Scheduling Semiconductor Device Test Operations on Multihead Testers

Tali Freed and Robert C. Leachman

Abstract—**Past attempts to devise scheduling methods for the device test operations of semiconductor manufacturing firms fail to address a significant characteristic of multiple-head test systems—the dependency of processing rates on the lots processed simultanuously on the testers. Since the problem has never been modeled accurately in the scheduling literature, feasibility and performance of previously proposed scheduling methodologies for multihead testers may not be accurately assessed. In this paper, we describe the multihead tester scheduling problem, present an enumeration solution procedure, and illustrate the problems of previously suggested tester scheduling algorithms.**

I. Introduction

SEVERAL attempts have been made to devise scheduling methods for the device test operations of semiconductor manufacturing firms. These models, however, fail to address a significant characteristic of multiple-head test systems—the dependency of processing rates on the lots processed simultanuously on the testers. The test operation lead time (changeover plus processing time), and hence, the tester's throughput, are a function of the combination of lots tested concurrently on the various test stations (heads) served by the tester central processing unit (CPU). For a multihead tester, whose CPU serves two to four heads (multiplexing testers), lot processing times can vary by a factor of three or four, depending on the lots processed concurrently. Lot lead times are fixed only for single head testers, which are used mainly for devices with relatively long test times.

Good scheduling methods for the test area are crucial to the firm's profitability. Due to the high cost of testing equipment, test floor capacity is limited, which often renders the test stage of the manufacturing process a bottleneck, at least for the back end subprocess. Also, since customization decisions are typically nonreversible, companies hold common component inventories upstream at the die bank and then customize them to order in assembly and test. This strategy is feasible only if product lead-time through assembly and test is short.

For various reasons processing priorities vary widely (e.g., upstream yield and lead time uncertainties, profit margins, customer and contract types). Therefore, it is important to devise test scheduling methods with objectives that reflect relative priorities.

References [1] and [2] describe a variety of semiconductor device testing environments and analyze their scheduling complexity. Based on this analysis, the most general and complicated scheduling problem arises in the multihead tester environment. Since the problem has never been modeled accurately in the scheduling literature, the feasibility and performance of previously proposed scheduling methodologies for multihead testers may not be accurately assessed. In this paper, we describe the multihead tester scheduling problem, present an enumerative solution procedure, and illustrate the problems of previously suggested tester scheduling algorithms.

II. Literature Review

The developments in the area of planning and scheduling semiconductor test operations started a decade ago. Prior to that, the equipment-intensive front end of the semiconductor manufacturing process had drawn much more academic attention than the relatively low-investment back end operations. However, the recent increase in device complexity has led to the development of complex capital-intensive test systems and to the necessity of developing efficient strategies for planning and scheduling the test operations.

Previous research on various aspects of planning and scheduling in semiconductor manufacturing can be classified into three major categories: 1) performance evaluation methods; 2) production planning models; and 3) shop-floor control techniques. The reader is referred to [3] and [4] for a more comprehensive discussion of these methodologies.

The area of shop-floor control of semiconductor manufacturing operations can also be classified into three major categories: 1) dispatching rules and input regulation strategies; 2) optimal control and knowledge-based systems; and 3) deterministic scheduling algorithms. The reader is referred to [4] for a detailed review of models that belong to the first two categories.

Most deterministic scheduling algorithms for semiconductor manufacturing have been designed for wafer fab applications and are not applicable to the fundamentally different test operations (e.g., [5]–[9]). Scheduling test operations has been the subject of a series of papers [10]–[17], which focus on finding good heuristics for solving a dynamic real-time scheduling problem. The test area in most of these papers is modeled as a job shop, with precedence constraints and deterministic lead times. Some of the papers also consider the sequence-dependency of setup times. References [17]–[19] focus on equipment and hardware requirements. References [17] and [18] use integer programming with Lagrangian relaxation to

solve the scheduling problem, and [19] uses Petri nets. In this paper, however, we model a process complexity that has not been modeled in the past. Multiple-head testers, which are common in the industry, have the unique characteristic that their processing times depend on tester configuration. We also present an enumerative solution technique, and we demonstrate that the performance of existing tester scheduling methods may lead to infeasible or inferior solutions when applied to the multiple head environment.

III. Description of the Single, Multihead Tester Scheduling Problem

The single multihead tester scheduling problem is to maximize the total value of the tester throughput in the time horizon T (e.g., a shift or a day), given a value (reflecting relative priority) for each test operation on a lot of devices (lot operation) and preventive maintenance activities. Process characteristics modeled include lot operation precedence constraints, configuration-dependent processing rates, and sequence-dependent changeover times.

In this section we explain the problem in detail. Due to its length, the mathematical programming formulation of the problem is not presented here. It can be found in [2].

A. Lot Operation and Maintenance Values

Most semiconductor companies assign lot priorities based on their lateness and "destination." For example, "hot" (high priority) lots are typically lots that were promised to customers and are already late for their due date (tardy make-to-order lots), while low priority lots may be on-time make-to-stock lots. Throughout this paper, we assume that each lot operation has an associated value, which is known at the beginning of the planning horizon and can be modified between time horizons. The value of each device is equal to the lot operation value divided by the lot size. Value determination methods are beyond the scope of this work, and we refer the interested reader to [20]–[22] for approaches to product value estimation. We suggest that the value should reflect factors such as expected financial contribution, lot "destination" (inventory or outstanding customer order), critical ratio (time until due date divided by expected processing duration), and resource consumption.

Since the value factor assigned to each lot captures its relative importance to the firm, a natural scheduling objective is to maximize the total value of lots processed over the planning horizon T. Most semiconductor test facilities operate for two or three shifts per 24 hours. Thus, it may be natural to solve the scheduling problem for each shift. If lot lead times are relatively long, it may be more reasonable to solve the problem on a daily or even weekly basis. A rolling horizon approach can also be used.

Existing tester scheduling methods fail to incorporate preventive maintenance (PM) into the scheduling methodology. PM activities may be frequent and time consuming; thus, PM is incorporated into the multihead tester scheduling problem as follows.

A maintenance start value is assigned to each head at the beginning of T, and a value increase factor determines the rate at which the value increases linearly with time during T. When the PM is performed, the cumulative value at its start time is added to the cumulative value of the schedule. This is an "incentive" system for performing PM activities around the time they are due, leaving some flexibility in the choice of the exact timing. To avoid performing unnecessary PM when the heads are idle (since according to the above logic low value is better than the zero added value of the idle state) we set a maintenance lower bound, below which PM cannot be performed.

B. Lot Operation Precedence Constraints

Semiconductor devices may be tested at one or several temperatures ([1]–[4]). In addition to temperature-based tests, erasable programmable memory devices may undergo before-erase and after-erase tests. Precedence constraints among lot operations are common. When hot or cold tests are more time consuming, it would typically be preferable to perform a room-temperature test first and yield out the scrap. The sequence of programming-based test operations is also significant.

C. Sequence-Dependent Changeover Times

Lot changeover times in semiconductor testing can range from a few minutes to several hours and be of the same order of magnitude as the lot processing times ([1], [2], [10], [11], [14]).

D. Configuration-Dependent Processing Rates

Each lot operation can be characterized by its device test time and handling time. The device test time is the time it takes the CPU to test the functionality of the device. The handling time is the minimal time between tests of consecutive devices of the same lot. A detailed description of tester mechanics and their effect on test and handling times can be found in [2].

Device test times range from a few seconds to a few minutes, depending on the complexity of the device. The device handling time is often of the same order of magnitude. A scenario in which a device handling time is longer than its test time is not uncommon, especially for simple and "mature" devices for whom the test is particularly short.

On a single-head tester, the lot processing (testing) time can be simply calculated as the product of the device test plus handling time and the lot size. However, a multihead system can have two to four heads, and each head can either process a lot, undergo maintenance, undergo changeover, or be idle due to lack of work. In such a tester, the CPU tests a single device from each of its processing heads in each cycle. The CPU approaches the heads sequentially, skipping the nonprocessing heads. However, the CPU has to wait if it arrives back at a testing head, but the device it is supposed to test next has not yet completed its handling. Thus, if the device handling time on any head is greater than the sum of the test times on the other heads, the CPU will incur idle time. A CPU is particularly prone to incur such idle time when some of its heads are undergoing changeover or maintenance, since

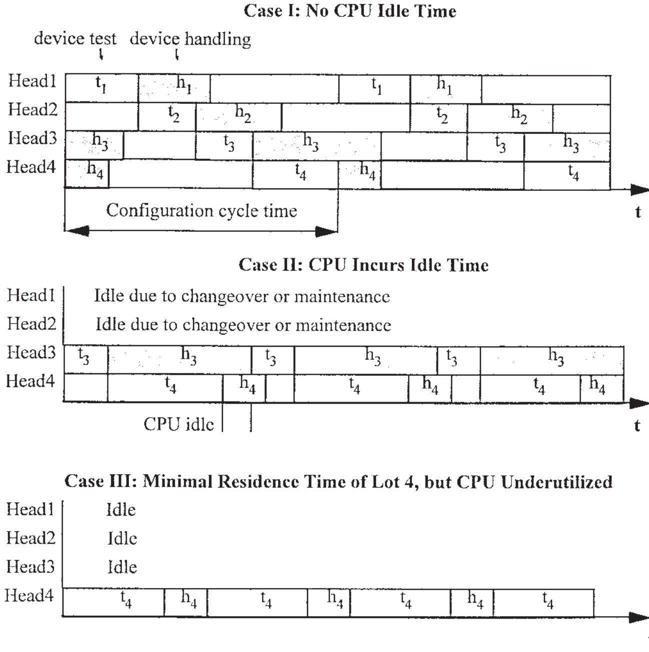

Fig. 1. Sample test configurations sequences.

in these cases the number of testing heads is reduced, and the probability that one of the handling times will dominate the total testing time is increased. Thus, the multihead tester scheduling problem is to select the optimal sequence of tester configurations during T. The selected configurations determine the tester throughput.

Note that if the sum of the test times on the other testing heads is greater than the lot handling time, the lot processing time will be longer than the minimum time, since each device will have to wait for its test. In this paper, however, our major concern is CPU idle time, which reflects production efficiency, not the individual lot processing time.

Fig. 1 illustrates the operation of a four-head test system in three cases: Case I, in which there is no CPU idle time; Case II, in which the CPU incurs idle time; and Case III, in which lot processing time is minimal but the system CPU is underutilized (the idle time per cycle is h_4—the handling time of lot 4).

In order to decrease the test time, some testers are capable of testing several devices in parallel on the same head (typically two or three), resulting in a shorter test time per unit. For example, if the test time per device when tested by itself is 3 s, two devices in parallel may take 4 s to test, and three devices may take 4.5 s.

It is important to distinguish between multihead testing (mutiplexing) and parallel testing of several devices on the same head. In parallel testing the parallel-tested devices must be identical and must be loaded and unloaded together onto the same head. In multihead testing, each head can test a different device type and is independent of the other heads in terms of loading and unloading its devices.

From a scheduling perspective, parallel testing of a lot of devices can be viewed as testing a smaller lot of devices, with a longer test time per device. We therefore assume throughout this paper that test time and lot size data are preadjusted to the parallel testing case if applicable.

The configuration cycle time is the time it takes the CPU to complete a testing cycle of a single device from each of the processing heads (see Fig. 1). The throughput of a configuration is the number of cycles that took place over the duration of the configuration, which is equivalent to the number of units tested on each processing head. The configuration value, which is added to the cumulative schedule value at the end of the configuration, is the total value of the activities that carry value and were performed on the heads, i.e., processing and maintenance. If the duration of the configuration is shorter than the duration of a maintenance activity or lot operation processing then the relative value fractions are summed up to obtain the configuration value.

The problem of maximizing the total configuration value during T is clearly NP-hard. Even without the additional complexity of the configuration-dependent processing times, due to the sequence-dependency of the setup times the problem can be interpreted as a special case of the Traveling Salesman Problem. In particular, this problem is a special case of the Orienteering Problem for K agents ([1], [2]), which is shown to be NP-hard in [23].

IV. Enumeration Program for the Single Multihead Tester Scheduling Problem

We use an enumeration program coded in C (approximately 2000 lines of code) to solve the problem to optimality. The program receives the tester initial conditions—the configuration at the end of the previous T. The activities on the heads are then continued in the current T. When an activity is completed, the program calculates the current time and the cumulative value of the work performed up to that point and develops all the potential choices for the next activity that can be selected at that point. For each choice, the program then updates the configuration. The program continues developing the solution spectrum, which can also be viewed as a decision tree, with branches starting at each decision point (node, hereafter), until the end of T is reached for all the branches. At that point, a comparison of the total cumulative value of the branches determines the optimal branch—the optimal schedule. If management wishes to receive several good schedules to choose from, the required number of schedules can be specified.

The program develops the schedule tree based on a depth first search (DFS) logic, which implies that the program develops complete branches one after another, as opposed to developing "layers" of nodes, each layer corresponding to a stage of the enumeration program. This strategy is advantageous since it allows for the termination of the program when the tree size or the number of fully developed branches exceeds limitations induced by computation time or memory requirements. The program then returns the best solution(s) achieved, as opposed to reaching the limit without any (or

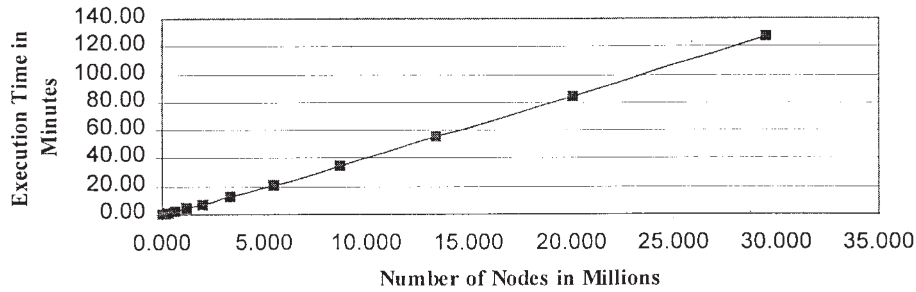

Fig. 2. Execution time versus number of nodes.

TABLE I
DATA SET

Factor	Range	Distribution	Average
Number of lots	4 - 20		
Number of product types	3		
Number of operations per lot	1 - 3		
Lot size	1400 - 2600	Uniform	2000
Test time per unit	1.2 - 3.6 seconds	Uniform	2.4
Handling time per unit	1.2 - 2.4 seconds	Uniform	1.8
Changeover time	5 - 55 minutes	Uniform	30

a sufficient number of) complete schedules. At each decision node, the lot operation or maintenance activity selected to be branched into next is the one which carries the highest value (greedy selection).

V. Performance Analysis of the Enumeration Program

The enumeration program was developed to run on a PC and a workstation. A 100 MHz Pentium microprocessor PC was used for the performance analysis. The program was applied to a set of randomly generated examples, with similar characteristics to industrial data (Table I).

We shall use this data to get a rough estimate of the tester capacity in an 8 h shift. The average configuration cycle time for three processing heads would be 7.2 s, and the average lot lead time 4.5 h (multiplying configuration cycle time by the average lot size and adding thirty minutes of changeover.) For two processing heads, the average configuration cycle time is 4.8 s, leading to an average lot lead time of 3.17 h. The average lead time per lot operation is, therefore, approximately four hours, i.e., on average two lot operations can be completed on each head during an 8 h shift. Twenty lots represent workload for approximately five shifts—a common situation in capacity-tight test facilities.

The program execution time was found to be linearly correlated to the number of nodes, with approximately 0.25 s required to generate 1000 nodes (see Fig. 2). Current technology (500 MHz microprocessor) would probably require approximately 50 μs per node.

The number of nodes is mainly a function of the number of lots to be processed. Table II presents the average number of nodes created for four, nine, and ten single-operation lots and their respective execution times.

TABLE II
NUMBER OF NODES AND EXECUTION TIME VERSUS NUMBER OF SINGLE-OPERATION LOTS

Number of lots	4	9	10
Number of nodes	225	3,424,339	34,243,401
Execution time	0.06sec	14.2min	2.3hrs

The number of operations per lot also affects the resulting tree size. A detailed analysis and formulae for the number of nodes as a function of the number of lots and operations per lot can be found in [2].

Clearly, the computation time may be prohibitively long for problems with large number of lot operations. Thus, a rolling horizon approach, in which a complete five-shift schedule would be generated every shift, but only a single-shift portion of it would be implemented, may be impractical. Instead, since the shift duration is much shorter than the makespan of a complete schedule, branches can be truncated by the end of the shift, resulting in a manageable problem.

Fig. 3 shows the number of nodes in a tree truncated by the end of an 8 h shift (truncated tree, hereafter) as a function of the number of lots (with two operations per lot). As can be observed, 13 lots result in approximately 10^6 nodes, or 4 m of execution time. Current technology may speed up the execution time of this size problem to less than 1 min.

In addition to execution time, another critical measure of the program performance is the computer memory required. On average, each generated node consumes approximately 1 KB, limiting the tree size developed on a 1 GB computer to 10^6 nodes, if all of the generated nodes are maintained in the memory. A memory-saving technique which involved the elimination of inferior branches removed this concern [2]. Fig. 4 shows the maximal memory utilization of the program (peak utilization during the program run) for varying number of nodes.

Using the combination of DFS and greedy branching proved to be powerful. For every problem in the data set, the optimal solution was reached within the first ten schedules created (within 5 min of execution time). Nevertheless, this performance, however promising, cannot be guaranteed.

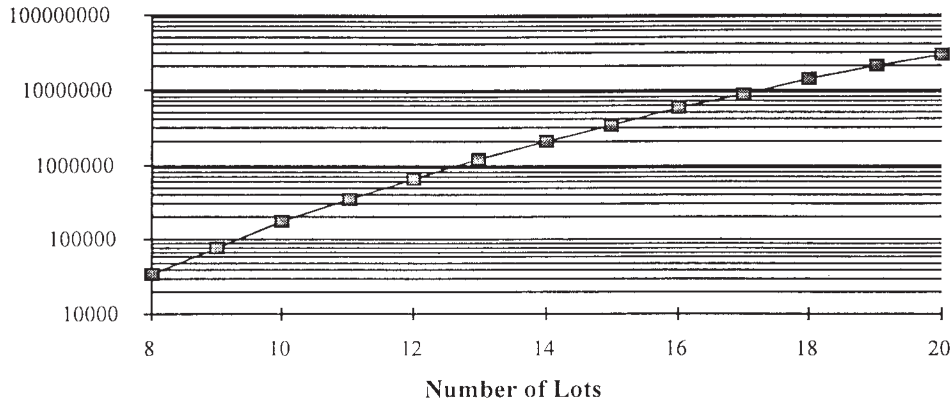

Fig. 3. Number of nodes in a truncated tree versus number of lots with two operations each.

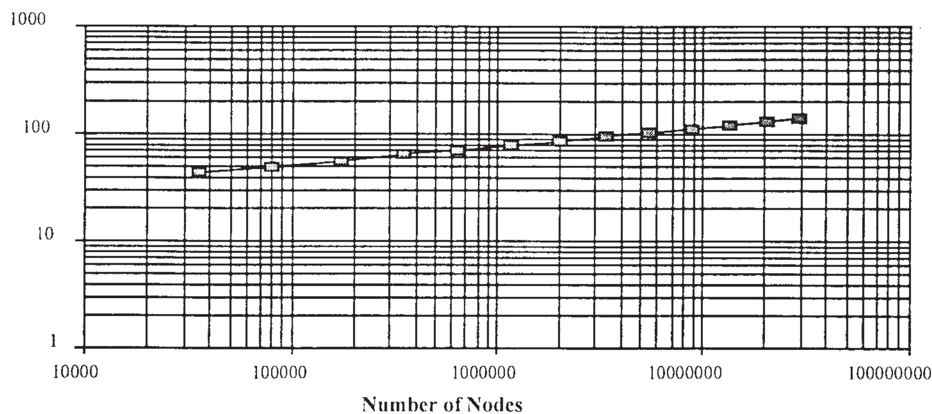

Fig. 4. Maximal memory utilization in KB for varying number of nodes.

VI. The Multihead Multiple Tester Scheduling Problem

A. Description of the Multihead Multiple Tester Scheduling Problem

Depending on the product mix, a semiconductor test facility would typically consist of several types of testers, each type capable of testing a subset of the products. The test-floor would typically be divided into "bays," which are groups of identical or similar testers. The bays perform test operations on disjoint sets of products. Thus, the test-floor scheduling problem can typically be easily decomposed into several disjoint bay scheduling problems. However, since the testing capabilities of testers in the same bay may have complete or partial overlap, determining efficient production planning techniques for this situation is not trivial, as discussed in [27].

The multihead multiple tester scheduling problem is to determine the allocation of lot operations to testers and the sequence of configurations on each tester during the shift. The problem is clearly NP-hard, since it is a generalization of the multihead single tester scheduling problem. Decomposition algorithms for parallel independent testers have been proposed in the past (e.g., [15], [16]), but are not applicable to our problem since the head interdependency is not considered.

B. Extending the Enumeration Program for the Multiple Tester Scheduling Problem

Using the enumeration program to solve the multihead multiple tester scheduling problem requires the extension of the state-space representation to include the status of all the heads across all the testers. The set of lot operations to be processed is greater since it consists of the workload of all the testers combined. The specifications of each lot operation should include tester compatibility information.

As demonstrated in Fig. 5, the addition of a fifth head to the tester increased the number of nodes generated in the truncated tree from approximately 4 million to approximately 30 million nodes, and each additional head adds approximately an order of magnitude to the number of nodes. As can be expected from a complete enumeration technique, the extension of the enumeration program to the multiple tester scheduling problem is likely to be practical only for small problems.

VII. The Shortcomings of Existing Tester Scheduling Methods

Several scheduling algorithms for semiconductor device test operations have been published in the literature. References [10], [11], and [14] focus on scheduling a single machine with sequence-dependent changeover times and precedence

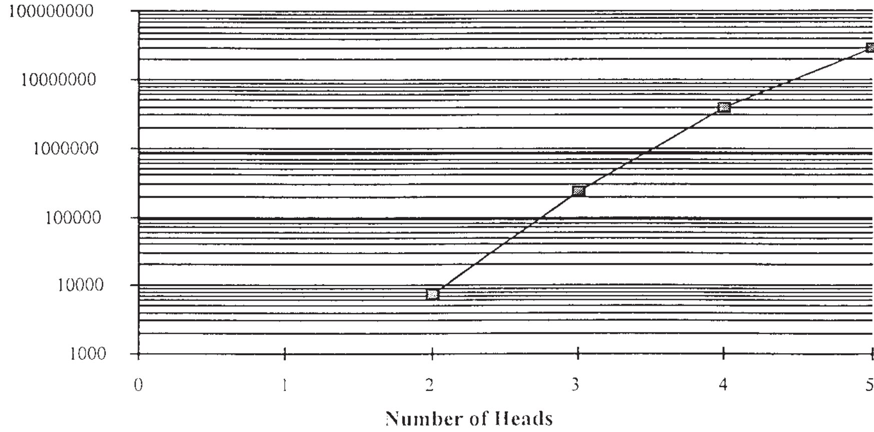

Fig. 5. Number of nodes generated in a truncated tree versus number of heads.

TABLE III
EXAMPLE DATA

Lot	Units	Test-time	Handling-time	Due-date
1	2000	2.4	1.2	0
2	2000	2.4	1.2	0
3	2000	2.4	1.2	0
4	1500	2.4	1.2	18,000
5	1500	2.4	1.2	18,000
6	1000	2.4	1.2	18,000
7	1000	1.2	2.4	26,000
8	1000	1.2	2.4	26,000
9	1000	1.2	2.4	26,000

TABLE IV
SCHEDULE OF PREVIOUS METHODOLOGY

Config.	Cycle-time	Units	Duration	Cum. time	head status
1	7.2	2000(x3)	14,400	14,400	PPP
2	0.0	0	2,400	16,800	CCC
3	7.2	1000(x3)	7,200	24,000	PPP
4	4.8	500(x2)	2,400	26,400	PPC
5	3.6	667(x1)	2,400	28,800	CCP
6	6.0	333(x3)	2,000	30,800	PPP
7	3.6	667(x2)	2,400	33,200	PPI
8	n/a	n/a	n/a	n/a	I I I

TABLE V
SCHEDULE OF NEW METHODOLOGY

Config.	Cycle-time	Units	Duration	Cum. time	head status
1	6.0	1000(x3)	6,000	6,000	PPP
2	4.8	500(x2)	2,400	8,400	PPC
3	4.8	500(x2)	2,400	10,800	PCP
4	4.8	500(x2)	2,400	13,200	CPP
5	3.6	667(x2)	2,400	15,600	PPC
6	6.0	333(x3)	2,000	17,600	PPP
7	4.8	500(x2)	2,400	20,000	CPP
8	4.8	500(x2)	2,400	22,400	PCP
9	6.0	667(x3)	4,000	26,400	PPP
10	3.6	333(x2)	1,200	27,600	PPI
11	n/a	n/a	n/a	n/a	I I I

constraints among operations on the same lot, while [12], [13], [15], and [16] present methodologies for scheduling multiple machines. Some of these methodologies are modified versions of the shifting bottleneck heuristic procedure for job-shop scheduling introduced in [24]. In this procedure, a network representation is used to capture operation precedence and tester dependence relationships. The sequence of operations for each tester is determined by the Extended Jackson sequencing heuristic ([25], [26]). This procedure may be applied together with a heuristic improvement procedure. Reference [16] uses the rolling horizon algorithms described in [14] and [15] to schedule the individual testers. However, all these algorithms assume that the processing time of lot operations are known and independent of tester configuration. Some also assume that once the assignment of lot operations to testers is determined, all operations of lot i must be performed on the same tester. The algorithms treat each head as a separate tester, assume that each lot has a due date, and solve the scheduling problem for two objective functions: 1) minimum maximal lateness $(\min L_{\max})$ and 2) minimum number of tardy lots $(\min N_t)$.

Although these performance measures have a customer service orientation, they fail to address throughput and priorities. These models also have two major shortcomings if applied to the multihead tester scheduling problem: 1) the solutions would probably be infeasible since the true operation durations are configuration dependent and 2) the solutions may be suboptimal due to the preallocation of lot operations to heads. The following example demonstrates these concerns. It also demonstrates the advantage of using multihead testers over single-head testers.

A. Example: Single-Operation Lots Processed on a Three-Head Tester

In this example, the changeover time is 2400 s. Test times, handling times, and due dates are given in seconds.

According to the algorithms suggested in [10], [11], and [14], each lot should be preassigned to a head. In order to balance the workload among the heads, we assume that lots 1, 4, 7 are assigned to head 1, lots 2, 5, 8 are assigned to head 2, and lots 3, 6, 9 are assigned to head 3. Assuming that no changeover is required for the first lot, these algorithms ("pre-

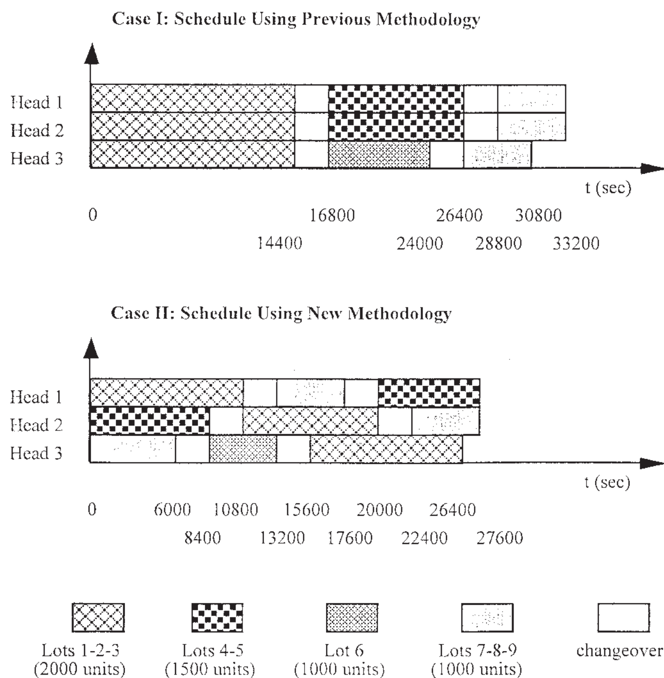

Fig. 6. Gantt charts for the schedules of the previous and the new methodologies.

vious methodology," hereafter) would result in the schedule described in Table IV. In contrast, the optimal schedule for the tester as determined by the enumeration program ("new methodology," hereafter) is summarized in Table V.

The Gantt chart for both schedules is presented in Fig. 6.

This example demonstrates that the previous methodology may generate very inefficient schedules when applied to a multihead tester environment. The total CPU idle time accumulated in the previous methodology is 4800 s (or 1.3 h). This idle time results from the following situations.

1) All heads undergo changeover simultaneously (configuration 2).
2) A single head is processing while other heads undergo changeover (configuration 5).
3) Two heads are processing and one lot's handling time is longer than the other lot's test-time (configuration 7).

In the new methodology, the CPU does not have idle time at all. The schedule makespan of the previous methodology is 33 200 s, i.e., 9.2 h, as compared with the makespan of the new methodology, which is only 27 600 s, or 7.66 h.

VIII. Conclusion

In this paper, we described the semiconductor test-floor scheduling problem in detail, capturing the unique structure and characteristics of the multihead device testing process. We developed an enumeration program for solving the problem and illustrated the infeasibility and inferiority of solutions generated by algorithms that assume configuration-independent processing rates.

As expected, the enumeration program can find the optimal solutions for relatively small single-tester and multiple-tester problems. However, it can serve as a benchmark, generating optimal solutions against which heuristics may be compared.

References

[1] T. Freed (Carmon) and R. C. Leachman, "A taxonomy of semiconductor device test scheduling problems," *Working Paper*, 1999.

[2] T. Carmon (Freed), "Production planning and scheduling for semiconductor device testing" Ph.D. dissertation, Dept. Ind. Eng. Operations Res., Univ. California, Berkeley, CA, 1995.

[3] R. Uzsoy, C. Y. Lee, and L. A. Martin-Vega, "A review of production planning and scheduling models in the semiconductor industry part I: System characteristics, performance evaluation and production planning," *IIE Trans. Scheduling Logistics*, vol. 24, no. 4, pp. 47–61, 1992.

[4] R. Uzsoy, C. Y. Lee, and L. A. Martin-Vega, "A review of production planning and scheduling models in the semiconductor industry part II: Shop floor control," *IIE Trans. Scheduling Logistics*, vol. 26, pp. 44–55, 1994.

[5] S. C. Graves, H. C. Meal, D. Stefek, and A. H. Zeghmi, "Scheduling of re-entrant flow shops," *J. Oper. Manage.*, vol. 3, no. 4, pp. 197–207, 1983.

[6] C. R. Glassey and M. G. C. Resende, "Closed-loop job release control for VLSI circuit manufacturing," *IEEE Trans. Semiconduct. Manufact.*, vol. 1, no. 1, pp. 36–46, 1988.

[7] W. Kubiak, S. X. C. Lou, and Y.-M. Wang, "Mean flow time minimization in re-entrant job shops with hub," *Oper. Res.*, vol. 44, pp. 764–776, 1996.

[8] J. Ou and L. M. Wein, "Dynamic scheduling of a production/inventory system with by-products and random yield," *Manage. Sci.*, vol. 41, no. 6, pp. 1000–1017, 1995.

[9] J. H. Ahmadi, R. H. Ahmadi, S. Dasu, and C. S. Tang, "Batching and scheduling jobs on batch and discrete processor," *Oper. Res.*, vol. 40, pp. 750–763, 1992.

[10] R. Uzsoy, L. A. Martin-Vega, C. Y. Lee, and P. A. Leonard, "Production scheduling algorithms for a semiconductor test facility," *IEEE Trans. Semiconduct. Manufact.*, vol. 4, pp. 270–280, 1991.

[11] R. Uzsoy, C. Y. Lee, and L. A. Martin-Vega, "Scheduling semiconductor test operations: Minimizing maximum lateness and number of tardy jobs on a single machine," *Naval Res. Logistics*, vol. 39, pp. 369–388, 1992.

[12] C. Y. Lee, L. A. Martin-Vega, R. Uzsoy, and J. Hinchman, "Implementation of a decision support system for scheduling semiconductor test operations," *Electron. Manufact.*, vol. 3, pp. 121–131, 1993.

[13] I. M. Ovacik and R. Uzsoy, "A shifting bottleneck algorithm for scheduling semiconductor testing operations," *Electron. Manufact.*, vol. 2, pp. 119–134, 1992.

[14] I. M. Ovacik and R. Uzsoy, "Rolling horizon algorithms for a single-machine dynamic scheduling problem with sequence-dependent setup times," *Int. J. Prod. Res.*, vol. 32, pp. 1243–1263, 1994.

[15] I. M. Ovacik and R. Uzsoy, "Rolling horizon procedures for dynamic parallel machine scheduling with sequence-dependent setup times," *Int. J. Prod. Res.*, vol. 33, no. 11, pp. 3173–3192, 1995.

[16] I. M. Ovacik and R. Uzsoy, "Decomposition methods for scheduling semiconductor testing facilities," *Int. J. Flexible Manufact. Syst.*, vol. 8, pp. 357–388, 1996.

[17] T. R. Chen, T. S. Chang, C. W. Chen, and J. Kao, "Scheduling for IC sort and test with preemptiveness via Lagrangian relaxation," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, no. 8, pp. 1249–1256, 1995.

[18] T. R. Chen and T. C. Hsia, "Scheduling for IC sort and test facilities with precedence constraints via Lagrangian relaxation," *Manufact. Syst.*, vol. 16, no. 2, pp. 117–128, 1997.

[19] H. H. Xiong and M. C. Zhou, "Scheduling of semiconductor test facility via Petri nets and hybrid heuristic search," *IEEE Trans. Semiconduct. Manufact.*, vol. 11, no. 3, pp. 384–393, 1998.

[20] T. E. Morton, S. R. Lawrence, S. Rajagopolan, and S. Kekre, "SCHED-STAR—A price-based shop scheduling module," *Manufact. Oper. Manage.*, vol. 1, no. 1, pp. 131–181, 1988.

[21] P. S. Ow and T. E. Morton, "The single machine early/tardy problem," *Manage. Sci.*, vol. 35, no. 2, pp. 177–191, 1989.

[22] W. W. Weng and R. C. Leachman, "An improved methodology for real-time production decisions at batch-process work stations," *IEEE Trans. Semiconduct. Manufact.*, vol. 6, no. 4, pp. 219–225, 1993.

[23] B. L. Golden, L. Levy, and R. Vohra, "The orienteering problem," *Naval Res. Logistics*, vol. 34, pp. 307–318, 1987.

[24] J. Adams, E. Balas, and D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Manage. Sci.*, vol. 34, no. 3, pp. 391–401, 1988.

[25] C. N. Potts, “Analysis of a heuristic for one machine sequencing with release dates and delivery times,” *Oper. Res.*, vol. 28, pp. 1436–1441, 1980.

[26] J. Carlier, “The one-machine sequencing problem,” *European J. Oper. Res.*, vol. 11, pp. 42–47, 1982.

[27] R. C. Leachman and T. F. Carmon, “On capacity modeling for production planning with alternative machine types,” *IIE Trans. Scheduling and Logistics*, vol. 24, no. 4, pp. 62–72, 1992.