As originally published in the Pan Pacific Symposium Proceedings.

ON IMPROVING OPERATIONS SCHEDULING IN ELECTRONICS MANUFACTURING

Daryl L. Santos, Ph.D.
Systems Science and Industrial Engineering Department
Small Scale Systems Integration and Packaging (S^3IP) – A NYS Center of Excellence
Binghamton University
Binghamton, NY, USA
santos@binghamton.edu

ABSTRACT
There is no doubt that manufacturers schedule operations – without doing so, product would never be shipped. However, what seems to receive comparatively little attention in many organizations is discussion on how to assign or release jobs throughout the facility in order to optimize the scheduling activity. This paper describes a general scheduling environment, known as the hybrid flow shop (HFS), which exists in many facilities. For example, surface mount assembly is one type of production flow that can be modeled as a hybrid flow shop. Fundamental concepts of HFS scheduling, performance metrics tied to costs and facility utilization, and developments towards improved scheduling techniques for HFS are reviewed in this work.

Key words: Production scheduling, scheduling complexity, hybrid flow shop, flow shops with multiple processors.

INTRODUCTION
Herein, "production scheduling" is defined as the allocation of limited resources (machines) to tasks (jobs or orders) over time with the goal of optimizing one or more performance measures. Little known in most industries, including electronics manufacturing, is just how complicated the production scheduling function is, mathematically speaking. This paper is organized by the following discussion areas:

- a brief overview of baseline production scheduling terminology,
- common assumptions used to generate a baseline schedule,
- the fundamental and common scheduling models or environments and a hierarchy of their complexity,
- common performance measures used to evaluate schedules and a hierarchy of their complexity, and
- mathematical approaches used to obtain feasible schedules.

The paper then devotes a section to the complexity of "static problems" in each of the following scheduling environments as they are found in a variety of electronics manufacturing settings: single machine scheduling, parallel processor (a.k.a., multiprocessor) scheduling, flow shop scheduling, and hybrid flow shop (a.k.a., flow shop with multiple processor) scheduling. Hybrid flow shop (HFS) scheduling will receive additional attention in this paper as, compared to the others, it is a "newer" problem as found in the literature.

SCHEDULING TERMINOLOGY
The following is a brief glossary of common terminology used in scheduling problems and in this paper:

- Job – a "job" is considered an entity and, for practical purposes, represents an order. A job may have one or more operations that need to be performed on it, depending upon the environment. In this paper, we assume that a job can only be processed on one machine at a time.
- Environment – also known as a model, the environment describes the number of machines in the facility and the routing of the jobs through the facility.
- Static problems – refers to situations in which the number of jobs "n" is fixed, and the processing times are known for each job. These types of problems are also known as deterministic scheduling problems. *Static problems are the focus of this paper.*
- Dynamic problems - in these problems, there is more uncertainty in the environment – the number of jobs is not known in advance, when they arrive is random, etc.
- Performance measure – described in more detail to follow, the performance measure (a.k.a., objective function) is used to assess the quality of the schedule.
- Technological constraints – generally refers to the processing order of the jobs, the ready times of the jobs, and the processing times of the jobs.
- Feasible schedule – denotes a schedule that does not violate any technological constraints. E.g., a job is not scheduled to take longer than its processing time, is not scheduled to start before it is ready, etc.
- Optimal schedule – denotes the best feasible schedule, in terms of the performance measure. Optimal schedules are difficult, if not impossible to obtain in a practical time frame due to the complexity of most scheduling problems.
- Preemption – describes a situation in which a job, once started on a machine, can be interrupted for another job, and then continued on the machine at a later time.
- Instance – refers to a specific scheduling problem in which the environment is known, the performance

measure is known, the number of jobs is known, the processing times are known, the ready times are known, and the job routings are known.

COMMON SCHEDULING ASSUMPTIONS

To follow is a list of common scheduling assumptions used to obtain a baseline schedule. These assumptions can be found in most texts related to production scheduling such as those by French (1982) and Pinedo (2002). It is important to note that in actual practice, some of these may be violated; nonetheless they are usually invoked to generate a baseline feasible schedule:

- Each Job is an Entity – for jobs with multiple operations (multiple-machine problems), no two operations of the same job may be processed simultaneously
- No Preemptions – an operation, once started, must be completed before another job is started on that machine
- One Visit – each job has one or more distinct operations, one on each machine stage, and cannot be processed more than once on a specific machine stage
- No Cancellation – each job must have all its operations performed
- Times Independent of Schedule - Two assumptions are usually followed: a) Set-up time is sequence independent (and is included in processing times) , b) Transfer times are either negligible or are included in processing times. Problems where the sequence significantly affects the set-up times are not considered in this work.
- In-Process Inventory Allowed – jobs may wait, in a queue, for next machine
- Idle Machines are Okay
- A machine may only process one operation at a time
- No Break-Downs will Occur During Scheduling Period
- Technical Constraints Known - Technical constraints are known in advance and they are fixed (e.g., processing orders)
- Static problems are studied/assumed in this paper. What are known and fixed are the following:
  - # Jobs (n)
  - # Machines (m)
  - Processing Times
  - Ready Times
  - Other quantities needed to define a particular ***instance***

COMMON SCHEDULING MODELS

The common production scheduling models are the following:

- Single Machine Scheduling – in single machine scheduling, there is only one machine to be scheduled.
- Multiprocessor Scheduling – in multiprocessor scheduling (a.k.a., parallel machine scheduling), there is only one stage of processing but there are multiple machines available to perform that task. Thus, a job can go to any one of the machines. Unless otherwise stated, the machines are assumed to be identical, so that a job's processing time is the same regardless of which machine it visits. In situations where the machines are not identical (either unrelated (no relationship between the speeds of the machines) or uniform (the speeds of the machines are fixed in relationship to each other)), the complexity of the problem increases.
- Job Shop Scheduling – in job shop scheduling, n jobs are to be processed on m machine stages (each stage having just one processor). However, each job has a unique routing through the shop that may or may not be the same as one or more other jobs. Job shop problems where there are multiple machines at one or more of the m stages are possible. While job shop scheduling does occur in electronics manufacturing, most electronics manufacturing examples are not job shops and will not be further discussed in this paper.
- Open Shop Scheduling – in open shop scheduling, there are multiple machines, but there are no routing restrictions on the jobs. Open shop scheduling is not common in electronics manufacturing and will not be further discussed in this paper.
- Flow Shop Scheduling – in flow shop scheduling, there are multiple machine stages and each job follows the same routing through the stages as all other jobs.
- Hybrid Flow Shop Scheduling – HFS scheduling is an extension of the flow shop problem wherein, at one or more stages, there are multiple (parallel) machines available to process the jobs.

Not surprisingly, the hierarchy of the complexity of scheduling problems in Figure 1, below, begins with the single machine problem with the HFS being one of the more complex environments.

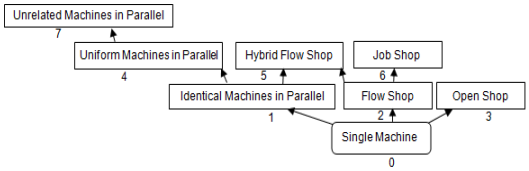

Figure 1. Complexity of Scheduling Environments (modified from Pinedo, 2002)

Later in the discussion of the mathematical complexity of many of these scheduling problems (single machine, multiprocessor, flow shop, and HFS), it will be easier to understand the structure of Figure 1.

PERFORMANCE MEASURES

This section gives a brief review of the common performance measures broken down into three different categories: Completion Time based criteria; Due-date based criteria; and Inventory & Utilization criteria. More detailed descriptions and other performance measures can be found in Pinedo (2002) or other texts.

Completion Time Based Performance Measures

These criteria are used to evaluate a schedule when the quality is gauged by when jobs complete. Some of the

common performance measures in this category are the following:

- Maximum flow time (F_{max}) – flow time is defined as the difference between when a job completes its last operation and when it was ready to begin its first operation. Like most scheduling criteria, it is desired to *minimize* the value of the largest flow time. This is done in situations where the cost of the schedule is directly related to the job that takes the longest time in the system.
- Maximum completion time (C_{max}) – also known as the makespan. Used to define when the last job exits the system. This is used when the cost of a schedule depends on how long the processing system is devoted to the entire set of jobs.
- Average flow time and average completion time – when the schedule's cost is related to average performance (instead of an individual job's performance), then minimizing the average flow time or average completion time of all of the jobs can be assessed. Minimizing average flow time is equivalent to minimizing the sum of all of the flow times; similarly, for average completion time and sum of all completion times.
- Weighted based versions – when different jobs have different weights (or priorities) – such as when a cost penalty is assessed by a customer, then average weighted flow time, average weighted completion time, or other variations may be used.

Due-Date Based Performance Measures

These criteria are used when a schedule is tied directly to how far due-date targets are missed. Prior to describing some of the common due-date based measures, it is important to define some values. The lateness of a job, i, is calculated as follows:

$$L(i) = C(i) - d(i)$$

Where C(i) represents when the job completes and d(i) represents the due date. Lateness, by the above calculation, can therefore be positive (it is tardy) or negative (it is early).

Lateness-based performance measures (e.g., minimizing the maximum lateness (L_{max}), minimizing the average lateness (which is the same as minimizing the sum of total lateness), etc.) are appropriate when there is a positive reward for completing a job early and that reward increases the earlier the job is.

When we are interested in whether a job finishes before its due-date, then tardiness is of importance. The tardiness of a job is calculated as follows:

$$T(i) = \max \{ 0, L(i) \}$$

Sometimes a binary variable, U(i), is created for a job based upon its tardiness. If it is tardy, $T(i) > 0$, then $U(i) = 1$. Otherwise, $U(i) = 0$.

Tardiness-based measures (minimizing maximum tardiness, minimizing the number of tardy jobs, n_T or $\Sigma_i U(i)$, minimizing the average tardiness, etc.) are appropriate when early jobs bring no reward; there are only penalties for jobs with positive lateness.

In some situations, the earliness is important to assess and we may not want jobs to be too early. The earliness of a job is calculated as follows:

$$E(i) = \max \{ 0, -L(i) \}$$

In the descriptions, above, many due-date based criteria were mentioned. In addition to the above, weighted based versions of many of the above have also been studied.

A hierarchy of the complexity of some of the scheduling criteria from the above two categories appears in Figure 2, below.

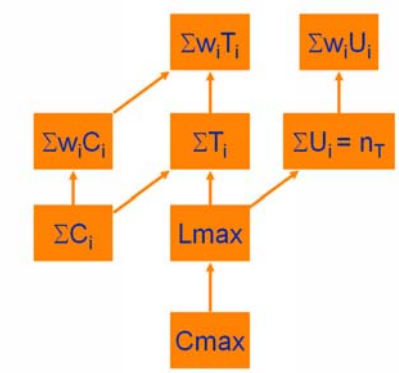

Figure 2. Objective Function Hierarchy (modified from Pinedo, 2002)

Inventory and Utilization Based Criteria

Most of the production scheduling research is aimed at criteria in the two aforementioned categories: completion based criteria and due-date based criteria. When schedule costs are tied to inventory and machine-utilization, then other criteria can be important to gauge a schedule including, but not limited to the following:

- Mean Number of Unfinished Jobs
- Mean Number of Jobs Waiting for M/C's
- Mean Number of Jobs Being Processed at Any Time

SCHEDULE GENERATION TECHNIQUES

A hierarchy of the complexity of scheduling environments has been provided in Figure 1. Discussion to follow will provide the reader an understanding of the "combinatorial explosive" nature of scheduling problems. Throughout the history of the study of scheduling problems (with most of the important findings beginning in the 1950s), techniques for finding a schedule for a particular problem instance can be broadly categorized in these two areas: optimal techniques and heuristic techniques.

In this paper, an optimal technique is one that is guaranteed to find an optimal solution by considering (explicitly or implicitly) all possible schedules and choosing the best from among them. Most optimal techniques are algorithms in the

sense that they find, in a certain amount of iterations, the best feasible schedule. Examples of optimal techniques are mixed integer linear programming (MILP) models, branch-and-bound (B&B) models, and dynamic programming (DP) models. Other optimal techniques take advantage of the structure of the problem and construct an optimal schedule. A classical example of this is Johnson's Algorithm (1954) which finds optimal makespans in two-stage flow shop problems. In rare cases, some optimal techniques are quick (such as Johnson's Algorithm) wherein an increase in the problem size – whether in number of jobs, number of machines, etc. – does not cause a large increase in solution times. In most cases, due to the total possible number of schedules to evaluate – examples of these calculations to follow – optimal search techniques like MILP, B&B, and others may fail to find an optimal solution due to constraints. As such, the search for an optimal solution may need to be abandoned in lieu of finding a good, hopefully near-optimal solution but in a quick time frame.

For the purposes of this paper, a heuristic is not guaranteed to find an optimal solution. Heuristics are designed to give good, hopefully near-optimal (if not optimal) solutions in a relatively quick time frame. Some heuristics are designed to take advantage of specific scheduling environments; for example, there are many heuristics developed for the minimum makespan flow shop scheduling problem (e.g., the NEH heuristic (Nawaz et al. (1983)), the CDS heuristic (Campbell et al. (1970)), and others).

In recent years, there are more generalized heuristics that have been developed that can be used regardless of the scheduling environment or even the performance measure. The Shifting Bottleneck Method (SBM) – for reference, see Pinedo (2002) – and the Exchange Heuristic – see Yang and Ignizio, 1987 and subsequent advancements – are methods that can be applied in many scheduling environments and with many different performance measures. Also, the newer "metaheuristics" such as Simulated Annealing, Tabu Search, Genetic Algorithms, Ant Colony Optimization, and Particle Swarm Optimization are not restricted to specific models. Generally speaking, the techniques mentioned in this paragraph are improvement methods; wherein, the heuristic begins with one or more feasible schedules and, based upon modifications of these schedules (i.e., searching in the neighborhood of known solutions), tries to find other schedules with improved performance measures.

In the descriptions of scheduling environments to follow (single machine through hybrid flow shop), their comparative complexities will be demonstrated by using a situation in which the number of jobs to be scheduled is not a large number: 10. Further, for simplification purposes, it is assumed that the ready times for all jobs in each of the examples are the same and no other special constraints (e.g., strict predecessor relationships) are considered.

SINGLE MACHINE SCHEDULING

Single machine scheduling is manifested in a variety of ways in electronics manufacturing. First of all, in some organizations, perhaps there is just one machine that is the bottleneck or "Herbie" (see Goldratt (1984)). Other machines in the same facility may have very little utilization. Thus, it may be more important to expend resources to effectively schedule that single bottleneck. In another case, from a macro-perspective, there may be some facilities that have multiple machines, but when the entire facility is devoted to one job at-a-time, then the entire facility can be treated as a single machine and scheduling the facility is thus a single machine scheduling problem. In yet another situation, surface mount assembly lines (for simplicity's sake, let's consider 3 steps: stencil printing, pick-and-place, and reflow), can be scheduled as a single machine model. Why? Because in most instances, these three steps are connected via conveyors and once the decision is made to schedule a particular board type, then the entire SMT line is devoted to that board type, before the next board type is sequenced to be assembled.

The most fundamental of all scheduling problems, the single machine model, can be a non-trivial problem, depending upon the performance measure. For example, average tardiness, total weighted tardiness, and others, are said to be NP-hard problems, a mathematical and computer-science term meaning that as the problem size increases, the time needed to search all possible schedules is non-polynomial in nature. For simplicity's sake, if the number of jobs increases by a factor of 2, not only would the time needed to search all possible schedules not only double, but more likely the increase in time would grow exponentially or at another quick rate. For single machine scheduling, if we want to schedule n jobs (with all jobs having the same ready times), then the total number of sequences to consider is $n!$.

For example, if we have 10 jobs in a single machine problem instance, then the total number of possible sequences is 10! or over 3.6 million schedules. Finding the best schedule from among the over 3.6 million possibilities would be like finding a needle in a haystack.

MULTIPROCESSOR SCHEDULING

There are multiple examples in which multiprocessor scheduling can appear in electronics manufacturing. Consider one example in which we have 3 SMT lines. As described above, an SMT line can be modeled as a single machine problem. If we have n jobs to be scheduled among m parallel machines, then the total number of schedules to consider is

$$ {}_{n-1}C_{m-1} \frac{n!}{m!} $$

In the above, "C" represents the combinatorial operator from mathematics. So, in an example where we have 10

jobs to be scheduled among 3 SMT lines, the total number of schedules is not small:

$$ {}_9C_2 \frac{10!}{3!} = 21{,}772{,}800 $$

FLOW SHOP SCHEDULING

In some electronics facilities where the material flow is not highly conveyorized (e.g., material moves from station to station on carts or pallets and other situations), the environment may be modeled like a flow shop. While not common, some facilities do not conveyorize SMT operations, thus, since the job orders can change in going from one machine (e.g., stencil printing to pick-and-place or pick-and-place to reflow), then the SMT line can be scheduled as a flow shop. Or, in another example, and generally speaking, in boards that require THT operations, then SMT operations (even if the SMT operation can be modeled as a single machine), and then an inspection stage, then this can be considered as a flow shop with three stages: THT, SMT, and inspection. The new roll-to-roll (R2R) flexible electronics facility at the Center for Advanced Microelectronics Manufacturing (CAMM) at Binghamton University can also be modeled as a flow shop. In the fundamental process steps (see Figure 3, below), a web can go from vacuum deposition, to photolithography, to wet etch.

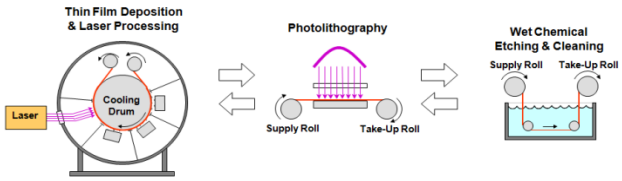

Figure 3. Flex Electronics Assembly: A 3-Stage Flow Shop

The number of possible schedules in a flow shop with n jobs and m stages of processing (with only one processor at each stage) is given by the following: $n!^m$.

So, if we consider 10 jobs that need to be scheduled in a 3-machine flow shop, then the total number of possible schedules is $10!^3$, which is an astronomical ~ 4.8×10^{19}.

HYBRID FLOW SHOP (HFS) SCHEDULING

Compared with other scheduling environments that have been studied in the literature for over 50 years, HFS literature is rather in its infancy. HFS environments are prevalent in many industries including, but not limited to the following (Santos et al., 2001): PCB assembly, flexible manufacturing systems, and petrochemical production. The process flow for a HFS is shown in Figure 3.

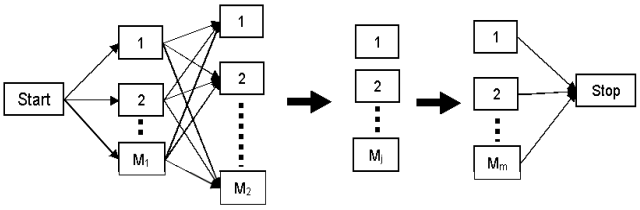

Figure 3. HFS Process Flow

Although one of the first articles on the environment did appear in the early 1970s (Salvador, 1973), most of the HFS literature did not begin to appear until the late 1980s (e.g., see Wittrock (1985), Brah (1988), and Gupta (1988), among others) and it began to grow in the 1990s (e.g., see Hunsucker and Shah (1994), Santos et al. (1995a, 1995b), Gupta and Tunc (1991, 1994), and others). If it is assumed that the machines are identical in processing speed at stages where multiple processors exist, then the number of possible sequences that can be generated for an HFS is provided by the following formula (Brah and Hunsucker, 1991):

$$ \prod_{j=1}^{m} {}_{n-1}C_{M_j-1} \frac{n!}{M_j!} $$

In the above, n is the number of jobs, m is the number of processing stages, and M_j is the number of identical machines at stage j. Given a problem instance of 10 jobs with 3 stages with 2 identical machines at each of the stages, the total number of schedules that can be generated based upon this formula is over 4.3×10^{21}.

Ongoing Work in HFS Scheduling

In addition to some of the HFS work referenced above, our current efforts are devoted to improving the scheduling of HFS problems wherein the multiple processors at a stage are not identical. When the processors are not identical, then the complexity, in terms of the number of possible schedules is determined by the following formula:

$$ \prod_{j=1}^{m} {}_{n-1}C_{M_j-1} \cdot n! $$

With 10 jobs, 3 stages, and two non-identical machines at each stage, the total number of schedules is approximately 3.5×10^{22}, which is noticeably larger than the prior example with a similar arrangement of stages and numbers of machines, but wherein the multiple machines were not identical processors.

All of the aforementioned references and discussion on the HFS and other environments have assumed that each machine can only process one job at a time (e.g., discrete parts processing) – an assumption that is safe to assume in many scheduling problems. When the environment includes processors that can process multiple jobs at a time (e.g., batch processing), then the problem complexity further increases. The HFS with a combination of discrete parts processing and batch processing machines is a problem that we are currently studying. Consider an electronics assembly operation as depicted in Figure 4.

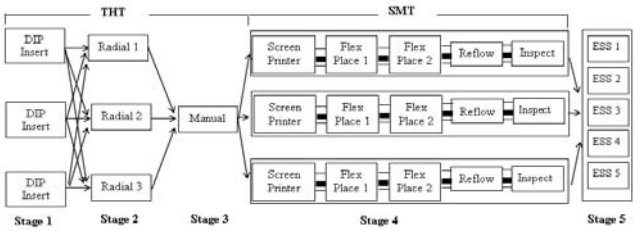

Figure 4. HFS with Discrete and Batch Processing

In the above, there are three stages of THT processing, one stage (of 3 lines) of SMT processing, and the fifth stage is a single stage of environmental stress screening (ESS). For the first four stages all boards are processed one at-a-time in each of the machines (or SMT lines). At Stage 5, each ESS chamber can process multiple boards at-a-time.

We have begun to apply common flow-shop based dispatching rules in empirical simulations of this environment (see Vishwanathan et al., 2007). We are also building theoretical foundations for the HFS with discrete and batch processors and wherein multiple processors at a stage are non-identical. As findings are discovered, they will be published. Only a few of our HFS related publications have been referenced in this work; the interested reader is welcome to contact the author for a more thorough listing of our prior efforts in the HFS environment.

ACKNOWLEDGMENTS

This work is partially supported by the Small Scale Systems Integration and Packaging Center – a NYS Center of Excellence at Binghamton University. Thanks also go out to the Integrated Electronics Engineering Center (IEEC) and Center for Advanced Microelectronics Manufacturing (CAMM) at Binghamton University. The author would also like to thank Dr. Purush Damodaran of Northern Illinois University for our ongoing collaborations in the HFS scheduling arena. For including him into his research group (’87-’93) and introducing him to the research area of production scheduling, the author would also like to personally thank Dr. J.L. Hunsucker – Industrial Engineering Professor Emeritus of the University of Houston, and Founder and President of the National Aquatic Safety Company (NASCO), Dickinson, TX.

REFERENCES

Brah, S.A., *Scheduling in a flow shop with multiple processors*. Ph.D. Thesis, University of Houston, Houston, TX, 1988.

Brah, S.A., and Hunsucker, J.L., 1991, “Branch and bound algorithm for the flow shop with multiple processors”, *European Journal of Operational Research*, 51(1), pp. 88-99.

Campbell, H.G., R.A. Dudek, M.L. Smith, “A Heuristic Algorithm for the *n* Job, *m* Machine Sequencing Problem *Management Science*, Vol. 16, No. 10, pp. B-630-B-637 (1970).

French, S., *Sequencing and Scheduling*, Prentice-Hall, 1992

Goldratt, E.M., *The Goal*, North River Press, 1984.

Gupta, J.N.D., “Two-stage hybrid flowshop scheduling problem”, *Journal of Operation Research Society***,** Vol. 39, 1988, pp. 359-364.

Gupta, J.N.D., and Tunc, E.A., 1991, “Schedules for a two-stage hybrid flowshop with parallel machines at the second stage”, *International Journal of Production Research***,** 29(7), 1489-1502.

Gupta, J.N.D., and Tunc, E.A., 1994, “Scheduling a Two-Stage Hybrid Flowshop with Separable Setup and Removal Times”, *European Journal of Operational Research***,** 77, 415-428.

Hunsucker, J.L. & Shah, J.R., “Comparative performance analysis of priority rules in a constrained flow shop with multiple processors environment”, *European Journal of Operational Research*, 72(1), 1994, 102-114.

Johnson, S.M., “Optimal two- and three-stage production schedules with setup times included,” *Naval Res. Log. Quart.*, 61-68, 1954.

Nawaz, M., Enscore, E.E., and Ham, I., "A heuristic algorithm for the m-machine, n-job flow shop sequencing problem", *OMEGA, International Journal of Management Science* 11(1), 1983.

Pinedo, M.L., *Scheduling: Theory, Algorithms, and Systems, 2nd Ed*., Prentice-Hall, 2002.

Salvador, M.S., “A Solution to a Special Case of Flow Shop Scheduling Problems”, in: S.E. Elmaghraby (ed.), *Symposium of the Theory of Scheduling and Applications***,** Springer-Verlag, 1973, New York.

Santos, D.L., Hunsucker, J.L., & Deal, D.E., “Global lower bounds for flow shops with multiple processors”, *European Journal of Operational Research*, Vol. 80, No. 1, 1995a, pp. 112-120.

Santos, D.L., Hunsucker, J.L., & Deal, D.E., “FLOWMULT: permutation sequences for flow shops with multiple processors”, *Journal of Information and Optimization Sciences*, Vol. 16, No. 2, 1995b, pp. 351-366.

Santos, D.L., Hunsucker, J.L., & Deal, D.E., “On Makespan Improvement in Flow Shops with Multiple Processors”, *Production Planning & Control*, Vol. 12, No. 3, 2001, pp. 283-295.

Vishwanathan, K., Kulkarni, N., Pachamuthu, A., Santos, D.L., and Damodaran, P., “A Comparative Study of Dispatching Rules in Hybrid Flow Shops with Discrete and Batch Processors,” *Industrial Engineering Research Conference Proceedings*, Nashville, TN, May 2007.

Wittrock, R.J., “Scheduling algorithms for flexible flow lines,” *IBM Journal of Research and Development*, 29(4), 401-412, 1985.

Yang, T. & Ignizio, J.P., “An algorithm for the scheduling of army battalion training exercises”, *Computers and Operations Research*, Vol. 14, 1987, 479-491.