delays of a few orders. Trilling (1966) used a ratio of delay cost rate of a job to the processing time of that job on the machine in question. The job with the highest ratio is dispatched first.

The authors also report on the priority rules employed in industry, where job lateness is a primary concern. Hence, EDD and least slack rules are most used. However, LPT becomes a popular rule because schedulers rank jobs by the index of importance and it is reasonable to expect some positive correlation between importance and processing time.

Holloway and Nelson (1974) study the problem of job shop scheduling with stochastic processing times. Their performance measures are the mean, variance, and maximum of tardiness. For dispatching, the authors use HSP, a multi-pass heuristic that produces delay schedules, HSP-NDT (a non-delay version), DDATE, SPT, SLACK, and a SPT-SLACK combination.

The authors study three problems of different size, tightness, and utilization, where the processing times were constant or from one of three distributions. They report that HSP did well

on low-variance problems and on maximum tardiness. HSP-NDT beat HSP in some situations. Of the non-delay rules, the deterministic problems were a good predictor of relative performance on problems with variability. DDATE improved its relative performance as variability of processing times increased.

Panwalkar and Iskander (1977) list 100 dispatching rules. They categorize these rules into five classes: simple dispatching rules, combinations of simple rules, weighted priority indexes, heuristic scheduling rules, and other rules. The simple dispatching rules are organized into processing time, due date, number of operations, cost, setup time, arrival time, and machine rules. Combination rules sequence jobs by considering first one characteristic and then another. Weighted priority indexes sequence jobs by combining values from different job characteristics (by adding or dividing).

Blackstone, Phillips, and Hogg (1982) state that the best measure of performance is cost effectiveness. Their work covers this objective function and others: tardiness, lateness, flowtime, inventory. They note that analytic measures depend upon Jackson's decomposition, which assumes a FIFO dispatching rule. Other rules lead to interrelated queues and thus researchers use simulation.

For the single-server model, it is known that SI (shortest imminent processing time) minimizes mean flowtime and mean lateness and EDD (earliest due date) minimizes maximum lateness and maximum tardiness. Mean tardiness cannot be optimized by any dispatching rule; thus, the authors conclude that "no single dispatching rule yet developed will optimize delay costs in the job shop environment."

The authors consider a number of dispatching rules. Their first is SI. They report that SI is not affected much by imperfect data, it performed best on mean flowtime, and it minimized the number of tardy jobs and mean lateness for exogenous (constant and random) due date assignment. It also performed better when internal due dates are less than seven times processing time and utilization is high. Modifications to SI to clear jobs that have been waiting include

alternating rules and truncation. Truncation versions seem useful for shops having control over due dates and concerned about very late jobs.

The authors also consider a number of due date rules: EDD, Slack, and Slack-per-operation. Slack-per-operation is the best and, compared to SI or FIFO, produces a smaller variance of job lateness independent of due date assignment. Compared to EDD and slack, it performs best on lateness variance, cost-per-order, job tardiness, number of tardy jobs. Slack may be defined in static or dynamic terms, although rules that use the latter are better. The authors also report that critical ratio rules are also in use.

Among rules that used neither processing time nor due date information are FIFO and NOP (number of operations). The authors report that both perform worse than other good rules. They also conclude that the look ahead-rules NINQ (number in next queue) and WINQ (work in next queue) are not as good as SI in flowtime and inventory criteria and that a value-based rule usually becomes longest processing time, which generally behaves poorly.

The authors report that weighted combinatorial rules are not better than any single rule, although COVERT (delay cost over time remaining) may be useful in shops willing to estimate delay costs. The authors also discuss dispatching heuristics. The look-ahead heuristic LAH allows insertion of idle time in order to process a critical job. Heuristics improve the performance of dispatching rules but their implementation may not be cost effective; a complete study has not been done.

Green and Appel (1981) examine the problem of job shop scheduling by asking the following questions: What traditional dispatching rules do experienced schedulers select? Would dispatch rule selection be influenced by urgency? Would schedulers select a dispatch order based on organizational influence and/or peer pressure? The authors asked schedulers in a number of plants to denote which of the following rules they used: Due Date, Slack, Operations Due Date, Slack per Operations, SPT, FCFS, COVERT, Program in Greatest Trouble (PGT), or Friend Needs a Favor (FNF). The authors report that influence systems affect scheduling. PGT (a

coalition rule) was highly valued, but FNF (an individual rule) was rejected. Traditional and theoretical rules were not highly valued.

Kanet and Hayya (1982) also consider the problem of job shop scheduling. Their paper tries to determine if operation due dates are better. Their comparison is done by running controlled simulation experiments using the following dispatching rules: Earliest Due Date: DDATE and OPNDD; Smallest Job Slack: SLACK and OPSLK; Critical ratio: CR and OPCR. The slack is static; the critical ratio is dynamic: time until due-date over remaining allowance or operation allowance. For critical ratio, each operation must be assigned an allowance. This allowance is proportional to the processing time. The allowance multiple controls the difficulty of meeting due dates.

The authors consider the performance measures of lateness (mean and standard deviation), fraction tardy, conditional mean tardiness, flowtime (mean and standard deviation), and maximum tardiness. The authors report that all rules were outperformed by their operation counterparts on all measures. Introducing operation due dates shifts distribution of job lateness to the left (less) and compresses it. One unexpected result was that OPNDD (due date) outperformed OPSLK (slack) and always minimized maximum tardiness. The CR rule gives lower lateness variances, but the OPCR rule was even better. The authors note that, as the allowance multiple is increased, due dates become larger and DDATE and SLACK rules act more like SPT, minimizing flowtime. For the critical ratio rules, a larger allowance means longer jobs get lower ratios, causing CR to act like LPT.

Baker and Bertrand (1982) take up the problem of single-machine dynamic scheduling and study different due date assignment methods and dispatching rules. They are concerned with average tardiness. The authors introduce a new dispatching rule: the modified due date rule (MDD), where the modified due date is defined as the maximum of the due date and earliest finish date. This rule is a combination of EDD and SPT that implicitly responds to changes in the amount of slack. The authors report that MDD dominates both SPT and EDD on average tardiness.

Baker and Kanet (1983) extend the MDD rule to attack the problem of job shop scheduling. They consider the performance measures of mean job tardiness and proportion of tardy jobs. They introduce the modified operation due date rule (MOD), where the modified operation due date is the maximum of operation due date and earliest operation finish time. The authors also consider other dynamic rules: Slack per operation, Critical ratio, and COVERT under different allowances and utilization levels.

The authors report the following results: MOD performs better than MDD and SPT and is sometimes better than operation due date, critical ratio, and slack per operation rules. The authors thus conclude that MOD is an important dispatching mechanism.

Baker (1984) considers the job shop scheduling problem and attempts to clarify some conflicting results between different rules of dispatching and due-date assignments. He points out that two factors are of primary interest: flowtime and due-date performance. He reports that SPT is the best tactic for reducing mean flowtime. However, no single priority rule dominates performance comparisons. He reports that the critical ratio is best for minimizing conditional mean tardiness (that is, the average tardiness of tardy jobs); SPT is best for number of tardy jobs; however, for mean tardiness, the results are mixed.

Fry, Philipoom, and Blackstone (1988) consider the problem of job shop scheduling with 90% utilization. For the due date assignment, they use the total work method with two different allowances. They consider the following performance measures: mean flowtime, mean tardiness, and root mean square tardiness (designed to punish large tardiness).

They study truncated and alternating versions of the SPT dispatching rule. The authors define the critical percentage (%C) as the percentage of jobs that enter a higher priority queue because they have been waiting a long time. They report that the best rule depends upon the this critical value.

Vepsalainen and Morton (1987, 1988) consider the problem of minimizing weighted tardiness in job shop scheduling. They use the following dispatching rules: FCFS, EDD, Slack per remaining processing time (S/RPT), WSPT, weighted COVERT, and Apparent Tardiness

Cost (ATC). A lead time estimation is necessary for last two rules. Lead time estimates are computed as a fixed multiple of processing times, derived from rough waiting-line analysis and job priority indices, or found from iterating simulations.

The authors first studied the flow shop problem, where ATC and COVERT dominated using any lead-time estimate. Using the iteration estimate resulted in better weighted tardiness. ATC did better on fraction of jobs tardy and inventory measures. In the job shop, under different loads and due date tightnesses, the authors report that the ATC rule did better, followed closely by COVERT. The use of the priority estimates yielded a smaller fraction of jobs tardy.

2.3.4 Summary

In addition to introducing some notation, this section covered two important methods of job shop scheduling: the shifting bottleneck procedure and dispatching rules. Both methods have limitations, however. The shifting bottleneck procedure searches for a schedule and continuously improves upon it. Thus, it is only a local search technique. The dispatching rules that have been studied are mainly short-sighted techniques that do not consider other machines. The research in this dissertation extends this work by considering smart-and-lucky searches and investigating look-ahead and look-behind dispatching rules.

2.4 Flow Shop Scheduling

This dissertation considers a three-machine problem where all jobs are processed on a certain machine and then go to one of two second-stage machines. This problem is similar to a flow shop, and it was useful to review the flow shop scheduling problems that have been previously studied.

The flow shop problem is actually a collection of problems that deal with the minimization of some regular objective function for a set of jobs in a flow shop, where each job consists of a number of different operations that must be processed on a set of machines. The primary feature

of a flow shop is that the sequence of operations is the same for each job. That is, each follows the same path through the shop.

All flow shop analysis starts with Johnson (1954) who studied the minimization of makespan for two-machine flow shop problems and for some special three-machine flow shop problems. His algorithm starts jobs with the smallest tasks on machine 1 as soon as possible and jobs with the smallest tasks on machine 2 as late as possible. For the two- and three- machine cases, it can be shown that only permutation schedules need be considered. Permutation schedules are schedules for the machines in which the sequence of jobs is the same for each machine. Johnson showed that for four machines, passing may be necessary for optimality.

Although most analytical research in shop scheduling has dealt with the makespan objective function, this research is interested in the minimization of total flowtime, also known as the mean completion time, the sum of completion times, the mean time in system, or the total time in system. Therefore, this section concentrates on a number of papers on the minimization of total flowtime, a less-commonly studied objective, and then moves on to other objectives, including maximum lateness, and the number of tardy jobs. Also included in this section are reports on some NP-completeness results and problems with separated setup times.

2.4.1 Makespan

Special cases of the flow shop makespan problem have been studied by a number of researchers, including Mitten (1958), Conway, Maxwell, and Miller (1967), Burns and Rooker (1975), and Szwarc (1977). Garey, Johnson, and Sethi (1976) proved that the general three-machine problem was NP-complete. Problems with release dates, preemption, precedence constraints, or more than three machines have also been studied.

2.4.2 Total Flowtime

Ignall and Schrage (1965) describe a branch-and-bound algorithm for F2 // ΣC_j and for F3 // C_{max}. For the total flowtime problem, the authors consider two values: the sum of

completion times if the first machine is the only capacity constraint and the sum if the second machine is the capacity constraint. The larger of these two values is the bound at each node.

Krone and Steiglitz (1974) consider the static flow shop scheduling problem with the objective of minimizing the mean flowtime of n jobs that must visit m machines. They present a heuristic method. The authors note that two previous researchers performed separate sampling experiments of the flow shop scheduling problem. Heller concluded that permutation schedules should be investigated while Nugent found that good schedules tended to be permutation schedules that had allowed some jobs to pass others (a local reversal).

For the flow shop problem, the authors consider semiactive schedules (where no operation can be shifted forward in time) and define a schedule as an array S that gives a sequence for each machine. S_{ij} is the job that is jth on the ith machine. The authors define a two-phase heuristic. In the first phase, the search considers only permutation schedules. In the second phase, the search takes an initial permutation schedule and allows deviations from the uniform ordering. Each phase is a local search, although the neighborhood structures are different for each phase.

Kohler and Steiglitz (1975) study the two-machine flow shop problem with the objective of minimizing the mean flowtime. They present algorithms that they combine with lower bounds that guarantee the accuracy of the heuristics. The authors use the lower bound of Ignall and Schrage. They compute an initial solution by moving down a branch-and-bound tree without backtracking, selecting the node with the lowest lower bound at each stage. They consider different local searches, with either a random start or the above initialization, hillclimbing (first improvement), and one of four neighborhood structures: backward insertion, forward insertion, a double adjacent pair interchange (switches two pairs), and finally the union of the first and third.

The authors report that the good initialization helped the local search perform better than the random starting search.

The authors then present three different branch-and-bound algorithms that differ in the way the nodes are eliminated. Since some of the algorithms exceed computational limits, they determine a bracket that is the relative difference between the final upper bound and the greatest

lower bound. If the goal is to find a solution that is within a certain amount of the optimal, this bracketing idea allows great speedup since a good (sometimes optimal) initial upper bound was used and the lower bound is very close. For a given $r < 1$, the algorithm stops when the lower bound L is greater than or equal to r times the best upper bound U, i.e., $rU < L <$ optimal $< U$.

Miyazaki, Nishiyama, and Hashimoto (1978) present a heuristic algorithm for F // $\sum C_j$. Given a sequence, the authors interchange two adjacent jobs and determine the change in flowtime. They then derive a number of conditions that are sufficient to say that the second job should never directly precede the first. Using these different conditions they create temporary sequences and assign each job an ordinal number corresponding to its place in these sequences. The jobs are then sorted by the sum of these values. (more here)

Miyazaki and Nishiyama (1980) extend this work by creating permutation schedules for the weighted mean flowtime flow shop problem. The authors derive some precedence conditions and create an algorithm to generate good solutions. The first analysis considers the effect of interchanging two jobs in the permutation sequence to create. If this switch causes no decrease in the weighted flowtime, then the second job cannot directly precede the first in an optimal sequence. The authors then embark on finding a number of relations for two jobs that are independent of all other jobs and that will determine if a job should precede another.

For their algorithm, the authors evaluate these various relations for each job. If each of the relations yields the same sequence, this sequence is optimal. If not, each job is given a value for each relation that is its performance on the relation (1=best, . . .). These values are added over the relations and the jobs are ordered by these sums.

Szwarc (1983) discusses the mean flowtime flow shop problem and a number of special cases. The author develops two properties that create a precedence relation between two jobs, the first for a fixed presequence, and the second for an arbitrary presequences. Using these properties, the author presents a special case of a 2-job problem, and a case where the final completion times depend only on the completion times on some earlier machine. This last case

holds for the two-machine case where the $t_{1r} \geq t_{2r}$ for all jobs J_r and the case where the columns and rows form a perfect order under dominance.

Ahmadi *et al.* (1989) study the two-machine flow shop subproblem with one or two batch processors as a segment of a larger job shop. They study the objective functions of makespan and total flowtime. They consider two types of processors. The first is a batch processor that has a fixed capacity and the time necessary to process the jobs in a batch is a constant. The other type of processor is called a discrete processor, which refers to a standard single-machine processor. The authors consider the six types two-machine problems that can be formed.

To minimize the makespan of a batch-discrete shop, full batches are optimal, permutation schedules dominate non-permutation schedules, and the optimal solution is to sequence jobs by LPT and fill each batch. For the discrete-batch shop, the analysis is similar, but in this case the jobs should be ordered by SPT. For two-batch-machine-shop, all of the jobs are equivalent. The optimal policy is to completely fill the batches on machine one and to use a dynamic program to fill the batches on machine 2, using the completion times on machine 1 as release dates.

To minimize the total flowtime of a discrete-batch shop, SPT should sequence the jobs on the discrete machine 1. A dynamic program can be used to perform the batch dispatching on machine 2. For the two-batch-machine-shop, the batches on machine 1 should be completely filled and the completion times from this can be used as release dates to batch machine 2, which is dispatched using the same dynamic program. For the problem where a discrete processor is fed by a batch machine, the batches should be completely filled, but the authors prove that this case is strongly NP-complete. They do consider special cases relating the processing times on one machine to another and derive two optimal algorithms.

For the NP-complete problem, the authors propose a heuristic determines the number of shortest batches that are shorter than the batch processing time. To these batches the shortest jobs are dealt. The remaining jobs are sequenced by SPT to form batches. The authors show that the error tends to 1/2 as the number of batches tends to infinity.

Ahmadi and Bagchi (1990) present a lower bound on the total flowtime. Given a partial schedule J_r of r jobs and fixing a machine j, the authors bound the final completion times based on the completion times on machine j. Then, they search over sequences on machine j, considering the sequence constraints and the machine j release dates for each job. This is the NP-hard problem of solving $1/r_j/\Sigma C_j$. The authors solve the problem by allowing preemption. The lower bound is the greatest found by performing this procedure over each of the machines.

Van de Velde (1990) studies the two-machine flow shop problem and minimizes the sum of completion times by using a Lagrangean relaxation. The author decides to relax the precedence constraints between the two operations of a job (instead of the capacity constraints on each machine). Let the vector of multipliers be $\lambda = (\lambda_1, \lambda_2, ..., \lambda_n)$. The relaxed problem is one of minimizing the weighted sum of operation completion times, solved by SWPT on each machine separately. The author, however, wants to restrict the solution to permutation schedules.

Thus, the author reformulates the problem into a linear ordering problem, which is polynomially solvable in certain cases. The author lets all $\lambda_i = c$, where $0 \leq c \leq 1$, and if $c = 0$ or $c = 1$, the problem yields the bounds of Ignall and Schrage. Thus, for c, the relaxation can be solved optimally with a permutation schedule. The author claims that the lower bound as a function of c is continuous, concave, piecewise-linear function, and the maximum can be found in polynomial time.

One partial sequence dominates another if the sum of completion times is smaller and the makespan is also smaller. The author develops a number of sufficient conditions for dominance. Also, if $p_{2i} = p_{2j}$ and $p_{1i} \leq p_{1j}$, then the author proves that there exists an optimal sequence where job i precedes job j.

2.4.3 Maximum Lateness (Lmax)

Masuda, Ishii, and Nishida (1983) find a solvable case of F2 / / L_{max} and present an algorithm for the general problem with a worst case bound. The authors first prove that the EDD sequence is an optimal schedule if for i and j : $d_i \leq d_j$ if and only if min $\{a_i, b_j\} \leq$ min $\{a_j, b_i\}$.

This says that the EDD and Johnson sequence are the same. However, this property does not provide any precedence for the general case, but the authors schedule the jobs according to EDD and give an error bound that is tight asymptotically.

Grabowski, Skubalska, and Smutnicki (1983) the flow shop problem of minimizing maximum lateness with release dates. They analyze two lower bounds, one with one bottleneck machine and a non-bottleneck preceding it and the other with two bottleneck machines and a non-bottleneck between them.

2.4.4 Number of Tardy Jobs

Hariri and Potts (1989) consider a permutation flow shop and develop a lower bound and a branch-and-bound procedure to minimize the number of tardy jobs. This problem is NP-hard. For a given partial sequence and a machine j, we can consider the time this machine is available, disregard the processing before machine j and suppose that the machines after machine j have infinite capacity. The additional processing can be subtracted from the due date and The Moore-Hodgson algorithm applied to find a minimum number of late jobs. Applying this procedure to each of the machines yields a lower bound.

The authors derive another bound using the consistent early set, a set of jobs that is feasibly early for each of the machine subproblems. The size of the smallest consistent late set (a set whose complement is a consistent early set) is the lower bound. The authors extend this to the idea of a consistent feasible early sequence for single-machine and *r*-machine problems.

In generating problems to test their bounds, the authors set due dates that were uniformly distributed in a range proportional to an estimate of the total processing time. The fractions ranged from 0.2 to 1.0. The branch-and-bound algorithm with the simple bound was good enough for the small problems, but the use of consistent sequences yielded the most efficient procedure on the larger problems. The due date range of 0.4 to 0.6 yielded the hardest problems since the range was not great enough to guide sequencing and the due dates were tight.

2.4.5 General Topics

Garey, Johnson, and Sethi (1976) prove a number of NP-completeness proofs for scheduling problems. They do allow zero processing times. The authors begin by discussing some of the terms of NP-theory. All of the proofs will be by transformation from 3-partition, which is NP-complete in the strong sense.

The first proof is that of the F3 / / C_{max} problem, where dummy jobs are created to give slots on the second machine that jobs corresponding the elements of A must fit. The next proof is for the F2 / / ΣC_j problem. Dummy jobs create slots on the second machine that dummy jobs of intermediate length and the jobs corresponding to the partition problem must fill. Long tasks are added to the end to insure that the spacers are completed as soon as possible. Lastly, they prove that J2 / / C_{max} is NP-complete, although this depends upon a job that is re-entrant n times to machines 1 and 2.

Gonzalez and Sahni (1978) consider a number of flow shop and job shop problems, with and without preemption, minimizing makespan and flowtime. The authors extend the makespan results to the preemptive problems.

F3/pmtn/C_{max} is NP-complete even if no job has more than 2 operations. They have jobs with 1-2, 1-3, and 2-3 flows. They transform Partition into this, and the result and proof are the same for non-preemptive jobs. F3/pmtn/C_{max} is strongly NP-complete, including jobs with operations on all three machines.

J2/pmtn/C_{max} is NP-complete if all but two jobs have only two operations. The two other jobs go 1-2-1 and 2-1-2. The result also holds for non-preemptive jobs. J2/pmtn/C_{max} is strongly NP-complete with one job that visits each of the two machines n times.

The authors move to some approximation algorithms and bounds. For the total flowtime job shop problem, the bound on the ratio of the flowtimes for any busy schedule to the optimal is m. And this holds if the busy schedule is SPT. The ratio for makespan is also bounded by m. A variation of Johnson's algorithm that considers pairs of machines has an error bound of $m / 2$.

Lageweg, Lenstra, and Rinnooy Kan (1978) report on a number of different lower bounds and a classification of them as well as showing that a new lower bound for the makespan problem is better than the rest.

They present two dominance criteria and then move to the lower bounds, which relax the problem by allowing some machines to have infinite capacity. The unit-capacity machines are called bottlenecks and the infinite-capacity one are non-bottlenecks. The authors limit consideration to at most two bottleneck machines M_u and M_v and combine consecutive non-bottlenecks into one by summing the processing times for each job. Additionally, a non-bottleneck can be eliminated by adding the minimum processing requirement on this machine to the lower bound. This leads to nine different schemes (excluding symmetric ones). The authors describe each of the nine and analyze the effort required for each. They describe some upper bound calculations.

They test all of the lower bounds on problems with six jobs and three, five, or eight machines. The better ones were tested on larger problems with 10 to 50 jobs and 3 to 5 machines. The best results were obtained using elimination criteria and the two bottleneck machine bound with a non-bottleneck between them and the head and tail non-bottleneck machines removed.

Some researchers have looked at the flexible flowshop problem, where there may exist two or more parallel machines at a stage of the flow. Wittrock (1988) attempts to minimize the makespan and the queueing time. He creates three subproblems: machine allocation, sequencing, and timing. In order to simultaneously minimize makespan and queueing, he develops a heuristic procedure to greedily sequence the jobs given an allocation of the jobs to machines. He applies the LPT heuristic to do this allocation at each stage. Timing consists of loading the parts into the system as late as possible while not delaying any subsequent operation. He also considers buffer limits.

Sriskandarajah and Sethi (1989) consider a number of simple heuristics for the flexible flowshop problem. They derive worst-case error bounds for list scheduling and a Johnson-like sequence on the two-stage case where the first stage has only one machine. Their best relative

Also, Blackstone, Phillips, and Hogg (1982) report on a heuristic that is called look-ahead but would be classified here as look-behind, and Savell, Perez, and Koh (1989) did develop a look-behind system.

More specifically, look-ahead models consider the states of the machines where the jobs will be processed after being processed on the machine being scheduled. Information about the workload of other machines may be useful in balancing the flow of product through the shop. Look-behind models consider the machines the precede the machine being scheduled in the flow and the jobs being processed on these machines. Some of these machines will be processing jobs that will next need processing at the machine being scheduled. If the times that these jobs will complete is known, these times form release dates to the machine being scheduled, and the scheduling decision can explicitly consider these imminent arrivals.

Other researchers have studied procedures that they called look-ahead scheduling, but the problem setting or interpretation is slightly different.

Vepsalainen and Morton (1987) called their weighted COVERT and apparent tardiness cost rules "look-ahead" since they are concerned with the remaining waiting time of a job. They use, however, average waiting times without looking at the current queues.

Koulamas and Smith (1988) are concerned with the scheduling of jobs on machines that are attended by a server that must unload a job from a machine and load another job onto the machine. Interference results from one machine finishing while the server is busy at another machine. This interference degrades the system performance by preventing the machine from being maximally utilized.

The authors study a two-machine system where each machine has a distinct set of job types arriving to it. The authors propose a look-ahead rule for scheduling a machine that considers the state of the other machine when deciding what job to sequence next. The rule attempts to schedule a job whose completion will not interfere with the completion of the job on the other machine.

This work is similar to the definition of look-ahead in this paper, although its objective (to minimize interference) is not related to the completion-time objectives being studied.

Zeestraten (1990) is concerned with minimizing makespan in a job shop with routing flexibility. That is, some operations may be able to choose from more than one machine for their processing. The author defines a look-ahead rule as one that considers the entire state of the system and all of the unscheduled operations and creates a partial schedule that specifies a few operations for each machine. This schedule is followed until another scheduling decision is made. At this point, the procedure is repeated, using the new information about state of the system.

The look-ahead rule is so called because it searches through the states that the system could reach in a period that is approximately twice the average cycle time. That is, it looks ahead in time without attempting to schedule all of the remaining operations. Thus, it falls somewhere between fixed schedules and real-time dispatching.

This type of scheduling is actually more global in nature, since it considers the entire shop when scheduling. It is not looking at specific machines.

In summary, although scattered look-ahead and look-behind techniques have been considered, this research attempts to categorize these ideas, conduct analysis of look-ahead and look-behind scheduling problems in order to derive good rules, and integrate the results into a job shop scheduling environment.

2.6 Class Scheduling

Manufacturing often involves machines that process different product types, and this phenomenon can be modelled as a class scheduling problem, a topic mentioned in the introduction (Section 1.4). A number of researchers have studied the class scheduling case of sequence-dependent setup times. Examples reported by Monma and Potts (1989) include paint production machines that are cleaned between the production of different colors, a computer

system that must load the appropriate compiler for a type of task, and a limited labor force with workers switching between two or more machines.

Bruno and Downey (1978) study class scheduling problems with deadlines. Their problem is the single-machine scheduling of classes of tasks with deadlines and a setup time or changeover cost between classes. They prove that, with setup times, the question of finding a feasible schedule is NP-complete. With changeover costs, the problem of finding a minimal cost feasible schedule is also NP-complete.

Monma and Potts (1989) present complexity results for class scheduling problems without deadlines. The assume that the class setups satisfy the triangle property. They show that minimizing makespan, maximum lateness, the number of tardy jobs, and the unweighted and weighted flowtime is NP-complete, although they discuss a number of optimal properties and dynamic programming algorithms that are polynomial in number of jobs but exponential in the number of batches. They also show that the corresponding parallel machine problems are NP-complete. The authors conclude that the design and analysis of heuristics for these problems is important.

Ahn and Hyun (1990) study the problem of minimizing flowtime and present a dynamic programming algorithm that is similar to those of Monma and Potts. This algorithm is exponential in the number of classes, and the authors develop an iterative improvement heuristic that finds near-optimal schedules.

Sahney (1972) considers the problem of scheduling one worker to operate two machines in order to minimize the flowtime of jobs that need processing on one of the two machines. Sahney derives a number of optimal properties and uses these to derive an intuitive branch-and-bound algorithm for the problem.

Gupta (1984) also studies the two-class scheduling problem and derives an $O(n \log n)$ algorithm to minimize flowtime. Potts (1991) presents an example that Gupta's algorithm does not solve and goes on to describe an $O(n^2)$ dynamic program to minimize flowtime and an $O(n^3)$ algorithm to minimize weighted flowtime.

Coffman, Nozari, and Yannakakis (1989) consider a class scheduling problem with two subassembly part types. A product consists of two parts, one of each type, both made on the same machine, and the product cannot be delivered until both parts are finished. Their objective is to minimize the flowtime of delivered products, and using properties of an optimal schedules, they authors create an $O(\sqrt{n})$ algorithm. The authors describe extensions to multiple copies, limits on the number of changeovers, and limits on batch size as solvable cases without details.

Mason and Anderson (1991) study the one-machine class scheduling problem with the objective of minimizing weighted completion times under sequence-independent setups that fulfill the triangle inequality. By using properties of an optimal sequence and aggregating jobs into composite jobs, the authors develop a brand-and-bound algorithm using their dominance criteria and lower bound.

Dobson, Karmarkar, and Rummel (1987) consider a class scheduling problem with the objective of minimizing flowtime under both the item-flow and batch-flow delivery schedules. The batch (or class) setups are sequence-independent and the processing times are equal for all parts (jobs) in one part type (class). They formulate integer programs for both problems, solving the item-flow problem and single-product batch-flow problems optimally. For the multiple-product batch flow, the authors can only provide some heuristics based on their other results.

Dobson, Karmarkar, and Rummel (1989) consider the above single-item problems on uniform parallel machines. They assume that the work is continuously divisible and determine the amount of work to allocate to each machine by solving a convex programming problem. The authors also apply their results to solve the item-flow problem.

Woodruff and Spearman (1992) consider an interesting class scheduling problem where the objective is to maximize the profit of feasible schedules. That is, jobs must be completed by their deadlines. The profit calculation includes the value of jobs that are not required to be processed but add some revenue and two different costs: holding and setup. The authors search for good solutions with a tabu search. Discussion of this search can be found in Section 2.8.3.

A more typical problem is studied by Ho (1992), who examines the problem of minimizing the number of tardy jobs where the jobs fall into exactly two classes. He develops and efficient branch-and-bound algorithm.

Gupta (1988) studies the class scheduling problem of minimizing the total flowtime. He develops a heuristic that is based upon the standard shortest processing time rule. Empirical testing shows that the heuristic produces good results on small problems.

A more sophisticated class scheduling problem is the minimization of the maximum lateness of a set of jobs with non-zero release dates. This problem is considered by Schutten and Zijm (1993), who develop a branch-and-bound algorithm and a tabu search over the sequences of jobs. They report good results on problems with up to 50 jobs.

In summary, this body of research does not yet include much work on problems with additional criteria, such as deadlines or release dates, or on shop problems. Moreover, this research has concentrated on branch-and-bound techniques and optimal algorithms for problems with just two classes. Little work has been done on heuristics for problems with an arbitrary number of job classes or on the use of smart-and-lucky searches to solve class scheduling problems.

2.7 Some One-machine Problems

In order to gain insight into new dispatching rules that may be useful in the job shop scheduling question, some one-machine problems will be investigated. The three one-machine problems reviewed in this section are the problem of minimizing total flowtime with deadline constraints, the problem of minimizing the number of tardy jobs subject to matching release dates, and the problem of minimizing the total flowtime of jobs with release dates. These questions will be the first to be studied as class-scheduling problems.

2.7.1 Constrained Flowtime

The first one machine problem studied in the preliminary work is a class scheduling extension of $1 / D_j / \Sigma C_j$, the constrained flowtime problem first studied by Smith (1956).

Smith, in this paper, proves the optimality of ordering the jobs by Shortest Processing Time to minimize the total flowtime problem $1 // \Sigma C_j$. He extends this to the problem where the jobs all have deadlines that must be met.

The makespan of the jobs is known, and following the SPT property of minimizing total flowtime, the algorithm tries to schedule the longest job to end at this time. However, the longest job's deadline may be less than the makespan, in which case the job is not available to end at this time. So, the algorithm searches for the longest job that can end at this time and schedules it. The algorithm then moves to the start time of this scheduled job. This time is now the completion time of the remaining unscheduled jobs, and the algorithm again searches for the longest available job. This algorithm is hereafter referred to as Smith's algorithm.

The weighted problem is strongly NP-complete (Lenstra, Rinnooy Kan, and Brucker, 1977), and different elimination criteria and branch-and-bound techniques have appeared. Potts and van Wassenhove (1983) study the Lagrangean relaxation of the deadline constraints, leading to the discovery of optimal solutions. Work by Posner (1985) and Bagchi and Ahmadi (1987) present improved varieties of this bound. Many other problems have been studied; see Herrmann, Lee, and Snowdon (1993) for a survey of dual criteria problems.

2.7.2 Release and Due Dates

The other one-machine problem in the preliminary results is the look-behind problem $1 / r_j / \Sigma U_j$, which is a strongly NP-complete problem in general (Lawler, 1982). The problem has been solved optimally if all release dates are zero by the Moore-Hodgson algorithm (Moore, 1968). The problem has been solved optimally by Kise, Ibaraki, and Mine (1978) if the release

and due dates match ($r_j < r_k$ implies $d_j \leq d_k$). They present an $O(n^2)$ algorithm (Kise's algorithm, described below) in this case.

Kise's algorithm orders the jobs by their release and due dates (a non-ambiguous ordering since the dates must match). The algorithm is an extension of Moore-Hodgson algorithm for minimizing the number of late jobs. Each job is scheduled after the partial schedule of on-time jobs while maintaining release availability. If the new job is tardy, the algorithm searches the on-time jobs for the job whose removal leaves the shortest schedule of on-time jobs. The removed job is made tardy and will be processed with the other tardy jobs after the on-time jobs. In this manner, the algorithm finds the largest subset of the jobs that can be delivered on-time. These jobs are scheduled in order of their release and due dates.

Unlike the Moore-Hodgson algorithm, the subalgorithm that removes a job cannot just choose the longest job, since the presence of release dates limits how the removal a single job affects the completion times of later tasks. They present an efficient way to determine which job should be removed.

2.7.3 Flowtime and Release Dates

A last one-machine problem that may yield some good results when studied as a class scheduling problem is closely related to the constrained flowtime problem. This is the problem of minimizing total flowtime subject to job release dates. Written as $1 / r_j / \Sigma C_j$, this problem is a strongly NP-complete question, as shown by Lenstra, Rinnooy Kan, and Brucker (1977).

Dessouky and Deogun (1981) present a branch-and-bound algorithm with a lower bound and dominance properties used to prune the search tree. They list a number of dominance properties that hold for a given partial sequence of the jobs. Their lower bound is derived from the EFT schedule but starts the selected job at the first release time. They can solve problems with up to 50 jobs. They also report that the EFT sequence generally finds good solutions (within 3% of optimal).

Bianco and Ricciardelli (1982) consider the weighted version of the problem. Let the weighted processing time be the processing time divided by the weight of the job. They present an improved branch-and-bound algorithm with further dominance properties for nodes with a partial schedule. They compute a lower bound by allowing preemption. The hardest 10-job problems to solve were those with a larger range of weights and a maximum release date near the expected sum of processing times (that is, the average interarrival and processing times were nearly equal).

Hariri and Potts (1983) also attack the problem of minimizing weighted flowtime with release dates. They use a Lagrangean relaxation of the constraints $C_j \geq r_j + p_j$ to find a lower bound. They make use of previous dominance properties and add another that measures the effect of interchanging two consecutive jobs. They solve problems with 10 to 50 jobs, and the hardest problems (some of which remain unsolved) were those with a range of release dates approximately the same as the expected sum of processing times.

Gazmuri (1985) studies the question probabilistically. He develops two cases. In the undersaturated case, where the expected processing time is strictly less than the expected time between release dates, the author develops an algorithm that partitions the jobs into smaller sets and schedules each of the sets optimally. The algorithm for the oversaturated case starts with the optimal preemptive schedule and patches preempted jobs by shifting the delayed segments to the left until the job is whole. For both cases, the heuristic is asymptotically optimal as the number of jobs goes to infinity.

Liu and MacCarthy (1991) use both mixed-integer linear programming and branch-and-bound techniques to solve the problem. They present a number of heuristics that are different priority rules and report that the heuristics generally find near-optimal solutions (within 1%) with little computational effort until the problem sizes exceed 100 jobs. The MILP can solve problems with only 10 jobs, the branch-and-bound procedure problems with 25 jobs.

Finally, Rinaldi and Sassano (1977) also report on a branch-and-bound technique for the weighted problem.

2.8 Smart-and-lucky Searches

In this section we will discuss a class of searches that have been receiving much attention from researchers recently. This class include tabu search, simulated annealing, and genetic algorithms. We will use the last of these in the research into different scheduling problems.

2.8.1 Introduction

One approach to difficult scheduling problems such as job shop scheduling is local search, an iterative procedure that moves from solution to solution until it finds a local optimum. Two examples are hillclimbing and steepest descent. The hillclimbing algorithm chooses a nearby point at random and moves there if the point improves the value of the objective function. The steepest descent procedure examines the entire neighborhood of an incumbent point and selects the point (if any exist) that is most improving.

Heuristic searches (or probabilistic search heuristics) attempt to improve upon the primary problem of these simple searches: convergence to local optima. We can use the term *smart-and-lucky* to describe these more complex searches: they are smart enough to escape from most local optima; they still must be lucky, however, in order to find the global optimum. Simulated annealing, the most popular of the methods described in this work, was developed independently by Kirkpatrick, Gelatt, and Vecchi (1983) and Cerny (1985). Glover claims that tabu search, which is also widely used, goes back to a paper of his from 1977. Holland developed the ideas of genetic algorithms, which have had the least success so far, in his 1975 book. This section reviews the basic concepts of these searches and a number of scheduling applications.

2.8.2 Simulated Annealing

Simulated annealing (SA) is a variant of the hill climbing algorithm. As mentioned before, both Kirkpatrick, Gelatt, and Vecchi (1983) and Cerny (1985) produced the initial papers on the

simulated annealing algorithm, so called because the algorithm views optimization as a process analogous to physical annealing, the cooling of a system until it reaches a low-energy state.

In annealing, as a system cools, configurations occur with weight dependent upon their energy and the system's temperature. In 1953 Metropolis *et al.* developed a statistical simulation of an annealing system. This algorithm randomly shifted the system from one configuration to another, using the physical laws that governed behavior to guide the procedure. Kirkpatrick, Gelatt, and Vecchi (and Cerny) modified his procedure to solve optimization problems. The crucial element was the acceptance of a non-improving move with a probability that was a function of the difference D in objective function and some cooling constant T. The standard formula was $e^{-D/T}$. This acceptance of a non-improving move is what differentiates a simulated annealing from hillclimbing.

Kirkpatrick, Gelatt, and Vecchi (1983) list four ingredients for a simulated annealing algorithm: a precise description of system (that is, a solution space), the random generation of moves, a quantitative objective function, and an annealing schedule. Simulated annealing makes a move by picking from a set of possible operations that can be applied to the incumbent solution. Usually, the choice is made randomly, although the set may be ordered somehow. At a given temperature, the algorithm may stop when some equilibrium or maximum number of moves is reached. Then the algorithm reduces the temperature geometrically or linearly. The initialization of the algorithm with a starting point may be random or may use a heuristic.

It can be proved that simulated annealing will converge to a global optimum. Aarts and van Laarhoven (1985) use a homogeneous Markov chain that reaches equilibrium at each temperature to prove this. They use a general form of acceptance probabilities, of which the standard exponential is a special case. However, reaching equilibrium at a temperature may require an exponential number of moves. They avoid the exponential chain length by defining quasiequilibrium conditions and conclude with a polynomial algorithm.

Kirkpatrick, Gelatt, and Vecchi (1983) studies a number of chip design problems and the traveling salesman problem. In his example, he places the cities in nine clumps and examines the

effect of temperature on the solutions found. At high temperature, the algorithm minimizes the number of long jumps; at a medium temperature, the coarse structure changes but stays minimal; and at low temperatures, the procedure improves the solutions at the small scale. Cerny also studies the traveling salesman problem.

Van Laarhoven, Aarts, and Lenstra (1988) study the job shop scheduling problem by making use of the disjunctive graph, where the makespan of a solution is the length of the longest path. The disjunctive graph has an arc for each precedence among operations for a job and an (undirected) edge connecting those operations on different jobs that use the same machine. A schedule can be found by directing the edges to form an acyclic graph. Given a feasible solution, any swap of a critical disjunctive edge will form another feasible solution. This becomes their move, and they claim that their simulated annealing is as good as shifting bottleneck and simpler.

A different type of simulated annealing is included in the Matsuo, Suh, and Sullivan papers (1988, 1989) on controlled search simulated annealing (CSSA). In this approach, the algorithm uses a good initialization, independent acceptance probabilities, a low initial acceptance probability, and a sequential search of smaller neighborhoods. The motivation for acceptance probabilities independent of the change in objective is its use in empirical testing of different cooling schedules. Their first paper addresses the single-machine weighted tardiness problem. The CSSA uses a fixed number of iterations and its move swaps an adjacent pair. The authors compare different cooling schedules and claim that the CSSA is a better procedure than standard SA.

In the paper on job shop scheduling, the authors use the shifting bottleneck algorithm for initialization. The CSSA identifies the critical path and then switches two operations on the critical path with look-back and look-ahead options that swap other operations to improve makespan. If a non-improving move is not accepted, they perform a local search to find a possible improving solution. They claim better makespans with the same effort as shifting bottleneck.

For flow shops, Ogbu and Smith (1990) present a simulated annealing that they call "probabilistic-exhaustive" because it uses independent acceptance probabilities with a random search of the entire neighborhood, moving to the last point accepted. They minimize makespan, and their moves are pair swaps and insertions. They get better processing time than the standard SA and better results than repetitive local searches. Vakharia and Chang (1990) look at a flow shop with jobs in families and family setup times. They also use independent acceptance probabilities. Their moves swap jobs within a family or swap families. They use different initializations and compare their search to other heuristics.

2.8.3 Tabu Search

Tabu search (TS) is a variant of steepest descent. Glover (1989, 1990) presents a good discussion of tabu searches. Given an incumbent solution, a TS searches the neighborhood of this solution, finding the best allowable move. A TS allows bad move away from local optima, prohibits moves which lead backwards through short-term memory (the tabu list) and has an aspiration level to override the tabu list under certain conditions (usually best found). A tabu search works because the tabu list forces the search to explore new areas of the solution space. The short-term aspect of the memory and the aspiration level allow the search to get to a global optimum however.

Table 2.1. Outline of a Simple Tabu Search.

1. Pick initial x. let $x'= x$. T, the tabu list, is empty.
2. Let $S(x)$ be the neighborhood of x. Take s' as best member of $S(x) \setminus T$.
3. Let $x = s'(x)$. If $c(x) < c(x')$, $x'= x$.
4. Check stopping criteria. Update T. Go to 2.

The best member is usually that which gives the best value of the objective function $c(x)$. The stopping criteria may be a total number of moves or a number of moves since the last best.

Updating the tabu list T is the crucial part of a tabu search. Usually this means adding moves that would reverse the step just taken and removing old tabu moves.

Laguna, Barnes, and Glover (1989) investigate a tabu search for a single-machine scheduling problem with linear delay penalties and setup cost dependencies. They use a heuristic initialization and insertion and swap moves. They claim results much better than a branch-and-bound.

Barnes and Chambers (1991) investigate the use of a tabu search to solve the job shop scheduling problem of minimizing makespan. The authors describe an improved approach that uses an actual makespan change when evaluating new schedules instead of a projected change.

The authors compare their job shop scheduling approach to Applegate and Cook's shuffle algorithm and the Shifting Bottleneck algorithm of Adams, Balas, and Zawack. They observe that their tabu search achieves better makespans in most cases over a range of problem sizes.

Barnes and Laguna (1992) examine the multiple-machine weighted flowtime problem with a similar tabu search. This problem reduces to a partition problem, and their search does prevent non-improving swaps: they claim that swaps are too small to move through the solution space effectively. Again, they claim results that beat a branch and bound approach.

Widmer and Hertz (1989) take up the flow shop and use permutation schedules. They define a distance between two jobs as the approximate increase in makespan and perform a TS upon the open TSP. A move in the solution space is the swap of a pair. They get slightly better makespans than six heuristics at the cost of terrible processing times.

An ambitious paper by Malek *et al.* (1989) examines parallel and serial tabu searches and simulated annealing on the traveling salesman problem. For each search, a move was a 2-opt (subsequence reversal). The parallel runs communicate periodically, sharing good solutions. Parallel SA performed a quick annealing on each processor and achieved superlinear speedup. According to the authors, parallel TS performed best, but the simulated annealing was more robust (less sensitive to parameter changes).

In an interesting article, Woodruff and Spearman (1992) study the problem of maximizing the profit gained from scheduling a set of jobs on one machine. The authors claim that the one-machine problem has significance for shops with a bottleneck operation, where the sequencing of the bottleneck determines how the shop performs. These jobs include jobs that are required in some sense, say to meet specific orders, and jobs that are filler, in that they are not required now, but some profit is realized by producing them. The jobs have due dates, and the schedule should be feasible with respect to these. The costs include the holding costs of finishing jobs too early and setup costs incurred when the machine must be switched from processing jobs of one family to those of another. Thus, this problem is a class scheduling problem, and it is an NP-complete question just to ask if there exists a feasible schedule.

The authors show that if the holding costs are zero, it is optimal to order the jobs in each family by EDD. If the holding costs are positive, this EDD within a family is still used as a heuristic in order to simplify the problem of finding a good schedule.

The authors use a tabu search to search the solution space for good sequences. The moves of the tabu search are insertions of jobs while maintaining EDD within a family. Complications are added by the presence of filler jobs that are not required to be in the schedule. The authors use a diversification parameter for two reasons: first, the parameter diversifies the search by being included in a modified cost function that provides a penalty for infeasible solutions while allowing the search to use these as passes into new areas. Secondly, this parameter allows the search to optimize its performance. This is done by initially performing tabu searches over a number of different values of the diversification parameter and then continuing the search with the best values.

Empirical testing of the algorithm on data motivated by an actual manufacturing environment showed that the search is a useful method of finding good solutions for this problem.

Glover, Taillard, and de Werra (1991) describe the main aspects of tabu search and discuss various refinements that may lead to a new generation of search. These refinements occur on the tactical, technical, and computational levels. Tactical improvements are concerned with

improving the neighborhood structure and defined moves. This may include the use of learning approaches, shifting penalty tactics, and strategic oscillation.

Technical improvements focus on the issues of neighborhood size, tabu list structures, and aspiration conditions. Computational improvements include reducing the time required to compute the objective function over the neighborhood and the parallelization of the search.

Laguna and Glover (1991) use target analysis to promote diversification in a tabu search used to find good solutions to a single machine problem with linear delay and setup costs. The tabu search uses both swap and insert moves.

The goal of using target analysis is to teach the tabu search to use good rules. The authors describe five phases of target analysis. The first phase is to take some specific problems and find very good solutions to those problems. In the second phase, with these very good solutions as targets, the problems are solved again and information is gathered on how well the current decision rules lead to the target. In the third phase, this information is integrated into a master decision rule. The fourth phase finds good parameter values for this master. Finally, the master is applied to the original problems to verify its merit.

Reeves (1993) examines how the definition of the neighborhood in a tabu search affects the balance between exploration and exploitation (or diversification and intensification). Given the proper balance, he finds that tabu search is a more efficient procedure than simulated annealing for the permutation flowshop problem.

2.8.4 Genetic Algorithms

A genetic algorithm is a smart-and-lucky search that manipulates a population of points in the effort to find the optimal solution. Each individual in the population is a string of genes, where each gene describes some feature of the solution. Genetic algorithms mimic the processes of natural evolution, including reproduction and mutation. The most powerful operator of a genetic algorithm is the crossover operator: the recombination of the genes of two parents to

create two offspring. This crossover allows the offspring to acquire the good characteristics of different points in the search space.

Most genetic algorithms perform the following steps, stopping when a fixed number of offspring has been created:

Step 0: Form an initial population.
Step 1: Evaluate the individuals in the population.
Step 2: Select individuals to become parents with probability based on their fitness.
Step 3: For each pair of parents, perform a crossover to form two offspring. Mutate each offspring with some small probability.
Step 4: Place the offspring into the population. Return to Step 1.

Holland (1975), Davis (1987, 1991), and Goldberg (1989) provide good descriptions of genetic algorithms. Holland's 1975 book introduced the genetic algorithm (GA), a procedure that mimics the adaptation that nature uses to find an optimal state. In genetic algorithms, solutions are represented as strings (chromosomes) of alleles, and the search performs operations on the population of solutions. Liepins and Hilliard (1989) identify these operations as 1) the evaluation of individual fitness, 2) the formation of a gene pool, and 3) the recombination and mutation of genes to form a new population. After a period of time, good strings dominate the population, providing an optimal (or near-optimal) solution.

The solution strings may be a sequence of binary bits or a permutation. The fitness of a solution is its objective function evaluation or ranking. In forming the gene pool, the algorithm takes, through some random process, those solutions that are more fit. The recombination is some type of crossover, one-point or multipoint. It is this powerful crossover mechanism that makes GAs special. The mutation operation, so important in SA, is a secondary operation here and serves only to maintain diversity.

Why do GAs work? A schemata is a pattern possessed by individuals in population. The fitness of a schema is the average fitness of those solutions that possess it. Genetic algorithms work because good (short) schema survive exponentially, and the population of solutions provides implicit parallelism.

When do they fail? A genetic algorithm may fail if the strings are inappropriate representations of solutions, the strings exclude important problem information (such as constraints), or the algorithm converges to a local optimum. The first issue is the largest for scheduling problems: representation is difficult for combinatorial problems because standard crossovers may lead to infeasibility.

In an effort to address this problem for the traveling salesman problem, where a solution is a permutation of cities, Oliver, Smith, and Holland (1987) propose a number of unusual crossovers. The first is the Order crossover, which substitutes a subsequence from the first parent into the second and then visits the remaining cities in the same relative order as before. The crossover maintains short schema. Their PMX crossover compares subsequences from both parents and then performs a swapping in the second parent. The Cycle crossover matches cities to form independent tours. Then, for each tour, the crossover picks a parent to supply the cities. After comparing the tree crossovers, they claim that the Order is better on TSP by preserving the short schema, which are important for this problem.

The job shop scheduling problem is much more complicated problem. Davis (1985) solves a simple job shop problem with a GA. Another method is an idea by Storer, Wu, and Vaccari (1990, 1992): searching problem spaces and heuristic spaces, which are discussed in detail in Section 2.9.

Storer, Wu, and Vaccari (1990) note that a solution is simply the application of a heuristic to a problem. Changing the problem or the heuristic generates a new schedule and thus a new solution. A problem is a vector of processing times, and a heuristic is a vector of dispatching rules that can be used to create a non-delay or active schedule. These data structures create simple strings and perform well under standard crossover as well as simulated annealing and tabu search. The authors perform all three searches on both types of spaces. For the tabu search, the tabu list was a set of tabu makespans. Otherwise, the SA and TS were standard. The authors also tried shifting bottleneck, probabilistic dispatching, and random search over their new spaces. They test their heuristics on a number of problems ranging in size from 10 to 50 jobs. The

problems were also classified as easy or hard. The easy problems had completely random routings, and the hard problems tended have greater competition for the machine resources at any point in time. The genetic algorithms on problem space generally found higher-quality solutions.

Fox and McMahon (1990) are interested in the job shop scheduling problem but study the traveling salesman problem in order to gain insight into using genetic algorithms for sequencing problems.

The authors consider the binary precedence matrix M where $m_{ij} = 1$ if city i precedes city j. $m_{ii} = 0$ for all cities i. This matrix incorporates micro- and macro-information. The authors consider row and column swap operators that exchange the successors or predecessors of two cities.

The first new operator is the intersection operator, in which an offspring inherits the precedences that exist in both parents. That is $c_{ij} = a_{ij}$ and b_{ij}, over all i,j. This will yield a matrix that is consistent (no cycles) but incomplete; in other words, a partial ordering. The matrix is completed through an analysis of the row and column sums.

The second new operator is the union operator. First, the cities are partitioned into two sets. This forms for each parent two precedence submatrices corresponding to the two subsets. The operator then takes one submatrix from one parent and the submatrix for the other subset from the other parent. Again, this leads to a partial ordering that must be completed. The authors claim that this operator contains little micro-information from the parents, unless a Markov process is used to partition the cities.

The authors compare different operators for the TSP, including a random operator used as a benchmark. If the proposed operators are no better than this, they should not be pursued. Their city topologies included random distances, clustering, concentric circles, and a 30-city problem from Oliver, Smith, and Holland (1987). The population size was 900 with no mutation. The also tried searches with and without elitism (the keeping of the best solution).

Among non-elite searches, the two new operators were great, even when considering processing time. Their advantage disappeared when the searches got to use elitism.

The authors claim that this work will be useful in scheduling problems, although execution time remains a big problem.

Biegel and Davern (1990) begin with a general discussion of genetic algorithms. They then describe how a genetic algorithm could play a part in the scheduling of a shop. They discuss what data are essential and how the GA could form a schedule for a known group of jobs that follow the same route.

The authors move to the single-machine problem and consider how the analogous GA would work. Moving to the 2- and *m*-machine flow shop, the authors present a simple discussion of what the GA would do.

For the dynamic problem, the authors consider performing a rescheduling in real-time to deal with arrivals and use a FIFO dispatching rules on all other machines. The authors then list a number of areas for future research.

Cleveland and Smith (1989) use a genetic algorithm to schedule the release of jobs into a manufacturing cell to minimize weighted tardiness. They consider both the sequencing problem and the problem of determining the release times.

Nakano and Yamada (1991) introduce a binary representation to solve the job shop scheduling problem. This representation denotes the relative ordering of each pair of jobs on each machine. If an illegal chromosome is formed by a genetic operator, a nearby legal one replaces it. The authors report that their algorithm finds good solutions on some standard problems.

Whitley, Starkweather, and Shaner (1991) develop an edge recombination operator for the traveling salesman problem and report that a genetic algorithm with this operator is able to find very good schedules.

Starkweather *et al.* (1991) compare six sequencing operators for the traveling salesman problem. They discover that the different operators stress different types of information; since the TSP depends upon adjacency, the edge recombination operator is the best.

In Syswerda (1991), the author discusses the scheduling of a fighter simulation lab. Finding a feasible scheduling is made difficult by a number of constraints. A genetic algorithm that manipulates sequences of jobs and employs a schedule builder to translate the sequence into a legal schedule was able to find good schedules.

2.8.5 Summary

Simulated annealing and tabu search easily search complex spaces by lending themselves to usually simple types of moves, generally find good solutions fast, and are smart variations of standard searches such as hillclimbing and steepest descent. Genetic algorithms work in a different manner. They work well due to the survival of good schema and their implicit parallelism. They have been harder to implement on scheduling problems, however.

This research in this dissertation builds upon this work into smart-and-lucky searches by looking into some alternative search spaces.

2.9 Problem and Heuristic Space

This dissertation includes the development of global job shop scheduling models, whose solution will yield a schedule that can be followed for time period like an eight-hour shift. As mentioned in the last section, one way to find good schedules is to use a search. One recently-proposed idea is to search problem and heuristic spaces. This section will define mathematically how these spaces can be used and will review the ideas of a few papers in this area.

Define a *problem p* as a set of data about which an optimization question can be asked. A *solution s* is a point that is consistent with the problem structure, where the solution has some performance z measured by applying the objective function f to the solution with the problem data, $z = f(p,s)$. If the problem is fixed, there is a performance function C_p over the solution space such that $z = C_p(s) = f(p,s)$. Solving a problem translates as finding the solution that gives the minimum function value.

Traditionally, problem-solving searches have occurred in the solution space. In the solution space live points (for standard optimization) or sequences (for combinatorial optimization). Heuristics are also applied to problems. A *heuristic h*, applied to a problem *p*, yields a solution *s*, $s = G(h,p)$ or $s = h(p)$. Thus, for a given problem *p*, there exists a performance function $D_p(h)$ over the heuristic space such that $z = D_p(h) = C_p(h(p))$. This implies that a search over different heuristics could find a solution that gives the optimal objective function value.

Moreover, a solution can be generated in ways besides applying a heuristic to the original problem. Applying a heuristic *h* to a problem *y* in another space can generate a solution $s = h(y)$, which can be evaluated using the objective function. Thus, given a problem *p* and the heuristic *h*, there exists a performance function $E_{p,h}(y)$ over the other problem space, where $z = E_{p,h}(y) = C_p(h(y))$. And as before, a search over this new problem space provides a way to solve the problem.

The idea of searching heuristic and problems spaces was investigated by Storer, Wu, and Vaccari (1990, 1992). In these papers, the authors are investigating the general job shop scheduling problem. They define a heuristic space composed of vectors of dispatching rules. The heuristic uses each rule in turn for a fixed number of dispatching decisions. For example, if the problem is a ten-job by ten-machine problem, the vector might have five elements, where the dispatching rule in the first element is used for the first window of twenty decisions, the next rule for the next window, and so on, until all one hundred operations have been processed.

For a problem space, they use a space of vectors of processing times, where each position corresponds to a different operation. At each scheduling decision in the formation of the schedule, the dispatching rule selects a job by considering these alternative processing times. The operation of that job is scheduled using the actual job data in order to calculate the finish time of the operation, thus ensuring that the schedule produced will be feasible. Thus, a vector of processing times yields a sequence of operations for each machine.

The authors include a number of concepts related to searching these new spaces, including the definitions of neighborhoods, the use of these spaces in genetic algorithms, the fact that the spaces contain optimal solutions, and details of local search implementations.

The application to genetic algorithms merits special comment. Traditional genetic algorithms have difficulty with scheduling problems since the components of the strings (the jobs in the sequence) are not independent of each other. The problem space and heuristic space described above, however, consist of vectors that have independent elements. That is, the dispatching rule (or processing time) in the first element does not affect what values the other elements can have. Thus, a crossover operation that breaks two strings (vectors) and joins the separate pieces yields offspring that are valid points in the search space.

The authors extend this work in Wu, Storer, and Chang (1993) to the one-machine rescheduling problem. They use heuristics that adjust "artificial tails" to find schedules that are efficient and deviate little from the original schedule.

Bean (1992) uses the idea of random keys to provide an alternative search space for a number of problems: multiple-machine scheduling, resource allocation, and quadratic assignment: the keys are used to sequence the jobs (or other variables). Then some simple rule is used to generate a solution from this sequence. Good empirical results are reported.

Our research extends these ideas to one-machine class scheduling problems and investigates a similar heuristic space for the job shop scheduling problem.

2.10 NP-Completeness

Job shop scheduling problems are usually quite difficult to solve, and they are among the hardest optimization problems known. Combinatorial problems, as well as other types of problems, can be classified by their complexity, or how difficult they are to solve. The hardest problems are called NP-complete. The primary guide to the theory and use of NP-completeness is the book by Garey and Johnson (1979).

Most of the problems studied in operations research have been classified by their complexity, and as researchers study new problems, their complexity is identified also. There are three primary classifications: polynomial, NP-complete, and strongly NP-complete. Problems in the first set can be solved with algorithms that have a running time proportional to some polynomial function of the problem size. Algorithms to solve problems in the second set require effort that is a polynomial function of the problem size and the problem data. Such an algorithm is called a *pseudo-polynomial* algorithm. Algorithms to solve problems that are strongly NP-complete require an exponential amount of effort.

The complexity of a problem can be determined by identifying the problem as a more difficult case of a hard problem, by developing a polynomial algorithm to solve the problem, or by transforming a previously classified problem into the new problem. By being able to determine the complexity of a problem, researchers are able to know what types of approaches may be possible in finding optimal solutions to the problem. A problem that is known to be NP-complete will not be solved optimally by any polynomial-time algorithm. Pseudo-polynomial dynamic programs will not solve strongly NP-complete problems. This does leave the possibility, however, that certain special cases may be more easily solved, that algorithms that search for an optimal solution may have good average performance (e.g. the simplex method of solving linear programs), and that polynomial-time heuristic algorithms may be able to find generally high-quality solutions.

This research is concerned with a number of different scheduling problems, which will classified by their complexity as necessary. We will directly prove NP-completeness for one of the problems that we study. Each of the other problems is a more difficult case of a previously considered NP-complete problem.

2.11 Chapter Summary

This chapter has attempted to cover a broad spectrum of topics and literature. Obviously, much research has been done in the fields of job and flow shop scheduling and the field of

managing semiconductor manufacturing. However, no system to optimize the scheduling of such a process has emerged, despite the use of many different and sophisticated procedures.

Still, new tools are needed that may have good performance when applied to specific settings. There do exist new ideas, such as class scheduling, look-ahead and look-behind approaches, smart-and-lucky searches, and problem and heuristic spaces. The literature in these areas reports on some initial research.

However, the full capabilities of the these methods, especially when joined to solve a scheduling problem, have not been fully explored.