Chapter 3 – Literature Review

In this Chapter we review the literature of job shop dispatching and scheduling of semiconductor wafer fabrication. Several surveys of job shop dispatching have been previously published. The interested reader can refer to Sisson [Sis59a, Sis61a], Gliffer and Thompson [Gli60a], Mellor [Mel66a], Day and Hottenstein [Day70a], Panwalker and Iskander [Pan77a], and Blackstone *et al.* [Bla82a].

We first define the job shop scheduling problem and review theoretical results for the single machine scheduling problem. For all but the simplest versions, the job shop scheduling problem is NP-complete. We briefly review several attempts to obtain an exact solution and then focus on the main topic of this review – rules and heuristics for dispatching job shops. We number the usual assumptions made in the literature, survey simulation research of job shops, and then summarize results for simple and complex dispatching rules and dispatching heuristics. We review recent literature on rules that combine simple rules and use heuristic search to find a good combination. Several heuristic methods are surveyed and one in particular, pattern search, is covered in more detail. Lastly, we review the literature of scheduling integrated circuit manufacturing.

3.1 – Problem Definition

Lenstra and Rinnooy Kan [Len84a] define scheduling as the optimal allocation over time of scarce resources, in the form of *machines,* to activities, known as *jobs,* subject to the constraint that at any time no machine is allowed to process more than one job and no job can be processed by more than one machine. Several textbooks thoroughly cover scheduling theory. Among these, the reader is referred to Conway *et al.* [Con67a], Baker [Bak74a], French [Fre82a], and Hax and Candea [Hax84a].

Conway *et al.* [Con67a] propose notation to classify machine scheduling problems. Problems are categorized according to four parameters – $A/B/C/D$, where A

denotes the job arrival process, B is the number of machines, C denotes the flow pattern in the shop, and D is the performance criterion that is to be optimized. Graham *et al.* [Gra79a] suggest a somewhat different classification scheme. Among the array of production scheduling problems discussed in the literature [Gra81a], a large portion of real-world scheduling problems fall into the category of job shop scheduling. The general job shop consists of at least two machines. A job can enter the shop at any machine and is allowed to leave the shop from any machine. Each job has associated with it a sequence of operations to be carried out in the shop. Each operation must be performed by a specific machine and has a duration that can either be deterministic or obey a probability distribution. The sequence of operations may require a job to visit a machine one or more times.

Several assumptions regarding the job shop scheduling problem are made in the literature. These assumption vary from paper to paper. The following are compiled by Gere [Ger66a].

- Machines are capable of processing only one job at a time. Scheduling more than one job at a time can be viewed as multi-processor scheduling, for which there are several surveys — Coffman [Cof76a], Gonzalez [Gon77a], Graham *et al.* [Gra79a], and Lenstra and Rinnooy Kan [Len80a].

- Preemption, or suspension of a job before completion for another job to begin, is not allowed. Preemptive scheduling is surveyed in Coffman [Cof76a], Gonzalez [Gon77a], and Graham *et al.* [Gra79a].

- Once a job enters the shop, it can only leave after completion of all its operations.

- Even though jobs may be made up of physically distinct parts, no job can be processed by more than one machine at a time.

- Set up time is included in the job processing time and hence processing times are independent of the sequence in which the operations are performed. The effect of set up time on job shop scheduling has been investigated by Baker [Bak68a] and Wilbrecht and Prescott [Wil69a].
- Transportation time between machines is negligible when compared to processing time.
- Uncapacitated queues, or in-process inventory, are allowed to build up in front of machines.
- Machines are 100% reliable. Baker and Nuttle [Bak80a] consider the case where scheduled maintenance is allowed for single machine scheduling.
- Labor is not a constraining factor. Nelson [Nel67a] studies job shops in which labor is a constraining factor.
- Due-dates are fixed for each job as it enters the shop and do not change after that.
- A penalty is usually incurred when a job leaves the shop after its due-date.
- Alternate job routes are not allowed. Morse [Mor58a] studies the case where alternate routes are allowed.

Two classes of job shop scheduling problems are discussed in the literature. In the *static deterministic* problem, N jobs are given at time zero and must be feasibly scheduled to machines over time, so as to optimize some performance criterion. Most real job shops are dynamic probabilistic in nature. In the classical *dynamic probabilistic* job shop scheduling problem, N jobs are given at time zero, but continue arriving throughout the planning horizon with inter-arrival times usually obeying a probability distribution. An assumption not commonly referred to in the literature is that machines may fail and require non-deterministic repair time. As with the static deterministic case, jobs must be feasibly scheduled so as to optimize a given performance criterion. The criteria most used for measuring performance are —

- $F_{\max}$ — maximum flowtime, the longest time any job spends in the shop,
- $\bar{F}$ — mean flowtime,
- $W_{\max}$ — maximum waiting time, the longest time any job waits in the shop,
- $\bar{W}$ — mean waiting time,
- $C_{\max}$ — maximum completion time (or makespan), the time required to complete processing on all jobs,
- $\bar{C}$ — mean completion time,
- $L_{\max}$ — maximum lateness, where lateness is the amount of time by which the completion of a job exceeds its due-date,
- $\bar{L}$ — mean lateness,
- $T_{\max}$ — maximum tardiness, where tardiness is the positive lateness,
- $\bar{T}$ — mean tardiness,
- n_T — number of tardy jobs,
- $I_{\max}$ — maximum machine idle time,
- $\bar{I}$ — mean machine idle time,
- $\bar{N}_c = (1/C_{\max}) \int_0^{C_{\max}} N_c(t)\, dt$ — mean number of completed jobs, where $N_c(t)$ is the number of completed jobs by time t.
- $\bar{N}_p = (1/C_{\max}) \int_0^{C_{\max}} N_p(t)\, dt$ — mean number of jobs being processed, where $N_p(t)$ is the number of jobs in-process at time t.
- $\bar{N}_W = (1/C_{\max}) \int_0^{C_{\max}} N_W(t)\, dt$ — mean number of jobs waiting for a machine

or not ready for processing, where $N_W(t)$ is the number of jobs waiting for processing or not ready for processing at time t.

- $\overline{N}_u = (1/C_{\max}) \int_0^{C_{\max}} N_u(t)\, dt$ — mean number of jobs still to be completed, where $N_u(t)$ is the number of jobs still to be completed, by time t.

We next define an important class for performance measures. Let C_i be the completion time of job i. A measure of performance R is said to be *regular* if it is non-decreasing in completion times, *i.e.* if

$$C_1 \leq C'_1;\ C_2 \leq C'_2;\ \cdots;\ C_n \leq C'_n$$

is satisfied, then

$$R(C_1, C_2, ..., C_n) \leq R(C'_1, C'_2, ..., C'_n).$$

Observe that $\overline{C}$, $C_{\max}$, $\overline{F}$, $F_{\max}$, $\overline{L}$, $L_{\max}$, $\overline{T}$, $T_{\max}$ and n_T are all regular measures.

Two measures, R_1 and R_2, are termed *equivalent* if the following holds — a schedule S is optimal with respect to R_1 if and only if schedule S is optimal with respect to R_2. The following results are proved in French [Fre82a] regarding the equivalence of some of performance measures.

- $\overline{C}$, $\overline{F}$, $\overline{W}$, and $\overline{L}$ are equivalent measures of performance.
- If a schedule S is optimal with respect to $L_{\max}$, then S is optimal with respect to $T_{\max}$.
- $C_{\max}$, $\overline{N}_p$, and $\overline{I}$ are equivalent measures of performance.
- $\overline{N}_u$ and $\overline{C}/C_{\max}$ are equivalent measures of performance.
- $\overline{N}_W$ and $\overline{W}/C_{\max}$ are equivalent measures of performance.

- For the single machine scheduling problem $\bar{C}$, $\bar{F}$, $\bar{W}$, $\bar{L}$, $\bar{N}_n$, and $\bar{N}_W$ are equivalent measures of performance.

3.2 – Single Machine Scheduling

Pioneering work on single machine scheduling is that of Jackson [Jac56a] and Smith [Smi56a].

The following two theorems for single machine scheduling allow us to only consider permutation schedules when searching for an optimal single machine schedule [Con67a]. Permutation schedules are schedules that are completely determined by the job sequence.

Theorem – For an n-job single machine static scheduling problem with a regular measure of performance B there exists an optimal schedule with no inserted idle time.

Theorem – For an n-job single machine scheduling problem with a regular measure of performance B no improvement can be gained by allowing preemption.

Graham *et al.* [Gra79a] provide an excellent survey of results for single machine problems. We limit ourselves to four basic results. The first result is old, and is proven, among several authors, by Conway *et al.* [Con67a], Rinnooy Kan [Rin76a], and French [Fre82a].

Theorem – For the n-job single machine scheduling problem a minimum $\bar{F}$ schedule is such that

$$p_{i(1)} \leq p_{i(2)} \leq \cdots \leq p_{i(n)}$$

where $p_{i(k)}$ is the processing time of the k-th job processed, *i.e.* processing the shortest operation next is optimal.

The second result, which concerns scheduling with due-dates, is due to Jackson [Jac55a].

Theorem — For the n-job single machine scheduling problem a minimum $T_{\max}$ schedule is such that

$$d_{i(1)} \leq d_{i(2)} \leq \cdots \leq d_{i(n)}$$

where $d_{i(k)}$ is the due-date of the k-th job processed, *i.e.* processing the job with the earliest due-date next is optimal.

The third basic single-machine result reviewed here concerns preemption. Two forms of preemption have been studied in the literature. In *preempt-resume* processing is resumed at the point of preemption, whereas in *preempt-repeat* current processing is restarted from the beginning of the preempted operation.

Theorem — If (1) the job arrival stream is independent of the dispatching rule used, (2) processing time of a job is independent of preemption, and (3) preempt-resume is allowed, then the policy that minimizes $\bar{F}$ is one in which the job with the shortest remaining processing time is being processed. If a shorter job arrives during processing, this policy requires preemption of the current job in favor of the shorter job.

This result is proven by Schrage [Sch68a].

The final result is due to Klimov [Kli74a]. It regards scheduling jobs that require a sequence of operations on a single machine. More specifically, let us assume that jobs arrive according to a Poisson process with rate λ. Each job is processed k times, with the j-th operation having duration that is distributed according to d.f. $\beta_j(t)$. A job waiting for its j-th operation is queued in buffer j, $j = 1, \ldots, k$, in order of arrival. There is no delay from the time a job finishes processing to the time it enters its next buffer. If preemption is not allowed, the following theorem holds.

Theorem — A policy that minimizes $\bar{F}$ is one that processes next the first job in the buffer with the largest index, *i.e.* shortest remaining processing time is optimal.

3.3 — Multi-Machine Scheduling

The general job shop problem is extremely difficult to solve optimally [Gra79a]. Say Conway *et al.* [Con67a] about this class of problems.

> The general job-shop problem is a fascinating challenge. Although it is easy to state, and to visualize what is required, it is extremely difficult to make any progress whatever toward a solution. Many proficient people have considered the problem, and all have come away essentially empty-handed. Since this frustration is not reported in the literature, the problem continues to attract investigators, who just cannot believe that a problem so simply structured can be so difficult, until they have tried it.

An O($n$log n) minimum makespan scheduling algorithm is proposed by Jackson [Jac56a] for the job shop with two machines and no more than two operations per job. When there are more than two machines or more than two operations per job, the problem has been shown to be NP-complete [Gar76a]. Several integer and mixed-integer programming formulations have been proposed for the static deterministic job shop scheduling problem. For a survey of these studies see Conway *et al.* [Con67a], Baker [Bak74a], and Rinnooy Kan [Rin76a]. Manne [Man60a] was among the first to propose an integer programming formulation for job shop scheduling. Greenberg [Gre68a] presents a branch and bound solution technique for Manne's formulation. Other work involving integer programming for job shop scheduling includes Wagner [Wag59a], Bowman [Bow59a], Dantzig [Dan60a] and Fisher [Fis73a, Fis73b]. Story and Wagner [Sto63a] report on computational experience with integer programming for job shop scheduling. Integer programming is itself NP-complete [Kar72a, Bor76a] and is hence computationally infeasible for problems encountered in practice. Ashour [Ash70a] investigates relaxing the integrality constraint of the integer program and solving the resulting linear program with subsequent rounding off via a heuristic. He reports that the resulting solutions do not seem sufficiently close to the optimal to warrant the computational effort involved. Schedule construction algorithms begin with a partial schedule and at each iteration heuristically schedule another job until all jobs are scheduled. Gliffer and Thompson [Gli60a] propose a schedule construction algorithm for static deterministic job shop scheduling. Exhaustive enumeration generates all possible schedules and selects the best. This is clearly infeasible for problems of even relatively small dimension. A job shop having n jobs and m machines may have up to $(n!)^m$ possible schedules, some of which may be infeasible because of routing constraints. For example, a 10 job, 5 machine job shop can have 6.29×10^{32} feasible schedules (about 100,000 times the mass of the Earth in grams). Branch and bound techniques, Ashour [Ash74a], Brooks and White [Bro65a], Balas [Bal69a], Lageweg *et al.* [Lag77a], lessen the computational burden, but only allow solving small problems. In fact, a solution to a 10-job, 10-machine static deterministic job shop problem proposed in 1963 by Muth and Thompson [Mut63a] was only proven optimal in 1986 [Len86a].

The most used approach to model dynamic probabilistic job shops is via network of queues. Early analytical studies of network of queues are due to Jackson [Jac57a, Jac63a]. His analyses, however, rely on assumptions that render them difficult to apply to real job shops. These assumptions are:

- The inter-arrival times for jobs arriving from outside the shop are exponentially distributed.
- Processing times of jobs are exponentially distributed.
- Job routes in the shop are determined by a fixed transition probability matrix.

Lemoine [Lem77a] surveys the state-of-the-art of the theory of network of queues.

The heuristic approach most widely discussed in the literature for scheduling job shops (both static and dynamic) is dispatching. A dispatching rule is used to select the next job to be processed by a machine from a group of jobs queued in front of that machine. Dispatching rules may use global shop information, but almost the totality of research has focussed on rules that involve only local information, *i.e.* information regarding the current job step and the station where the job is processed.

Panwalker and Iskander [Pan77a], and Blackstone *et al.* [Bla82a] survey dispatching rules. Panwalker and Iskander classify rules into five categories.

- Simple rules usually rely on information related to a specific job, such as due-date, processing times, size of next queue, arrival time, and number of operations. Random selection is considered to be a simple rule.
- In rules that combine simple rules, queued jobs are grouped and each group obeys a distinct dispatching rule.
- Rules that combine several rules with weighted indices are termed weighted priority rules.
- Heuristic scheduling rules take into account more complex information, such as alternate routes, machine failures, and anticipated machine loads. These rules are usually used in combination with simple rules. We later survey heuristics that are described and compared by Gere [Ger66a].
- All rules that cannot be placed in one of the first four classes are classified as *other rules.*

Others have offered different classifications. One worth mentioning is that of Conway *et al.* [Con60a]. They classify priority rules into four categories:

- lateness rules, or rules that determine priority as an increasing function of job lateness;
- arrival order rules, or rules that assign priorities according to the order in which jobs arrived at a machine;
- rules that are determined by some property of the job itself;
- the random rule, which is the rule that assigns priorities at random.

Panwalker and Iskander [Pan77a] collect over 100 dispatching rules and cross-reference them to over 30 papers. Their cross-reference shows the most discussed rules in the literature are, in decreasing order:

- shortest imminent processing time (SIPT) — Select the job with the shortest processing time at this machine.
- first in first out (FIFO) — Select the job that first arrived at the queue of this machine.
- earliest due-date (DD) — Select the job that has the earliest due-date.
- longest imminent processing time (LIPT) — Select the job with the longest processing time at this machine.
- at random (RANDOM) — Select a job at random.
- least slack time (LS) — Select the job with least difference between due-date and expected remaining processing time.
- least slack time per number of operations remaining (LS/OP) — Select the job with least ratio of slack to number of operations remaining.
- longest remaining operation (LRPT) — Select the job with the longest remaining processing time.
- shortest remaining operation (SRPT) — Select the job with the shortest remaining processing time.

- fewest operations remaining (FOPRM) — Select the job with the fewest remaining operations at the shop.
- first into shop first out (FASFO) — Select the job that has been the longest in the shop.
- most operations remaining (MOPRM) — Select the job with the most remaining operations.
- truncated shortest imminent operation (SIPT/FIFO) — For jobs in the queue more than T time units select the job that first arrived in the queue. If no job is in the queue for more than T time units, select the job with shortest operation at this machine.

Several of these rules as well as others will be discussed in the remainder of this paper. Most studies involve the use of simulation. We next review some aspects of simulation of job shops.

3.4 — Job Shop Simulation Research

Because of the stochastic nature of the dynamic job shop, most studies of job shop dispatching make use of digital simulation. Early job shop simulation research is due to Jackson [Jac57b], Jackson and Nelson [Jac57c], and Baker and Dzielinski [Bak60a]. Several reviews of job shop simulation have been published. In particular, see Conway *et al.* [Con67a], Moore and Wilson [Moo67a], Day and Hottenstein [Day70a], and Baker [Bak74a]. Day and Hottenstein [Day70a] and Gere [Ger66a] describe simulation program housekeeping in detail.

Simulation models typically have several input parameters — job arrival process (rates and probability distributions), service process (rates and probability distributions), number of machine centers, number of machines in each center, dispatching rule, and a mechanism for generating job routes. Arrivals are, in most studies, Poisson with arrival rates set so as to avoid overload. Reinitz [Rei63a] justifies the use of the Poisson arrival process. Kuratani and Nelson [Kur60a], however, conclude that the job inter-arrival process should have a distribution resembling both the exponential and the Erlang distributions. Processing times at each machine are also assumed to obey probability distributions. There is more variety of distributions reported in the literature for processing time than for inter-arrival times, *e.g.* exponential [Bak60a, Con62a], Erlang [Kur60a], hyper-exponential [Kur60a], and normal [Con60a].

Shop size, or number of machines in shop, range from five [Con60a] to 1000 [LeG66a]. Baker and Dzielinski [Bak60a] compare shops with nine to thirty machines and conclude that shop size is not significant. Conway and Maxwell [Con62a] and Nanot [Nan63a] agree that shop size has no significant effect on the relative performance of dispatching rules. Therefore, Blackstone *et al.* [Bla82a] conclude, a small shop can be used, with considerable computational savings. The mechanism most reported in the literature for generating job routes is the use of a probability transition matrix to determine what will be the next station to visit, upon completion of service at the current station. Conway and Maxwell [Con62a] conclude that changes in the transition probability matrix do not significantly influence the relative performance of dispatching rules. A common practice is to set bounds on the number of operations a job is allowed to perform [Con60a, Kur60a]. Morse [Mor58a] observes that when alternate routes are allowed performance improves. As with any simulation study, attention should be paid to the statistical analysis of simulation output [Law83a]. Statistical issues discussed in the literature regard warm up, data collection, and methods for output analysis. The central issue of warm up is to decide when to start data collection. Conway [Con65a] and Kuratani and Nelson [Kur60a] discuss warm up strategies. Data collection should be done in a way that all estimates are meaningful. Conway *et al.* [Con60a] suggest that a simple scheme is to select a fixed number of consecutive jobs from the output and use only values from those jobs to compute simulation statistics. Hershauer and Ebert [Her75a] use an online method, due to Fishman [Fis71a], for computing and collecting the sample size needed to estimate the mean of a simulated process with a desired degree of accuracy. They do not, however, report any numerical results. Several statistical methods have been used to analyze simulation output — correlation [Dzi63a], Student's *t* [Row60a], analysis of variance [Bak60a], and factorial design [Bak60a, Con62a].

3.5 — Simple Dispatching Rules

Of the many dispatching rules reported in the literature [Pan77a] the most studied rule is shortest imminent processing time (SIPT). SIPT assigns the highest priority to the job that has the shortest processing time for that operation. Baker and Dzielinski [Bak60a] conclude that SIPT is the best rule if the objective is to minimize mean flowtime. Conway [Con65a] compares SIPT with 31 dispatching rules and concludes that SIPT dominates those rules despite the fact that it did not achieve the best performance in any of the experiments. In another study [Con62a] SIPT was shown to be insensitive to errors in estimated processing times as well as to shop size. In yet another study [Con65b] SIPT was shown to be the least sensitive of 39 rules to the due-date selection procedure used. Elvers [Elv73a] investigates the performance of 10 dispatching rules with regards to five variations of the TWK due-date setting rule, a rule based on total work content. His conclusion is that with respect to tardiness SIPT outperforms the other 9 rules investigated.

Of the simple rules, SIPT seems to be the most promising. Says Conway about SIPT [Con65a].

> Its performance under every measure was good, it was an important factor in each of the rules that exhibit a *best* performance under some measure, and it is simpler and easier to implement than the rules that surpass it in performance. It surely should be considered the *standard* in scheduling research, against which candidate procedures must demonstrate their

virtue.

SIPT does tend, however, to cause some jobs to become extremely late. This can be somewhat overcome by applying the truncated SIPT rule, where SIPT is applied to any job having entered the queue after some time T, while jobs entering before T have the highest priorities and are among themselves selected by the first-in-first-out (FIFO) rule. Another approach is to simply alternate between SIPT and FIFO or some other rule. Conway and Maxwell [Con62a] conclude that some form of truncated SIPT rule is the best hope for ridding SIPT of the increase in variance that often results from its use. Eilon and Cotterill [Eil68a] and Eilon *et al.* [Eil75a] report on the success of a truncated SIPT rule that they call SI^x. In this rule, F_i, a function of the time until due-date and remaining processing time, is computed for each job i. If any $F_i \leq 0$, the job with the smallest F_i value is selected. Otherwise, SIPT is applied. Oral and Malouin [Ora73a] describe another truncated SIPT rule they name SPT-T. This rule selects the minimum between SIPT + γ and slack time (difference between time until due-date and time of remaining operations), where γ is a control parameter. When γ is large the SPT-T rule becomes a pure slack rule. Yet another rule that attempts to reduce the variance in flowtime is the *COVERT* rule. This rule selects the job with the smallest ratio of delay cost to imminent processing time. In a Ph.D. thesis by Carroll [Car65a] this rule produced schedules that were superior to those generated by truncated SIPT, in the sense of less variance of flowtime.

Sen and Gupta [Sen84a] review the literature of static scheduling research involving due-dates. Their survey, however, does not cover dispatching rules. Studies involving rules that use due-date information require due-dates to be set for every job entering the shop. Conway [Con65a] classifies due-date setting rules into two categories — those that are exogenously determined and those that are set internally. In the former group he studies the constant (CON) and the random (RAN) due-date setting procedures. In the latter group he examines total work content (TWK) and number of operations (NOP) procedures. Baker and Bertrand [Bak81a] compare three due-date setting rules: CON, TWK, and slack (SLK) and conclude that due-date setting error is minimized on average by TWK. For other studies on due-date setting rules see Baker [Bak74a], Conway [Con65b], Elvers [Elv73a], Jones [Jon73a], Reiter [Rei66a], Seidmann and Smith [Sei81a], and Baker [Bak84a]. Ragatz and Mabert [Rag84a] develop a model for due-date management.

The main characteristic of due-date based rules is that they usually produce schedules that have less number of late jobs, and flowtimes with less variance than schedules determined by processing time based rules [Con65b]. The most studied due-date based rules are —

- Earliest Due-Date (DD) — Select the job with the earliest due date.
- Least Slack Time (LS) — Select the job with the minimum difference between time remaining to due-date and remaining expected processing time.
- Least Slack per Operation — Select the job with the minimum ratio of slack to total expected remaining processing time.
- Least Job Slack Ratio — Select the jobs with the least ratio of slack to remaining time until due-date.
- Modified Minimum Slack — Select the job with the minimum difference between time remaining until due-date and expected time remaining in shop (processing and waiting).
- Least Modified Job Slack Ratio — Select the job with the least ratio of modified slack to time remaining until due-date.
- Critical Ratio — Select the job with the smallest ratio of time remaining until to due-date to lead time remaining. This rule involves estimating

queueing time remaining.

Of the rules described above, least slack per operation appears to perform best [Ger66a, Her75a]. In a study by Rochette and Sadowski [Roc76a] earliest due-date produced the least mean tardiness when the shop was 80% loaded, but at a higher 95% load SIPT proved to be superior. The critical ratio rule is used much in industry, but little has been published about its performance. For studies on the critical ratio rule, see Berger [Ber70a], Putnam *et al.* [Put71a], Wassweiler [Was72a], Berry and Rao [Ber75a], and Berry *et al.* [Ber84a].

First-in-first-out (FIFO) has been studied extensively in the literature. Panwalker and Iskander [Pan77a] refer to 16 studies on FIFO. The general consensus is that FIFO is no better, in terms of flowtime, than RANDOM, except that it produces schedules with somewhat less variance than those generated by RANDOM. Conflicting results [Con65b, Roc76a] indicate that first-into-shop-first-out (FASFO) and FIFO have no significant difference [Bla82a]. Least number of operations remaining (LNOPR) performed poorly in a study by Rochette and Sadowski [Roc76a].

The number in next queue (NINQ) selects for processing the job whose next operation is at the station with the smallest current queue length. With the intent of doing better than SIPT, Conway and Maxwell [Con62a] test NINQ using SIPT as a tie breaker. They observe that while NINQ did better than RANDOM, it failed to outperform SIPT. A variation of NINQ is the work in next queue rule (WINQ). This rule selects for processing the job that will go next to the queue with the least work content. Conway [Con65a] compares NINQ and WINQ with SIPT. Neither one of the look-ahead rules produced mean flowtimes better than those generated by SIPT.

3.6 — Heuristics

Gere [Ger66a] investigates how heuristics can boost the performance of simple dispatching rules. He suggests eight heuristics, of which four are describes below.

- **Alternate Operation** — If scheduling a selected job causes some other job to have a negative due-date slack, revoke this assignment and attempt to schedule the next job on the priority list. If this job does not cause any slack to become negative, schedule it. Otherwise, schedule the original job indicated by that dispatching rule.
- **Look Ahead** — If one could look ahead and anticipate every possible conflict, an optimal schedule would be possible. This would require almost exhaustive enumeration and is therefore infeasible. One can, however, examine a small portion of possible conflicts with little cost. One example of such look ahead strategy is as follows. Upon scheduling a job, attempt to identify if any late or nearly late lot is about to arrive at that station. If one is found, idle time is scheduled until this late job arrives whereupon it is scheduled.
- **Insert** — If there is a job in the queue whose processing time is less than that of the duration of idle time scheduled by the above look ahead strategy, that short job should be scheduled. The insert heuristic says that the longest of any such job should be scheduled.
- **Re-do with Adjusted Due-Dates** — This heuristic is suggested for the static scheduling problem. First devise a schedule and apply it. For all late jobs, decrease their due-dates by the amount they were late and re-do the schedule.

Gere compares eight simple rules with and without his heuristics on several job shops and concludes:

- When the performance criterion is minimization of mean tardiness, the simple priority rule selected is not as important as the selection of the set of heuristics which boost the rule.
- When a rule is mixed with one or more heuristics there makes little difference which rule was chosen. Therefore a simple rule should be used.
- Heuristics that anticipate the future state of the system, such as alternate operation and look-ahead with insert improve schedules significantly both statistically and practically.

The implication of his conclusions is that developing more complex rules is a worthwhile endeavor.

3.7 — Search-Based Dispatching Rules

Fischer and Thompson [Fis63a] were amongst the first to suggest a rule that would adapt itself to the shop in question. Their rule, in essence, selects at random one out of two dispatching rules every time a lot dispatch is carried out. Initially, both rules are given identical probabilities of being selected. As learning takes place the probabilities are changed. Their most important, and surprising, finding is that even when one uses a random selection of the rules, the combination almost always outperforms either rule when used individually.

Hershauer and Ebert [Her75a] generalize the Fischer and Thompson idea by introducing a priority dispatching rule defined by a linear combination of several simple rules, each of which is initially assigned a relative weight. By using simulation and a search technique values for weights are found such that the combined rule is locally optimized.

Several authors have discussed issues related to solving optimization problems involving simulation. For a survey of the field, see Smith [Smi73a] and Biles and Swain [Bil79a]. Techniques for solving such problems can be categorized into three groups: (1) direct search techniques; (2) first-order surface response methods; and (3) second-order surface response methods. Direct search techniques are those that do not require derivatives. Most known of these methods are pattern search of Hooke and Jeeves [Hoo61a], the sequential simplex method of Spendley, Hext, and Himsworth [Spe62a], and the complex method of Box [Box65a]. First-order methods estimate the gradient experimentally and apply the method of steepest descent. To estimate the gradient about the current point, a designed experiment, such as a 2^N factorial design, is conducted around the point with a simulation run for each point. Then, a first-order regression model is fit on this data. In second-order response surface methods non-linear programming algorithms are applied to a surface obtained by fitting a second-order regression model to the data obtained from a second-order response surface experimental design. Biles and Swain compare algorithms of the three classes in several problems and conclude that the effectiveness of the procedures depends on the error of the estimates made with the simulation and the topology of of the search surface. First-order methods are inefficient in the presence of large errors while second-order methods fail in the presence of irregular surfaces. Direct search methods, the authors conclude, are the most promising in these situations. Of these techniques, Hooke and Jeeves pattern search [Hoo61a] has received the most attention.

Taubert [Tau68a] was the first to use the Hooke and Jeeves pattern search in the context of scheduling. He shows the effectiveness of pattern search in solving the aggregate scheduling problem by searching for a local optimum on a 20 dimension response surface.

Hershauer and Ebert [Her75a] were first to suggest the use of pattern search [Hoo61a], to optimize the weighted balance in a multi-rule dispatching scheme.

Pattern search is a direct search method used when the first derivative of the objective function is either unavailable or its computation is unreliable. This is typically the case in the search problems that show up in optimal multi-rule dispatching. Small changes in the vector of weights **w** can alter the sequence in which jobs are processed and cause step changes in the observed output. This makes the search surface, $h(\mathbf{w})$, rugged or discontinuous and consequently $\nabla h(\mathbf{w})$ is undefined in many points.

The algorithm is given an initial solution $\mathbf{w}_0 \in \mathbf{R}^N$ and an initial step size δ_0. At each iteration, an *exploratory search* seeks an improving neighbor, $\mathbf{n} \in \mathbf{R}^N$, of the current best **w**. Each evaluation of the function $h(\mathbf{w})$ is carried out by running the simulation model with weight vector **w**. If an improving neighbor is found, a *pattern move* is carried out. In a pattern move, an exploratory search is conducted around point $2\mathbf{n} - \mathbf{w}$. This point is a likely candidate for improvement since, at least intuitively, it follows a historic *pattern* of improvement in the objective function. If an objective value better than $h(\mathbf{n})$ is found by the pattern move, **w** is set to this improved solution and another pattern move is applied. Otherwise **w** is set to **n**. On the other hand, if in the exploratory search conducted around **w** in the main loop of the algorithm, no improving point is found, the search step size is decreased and a new iteration begins. The algorithm terminates when the step size is smaller than a given minimum step length, $\underline{\delta}$.

Harrison and O'Grady [Har85a] suggest two modifications to the algorithm. The first resets the current search step size to its initial value every time an improving solution is found. This attempts to avoid an either too coarse or too fine sampling step. The second, and perhaps most important modification, takes place after the current step size falls below the given minimum step length $\underline{\delta}$. Instead of terminating the search, as one would do in the original version, the modified algorithm is continued with the current step size being increased every time an exploratory search fails. This measure is designed to help the algorithm avoid getting caught in local minima. The search is declared over when the step size reaches a value greater than a given maximum step length $\bar{\delta}$.

Besides the modifications described above, Harrison and O'Grady observe that there is no unified format in which individual manufacturing environments can be placed, and consequently methods must be tailored to specific cases, making them difficult to implement in practice. They propose a general representation of priority that they suggest overcomes this difficulty. They describe how this adaptive scheme can be feasibly implemented as a real-time factory control system. In numerical experimentations a weighted priority rule is set using shortest imminent processing time, earliest due-date, latest due-date, minimum slack time, random, and maximum slack time, and a pattern search attempts to optimize the sum of mean flow time, mean idle time, mean waiting time, mean lateness, mean earliness, and mean tardiness. Results on a ten-machine shop show an improvement of as much as 31 percent and on the average of 16 percent with respect to the same rules used individually. The main result is that improvement was observed in every case studied.

Bunnag and Smith [Bun85a] implement a multifactor priority rule scheme with pattern search using a generalized objective function. Their objective is to minimize the sum of the cost of tardiness, in-process inventory, and idle machine time. The priority rule used is a linear combination of shortest imminent processing time, dynamic slack time per remaining operations, time remaining per remaining unit of processing time, and work in next queue. The initial base point is selected by choosing from the extreme point set, {(1,0,0,0), (0,1,0,0), (0,0,1,0), (0,0,0,1)}, the point with the best objective function value. The multifactor rule is compared to four widely used rules, random, shortest imminent processing time, slack per remaining operations, and the critical ratio rule (ratio of remining time to remaining processing time) on several test problems. Improvement of the multifactor rule with respect to these rules varied from 7.26 to 43.22%.

3.8 — Scheduling Integrated Circuit Manufacturing

In the last few years there have been several publications related to scheduling and performance evaluation of integrated circuit (IC) fabrication.

Lohrasbpour and Sathaye [Loh84a] describe a simulation model of IC wafer fabrication lines that estimates work-in-process (WIP), throughput time, production rate, and labor and equipment utilization given a set of input parameters, *e.g.* product mix and wafer input rate. The authors discuss several applications of the model, namely capacity analysis, analysis of variability in wafer input rate, the effect of express (hot) lots, WIP reduction policies, and the effect of equipment downtime. Their most important observation is that by reducing the input coefficient of variation from 0.43 to 0.08 they were able to reduce cycle times by 20%, on average. They conclude that uniform starts are highly desired.

Dayhoff and Atherton [Day84a] describe the dynamics of wafer movement and identify several problems that can be analyzed by discrete event simulation, *e.g.* fab start-up, clearing of accumulated inventory, and the analysis of inventory/cycle-time/throughput tradeoff. The authors introduce the concept of fab signatures, a trio of plots displaying the effect of lot start rate on mean inventory, mean cycle time, and departure rate. The authors illustrate the use of the model by studying the effect of a poor dispatching rule, LRPT, on an IC fab.

Leachman [Lea86a] describes a linear programming based corporate-level planning model for the semiconductor industry.

Lozinski [Loz86a] identifies technological constraints that should be satisfied by an IC wafer fab scheduler.

Wein [Wei86a] compares several dispatching rules and lot release policies on three fictitious fabs (using data gathered at a Hewlett-Packard research fab) with respect to mean cycle time. Several of the scheduling rules used by Wein are derived by approximating the fab by a sub network of queues restricted only to stations that are heavily loaded and using a Brownian network model that approximates a multiclass queueing network model with dynamic control capability. Wein, in the same way was Lohrasbpour and Sathaye, observes that reducing the variability of the input process is important in reducing cycle time. In fact, Wein concludes, release control is more important in reducing cycle time than are dispatching rules. Wein introduces a release strategy for a single product fab, named *workload regulating input*, that releases a lot into the fab every time the remaining work content at a bottleneck station for all lots still in the fab falls below a prescribed value. This scheme outperforms uniform release in his simulations.

Chen *et al.* [Che86a] present a class of queueing network models for the analysis of wafer fabrication facilities. Using these models on several years' data of a real IC wafer fab, they are able to predict key performance measures within about 10% of the values actually observed.

Burman *et al.* [Bur86a] discuss how operations research tools can be used to analyze the behavior of IC wafer manufacturing lines. In particular they present simulation, queueing, and deterministic models and suggest when it is appropriate to use each class of models.

Spence and Welter [Spe86a] use simulation to study the fab cycle-time/throughput tradeoff curves with respect to the sensitivity of changes in the operation of a photolithography cell. In particular, they examine the effects of adding resources (operators and equipment), process improvement (reducing set-up times and rework), and operating rules (lot sizing and repairman wait time).

Watts [Wat86a] discusses the integration of scheduling in the computer integrated manufacturing (CIM) environment. Composite dispatching, a dispatching scheme for load balancing is proposed. This scheme partitions the work-in-process by current mask level. The priority of a lot is inversely proportional to the number of lots in its mask level set, *i.e.* the lot selected is from the mask level set with the least number of members.

3.9 – Summary

In this Chapter we reviewed the literature of job shop scheduling (in particular, job shop dispatching) and scheduling and simulation of semiconductor wafer manufacturing.

The assumptions most found in the literature are listed, as are the most cited system performance measures. The notion of equivalent performance measures is reviewed. We enunciate several important theoretical results for the single-machine scheduling problem.

For the multi-machine scheduling problem we review attempts at solving the problem exactly and then concentrate on heuristics for approximate solutions. In particular we review the literature of job shop dispatching heuristics and job shop simulation research. We list the relevant work related to the most studied simple dispatching rules as well as more complex heuristics and weighted dispatching rules. Optimization techniques for weighted dispatching are surveyed.

We finalize by reviewing the more recent literature of shop-floor level production planning in the semiconductor industry.