AN ABSTRACT OF THE THESIS OF

Tony R. Fountain for the degree of Doctor of Philosophy in Computer Science presented on August 21, 1998. Title: Just Enough Die-Level Test: Optimizing IC Test via Machine Learning and Decision Theory.

Abstract approved: Redacted for Privacy

Thomas G. Dietterich

This research explores the hypothesis that methods from decision theory and machine learning can be combined to provide practical solutions to current manufacturing control problems. This hypothesis is explored by developing an integrated approach to solving one manufacturing problem - the optimization of die-level functional test.

An integrated circuit (IC) is an electronic circuit in which a number of devices are fabricated and interconnected on a single chip of semiconductor material. According to current manufacturing practice, integrated circuits are produced en masse in the form of processed silicon wafers. While still in wafer form the ICs are referred to as dice, an individual IC is called a die. The process of cutting the dice from wafers and embedding them into mountable containers is called packaging.

During the manufacturing process the dice undergo a number of tests. One type of test is die-level functional test (DLFT). The conventional approach is to perform DLFT on all dice. An alternative to exhaustive die-level testing is selective testing. With this approach only a sample of the dice on each wafer is tested. Determining which dice to test and which to package is referred to as the “optimal test problem”, and this problem provides the application focus for this research.

In this study, the optimal test problem is formulated as a partially observable Markov decision model that is evaluated in real time to provide answers to test questions such as which dice to test, which dice to package, and when to stop testing. Principles from decision theory (expected utility, value of information) are employed to generate tractable decision models, and machine learning techniques (Expectation Maximization, Gibbs Sampling) are employed to acquire the real-valued parameters of these models. Several problem formulations are explored and empirical tests are performed on historical test data from Hewlett-Packard Company. There are two significant results: (1) the selective test approach produces an expected net profit in manufacturing costs as compared to the current testing policy, and (2) the selective test approach greatly reduces the amount of testing performed while maintaining an appropriate level of performance monitoring.

Just Enough Die-Level Test:

Optimizing IC Test via Machine Learning and Decision Theory

by

Tony R. Fountain

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented August 21, 1998
Commencement June 1999

Doctor of Philosophy thesis of Tony R. Fountain presented on August 21, 1998

APPROVED:

Redacted for Privacy

Major Professor, representing Computer Science

Redacted for Privacy

Chair of Department of Computer Science

Redacted for Privacy

Dean of Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for Privacy

Tony R. Fountain, Author

Acknowledgements

A number of people contributed, either directly or indirectly, to the research described in this dissertation. Three people deserve special recognition for their critical contributions.

Professor Tom Dietterich of Oregon State University served as thesis advisor. Dr. Dietterich provided direction and inspiration through all phases of this project, from problem definition to evaluation. It was both an honor and a pleasure to work with him on this project.

Mr. Bill Sudyka of Hewlett-Packard Company served as domain expert and technical consultant. Mr. Sudyka provided the manufacturing details and the test data that made this project possible. His enthusiasm, commitment and patience were unwavering during our exploration of IC manufacturing. His assistance far exceeded the call of duty.

Professor Bruce D'Ambrosio of Oregon State University served as initial project advisor and facilitator. Dr. D'Ambrosio provided funding and direction throughout most of my graduate studies. His assistance was instrumental in my attending Oregon State University and in determining the direction of my graduate studies. His generous support was essential to the successful completion of this project.

The benefits derived from my relationships with these individuals extend beyond the completion of this project. My gratitude extends far beyond this simple acknowledgement.

TABLE OF CONTENTS

Page

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

Just Enough Die-Level Test:
Optimizing IC Test via Machine Learning and Decision Theory

1. INTRODUCTION

This research explores the hypothesis that methods from decision theory and machine learning can be combined to provide practical solutions to current manufacturing control problems. The control of a modern computer-integrated manufacturing line is a formidable task. The process complexities, real-time pressures, and the economic bottom-line all combine to make process analysis and control difficult. Intelligent decisions require accurate information and rational decision policies. Although high-speed data networks are capable of providing volumes of data on process parameters, the translation of data into information and the application of this information to process control remain a challenge due to the complexity of the task. One consequence of this complexity is that the real-time application of process data is often ignored and naïve sub-optimal control policies are implemented. At times it is simply easier to treat all manufactured items the same rather than detecting and responding to individual differences. Recent developments in the fields of computational decision theory and machine learning offer attractive theoretical tools to address complex analysis and control problems. Computational decision theory offers efficient formulations of optimal decisions, and machine learning offers efficient data analysis. Taken together these methods provide a principled approach to solving some of the complex control problems encountered in a computer-integrated manufacturing environment. This research extends that body of knowledge by developing an integrated approach to solving one such problem: the real-time optimization of die-level functional test.

1.1 Integrated Circuit Manufacturing

An integrated circuit (IC) is a circuit in which a number of elements are fabricated and interconnected on a single chip of semiconductor material (Van Zant, 1997; Zorich, 1990). According to current manufacturing practice integrated circuits are produced en masse in the form of processed silicon wafers. A single wafer can yield up to several thousand ICs. While still in wafer form, the ICs are referred to as dice; an individual IC is called a die. The process of cutting the dice from the wafers and embedding them into mountable containers is called packaging.

During the manufacturing process the dice undergo a number of tests. One type of test is die-level functional test (DLFT). During DLFT, electrical signals are fed to each individual die, and the outputs are measured to determine whether the die is operating correctly. The conventional approach is to perform DLFT on all dice while they are still in wafer form. Once this testing is complete, the individual dice are cut from the wafers, sorted, then either packaged to yield ICs or are scrapped. Those that are packaged are then tested again. These final tests are similar to those that were performed in DLFT. Die-level functional tests are expensive (testers, personnel, production time), may cause damage to the dice, and in some cases may be unnecessary, since the ICs are tested again prior to sale. An optimal testing approach would minimize the number of functional tests while maximizing net profit.

1.2 Optimizing IC Test

An alternative to the exhaustive test policy for wafer test is a selective test policy. Under a selective test policy, wafers are tested only to the degree needed to make informed package decisions. The assumption is that die defects exhibit a spatial distribution that can be predicted by testing only a sample of the dice on any wafer. This assumption is exploited by the testing policy in order to make package decisions for all dice on a wafer while only testing a sample of the dice on that wafer. Thus, the decision to package a die is based on the expected utility of the package test result for that die, rather than on the observed functional test result for that die. In other words, a die is packaged if it is expected to pass package test and produce a profit regardless of whether it was functionally tested while in wafer form. Since all packaged dice are functionally tested after packaging, there is no chance that a bad die will be shipped to a customer. Thus the selective test approach attempts to eliminate unnecessary functional tests. The selective test approach is distinctive in that during wafer testing the policy is updated in real time with the observed functional test results. Thus testing behavior is customized for each wafer.

The selective test approach is based on the following four hypotheses:

1. Die defects exhibit stochastic spatial patterns.
2. Machine learning techniques can be developed to discover and capture these patterns.
3. Decision-theoretic models can be constructed to exploit these patterns through selective test policies.
4. Selective test policies can reduce the number of functional tests while improving net profits and maintaining adequate process feedback.

In this dissertation, these hypotheses are examined through the development and evaluation of a system that implements the selective test policy for die-level functional test. The overall approach can be summarized as follows:

1. The wafer test problem was formulated as a series of decisions. A flowchart was created in which the control points in the testing process were identified. Each control point corresponds to a decision in the selective test policy.

2. Statistical decision theory was employed to structure each of the decisions. This involved the following steps:

- A utility model for measuring wafer test costs and profits was developed. Profit is determined by the selling price of good ICs. The relevant costs include the functional test cost, the cost of packaging, and the cost of package test.
- The sources of uncertainty in the problem were identified. The selective test approach makes package decisions without complete knowledge of functional test results. Thus the functional test results for the untested dice are a source of uncertainty in this problem.
- Each of the wafer test decisions was expressed in terms of expected utility. Since the overall goal of the selective test policy is to maximize net profits, and since there is some uncertainty about test results, the appropriate decision rule is to act to maximize expected profits. Thus each of the wafer test decisions was formulated as an expected utility calculation.

3. Probabilistic inference techniques were used to derive computable models of the wafer test decisions. A belief network was developed to represent the stochastic model of wafer test results. Influence diagrams were developed for key control decisions. Inference algorithms were developed to perform probability and expected utility calculations.

4. Unsupervised machine learning techniques were developed to acquire the parameters to the stochastic model. The Expectation-Maximization (EM) algorithm and the Gibbs sampling algorithm were applied to the parameter estimation problem. Holdout validation was employed to evaluate alternative model structures. Performance was measured in terms of a negative log likelihood score.

5. Experimental tests of the selective test approach were performed. The trained stochastic models were combined with the utility model and embedded within a selective test control structure. The performance was evaluated in simulated wafer testing over historical data provided by Hewlett-Packard Company. The selective test policies were compared to the current exhaustive test policy, to a no-test policy, and to an optimal (Oracle) policy. Performance was measured in terms of net

profits and overall testing behavior, i.e., the number of dice tested, the number of packages, and the number of correct and incorrect package decisions. Additional tests were performed to measure the following:

- The robustness of the selective test policy to changes in utility parameters.
- The response of the selective test policy to test process problems as evidenced by abnormal wafers.
- The effects of various training policies on testing performance.
- The ability of the selective test policy to generalize to other data sets.

An overall evaluation of the system yields two significant results:

1. The selective test approach produces an expected net profit in manufacturing costs as compared to the current testing policy.
2. The selective test approach greatly reduces the amount of testing performed while maintaining an appropriate level of performance monitoring.

The details of the development and evaluation of the selective test system are provided in the remaining chapters of this thesis.

1.3 Thesis Overview

The remainder of this thesis is organized into the following chapters.

Chapter 2 describes semiconductor manufacturing in some detail. The application of mathematical models and operations research methods to IC testing is reviewed. The conventional approach to wafer testing is presented. This exposition provides the context and the details of the die-level functional test problem.

Chapter 3 presents the decision-theoretic analysis of the die-level functional test problem. The selective test approach is developed as an alternative to the conventional exhaustive test approach. The formulation of the selective test approach is accomplished by applying methods from decision theory and probabilistic inference. Belief net models are created for modeling wafer test results. Influence diagrams are developed for key wafer test decisions. Inference algorithms are developed to compute probability and expected utility values.

Chapter 4 presents the machine learning methods that were developed for generating the parameters to the stochastic models of wafer test results. The parameter estimation problem is formulated as maximum likelihood estimation with a latent-class model. The expectation-maximization (EM) algorithm and the Gibbs sampling algorithm are developed for estimating model parameters from historical test data. Results from learning experiments are presented.

Chapter 5 describes the experiments that were performed with the selective test approach. Various EM-trained and Gibbs-trained models are evaluated by simulated wafer testing. Issues of model adequacy and robustness are explored. Experimental results are presented and discussed.

Chapter 6 provides a summary of the research results and makes recommendations for future research.

2. PROBLEM DESCRIPTION

2.1 The Integrated Circuit Industry

Since its inception in the late 1950s, the integrated circuit (IC) manufacturing industry has experienced continual growth at a rate exceeding that of such high-growth industries as automobiles, telecommunications, and pharmaceuticals (Van Zant, 1997). One reason for this continual growth is the number of applications in which ICs are essential. Primary industrial consumers of ICs are computer and consumer-electronics manufacturers, the telecommunications industry, the automotive industries, and the military. In addition to traditional semiconductor consumers, engineers continue to develop new markets by inserting ICs into everything from car keys to household pets (e.g., http://www.identichip.com). There is no reason to suspect that this trend will not continue for some time.

Along with continued growth, the industry has experienced a phenomenal increase in product complexity, from the early devices containing only a few transistors to the recent release of the 400 MHz Intel Pentium II processor, which contains millions of transistors.

The evolution of circuit complexity is reflected in the names given to IC classes:

- SSI (small-scale integration): Up to 100 electronic components per chip
- MSI (medium-scale integration): From 100 to 3,000 electronic components per chip
- LSI (large-scale integration): From 3,000 to 100,000 electronic components per chip
- VLSI (very large-scale integration): From 100,000 to 1,000,000 electronic components per chip
- ULSI (ultra large-scale integration): More than 1 million electronic components per chip

In 1964, Gordon Moore predicted that the IC density would double every year. In the following 34 years the accuracy of Moore's prediction has been established and his prediction is now commonly referred to as "Moore's Law" (Maniwa, 1996; Boyd, 1997).

With every increase in IC performance, a new application develops, thus it appears that the growth of the IC market will continue for some time. Some analysts have predicted a 4Gb DRAM chip by the year 2005 (Van Zant, 1997), and a 100 billion-transistor chip by the year 2020 (Meindl, 1993). With the development of high-quality multimedia, voice processing, and other compute-intensive applications, there is little doubt that these advanced chips will find an active market.

IC manufacturing is a complex process, involving large capital investments and significant risks. It is also a high-volume, high-value operation where small changes in productivity can have a large effect on company profits. As IC performance improves, the manufacturing process grows more complex and more expensive. This is a direct result of two factors: (1) the miniaturization of components and (2) the increase in IC size. Smaller components require expensive specialized fabrication equipment and facilities. Larger ICs drive the yield requirements up, since they are bigger, more expensive, and there are fewer per wafer. Thus it is necessary to get more good dice from each wafer. A complicating factor with larger ICs is defect density and its effect on yield. Large surface areas and small components make particle contamination more likely.

In addition to IC trends, the manufacturing environment is evolving. For example, networked workstations, in situ sensors, and database servers are common components of the modern manufacturing environment. These components make the physical acts of data collection more feasible, but increase the need for principled methods to integrate evidence and evaluate control policies. In addition, the process technology is changing. For example, plasma etchers, x-ray lithography, and new generations of testers are being deployed in the manufacturing environment. All of these are expensive and thus place additional pressures on manufacturing yield. Finally, the product development environment is changing in response to pressures to minimize the time to market. These factors combine to make IC manufacturing an expensive enterprise. The cost of bringing one of the newest wafer fabrication facilities on-line can exceed $1 billion. With the increased costs of wafer fabrication comes increased pressure for efficient operation. The containment of manufacturing costs is currently the biggest problem for the semiconductor industry (Kohyama, 1994).

The concern for efficient operations has spurred the development of computer-integrated manufacturing (CIM) environments and the application of operations research and other mathematical tools to problems of manufacturing control (Lin, 1996; Cobb, et. al., 1994). The modern IC manufacturing plant could not function without some level of computer-integrated manufacturing. A key element of this environment is the statistical process control (SPC) component that gathers and analyzes manufacturing data in order to maintain process control and increase yields. Often the SPC component is the most complex program in the CIM environment, since it is required to process tremendous volumes of data and respond in real-time to correct problems and minimize downtime (Zorich, 1991). For example, in the fabrication plant considered in this study, a single wafer generates over 70,000 data values.

As the industry matures, technological advance faces the limits imposed by nature and therefore the role of efficient manufacturing operations becomes more significant. Unlike technological advances, manufacturing improvements are not limited by the physical barriers such as the speed of electrical conductance, which becomes a factor when constructing sub-micron devices. It is widely accepted that sometime early in the 21st century Moore's law will confront nature's laws of physics. At this time miniaturization will have reached its practical limits and further progress will be driven by the gradual improvement in conventional manufacturing methods or by radical new approaches such as quantum and DNA computing. Although technological advances will continue to drive the semiconductor industry for some time, the role of efficient manufacturing operations is expected to play an increasingly significant role in the industry's growth.

Another factor influencing the evolution of manufacturing practices is the globalization of the IC industry and the attendant increase in competitive pressures. The increased production among the Pacific-rim countries, in particular, has resulted in greater pressure on IC manufactures to optimize manufacturing operations in order to maintain profits. In addition, semiconductor manufacturing often involves a global value delivery system (VDS) in which manufacturing operations are distributed across continents. For example, it is common for fabrication plants in the United States to ship wafers to Asia for final processing. An efficient global VDS requires shared data and rapid international data access.

The history of IC manufacturing and the predictions for its future evolution all suggest an increased pressure on efficient operations and an attendant increase in the importance of methods for achieving this efficiency. Manufacturing trends are clear - changes in technology (e.g., linewidth miniaturization, larger wafers, new process technologies and testers) and changes to methods (e.g., global VDS) will combine to complicate the control problem, placing stricter requirements on control procedures.

The next section presents an overview of the IC manufacturing processes.

2.2 IC Manufacturing Processes

According to current manufacturing practice, integrated circuits (ICs) are fabricated en masse in the form of processed silicon wafers. A single wafer can yield up to several thousand ICs, also called chips. While still in wafer form, the ICs are often referred to as dice; an individual IC is called a die. Wafers are created in lots. A lot usually contains from 20-50 wafers (a lot is sometimes called a batch, sometimes "batch" refers to a group of lots). The wafers within a lot travel together through most processing steps. The structure of IC manufacturing gives rise to multiple levels of data granularity (Turney, 1995):

1. IC-level
2. Site-level: test areas of the wafer
3. Wafer-level
4. Lot-level

The multiple levels of granularity raise interesting issues in data analysis and modeling and present challenging problems for process engineers and managers (Turney, 1995; Ou and Wein, 1996; Albin and Friedman, 1989).

Figure 1 is a simplified schematic of the major IC manufacturing steps.

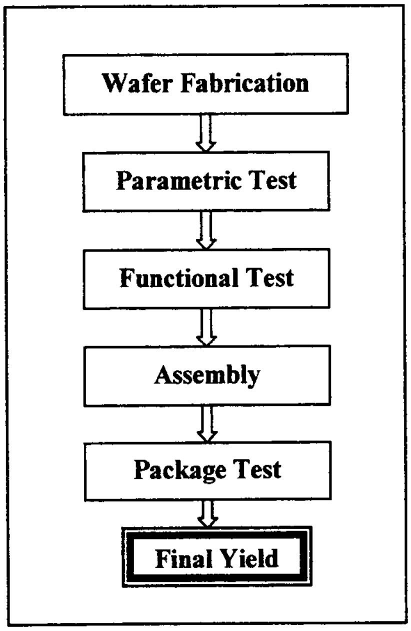


Figure 1: Major IC Manufacturing Steps

Brief descriptions of the manufacturing steps are provided below.

1. Fabrication:

   In simple terms, the fabrication process can be described as follows. Wafers undergo a series of processing steps. First the surface is coated with a thin film, either a semiconductor (e.g., silicon), dielectric (e.g., silicon dioxide), or a conductor (e.g., aluminum) (Van Zant, 1997; Zorich, 1990). Then a pattern is created on this new surface by a process referred to as masking or photolithography. Next the areas delineated by this pattern are either doped (i.e., an impurity is introduced in order to change the electrical properties) or etched (i.e., physical or chemical removal of the surface material). This description is greatly simplified since there are a variety of alternative methods for performing each of the steps, and a typical wafer requires more than 100 process steps.

2. Wafer parametric test:

   Although the above diagram depicts parametric testing as a distinct stage following wafer fabrication, in reality, parametric tests are performed throughout the fabrication process. Parametric tests typically measure physical and electrical characteristics of the wafer rather than specific characteristics of the individual dice.

   There are three general types of parametric tests (Van Zant, 1997, Zorich, 1990):

   - Device performance measures to infer process parameter control, e.g., diode leakage, voltage thresholds, resistor value tests, contact resistance, and capacitance.
   - Direct measures of physical parameters such as film thickness, width, and composition.
   - Direct measures of contamination, e.g., particle counts.

   Parametric tests are usually performed on special devices in test die or on special structures in the scribe line areas between dice (Van Zant, 1997).

3. Die-level functional test (DLFT)

   Functional testing is a distinct stage that typically occurs once the wafers are completely fabricated and the dice are completely formed and functional. Functional tests measure the operational quality of the individual dice. Electrical probes are connected to die contacts and input signals are fed to the circuit and output signals are measured. Functional testing typically simulates normal and abnormal operating conditions (e.g., high and low voltage tests). Sometimes functional testing is called final wafer probe.

   DLFT is described in more detail in section 2.5.

4. Wafer Assembly

   Wafer assembly is the process of turning wafers into packaged ICs. The individual dice are sawn from the wafers using a high-precision diamond saw. The resulting chips are mounted into packages, electrical contacts are bonded in place, and then a protective covering is added. Wafer assembly is often referred to as the package process.

5. Package test

   Once the ICs are packaged they are tested again to ensure that the packaging process was successful. It is at this stage that the integrity of the electrical contacts that were established during the packaging process is verified. Package testing also ensures that the chips were not damaged during the package process. Package tests usually repeat many of the functional tests that were performed during DLFT. Finally, the performance measures from package testing are used to sort the ICs and this determines final part disposition.

Although the above diagram only identifies final yield, there is actually a yield measure associated with each manufacturing step. At each step there is the option of removing wafers from the production line, for example, those that fail parametric test may be pulled for analysis prior to functional test. A more detailed description of yield measures is presented in the next section.

2.3 IC Manufacturing Control

The control of a typical IC manufacturing plant is dictated by a management-by-exception policy. Expectations are established, manufacturing parameters are monitored, and exceptions (i.e., unusual events) trigger responses. These responses can be automated to various degrees, from simple operator alerts to complex analyses via expert systems.

The key to an effective management policy is accurate measures of process behaviors. This is accomplished by monitoring key process parameters, including the following:

- Yields: various measures of the number of products surviving processing steps (see below). Yield sometimes refers to counts, sometimes to the fraction of surviving parts.
- Good-to-functional ratio: the ratio of the number of parts that work and meet specification limits to the number that work but do not meet specification limits. (Hansen, et al., 1997).
- Inventory levels.
- Machine downtime due to failures or maintenance.
- Cycle time - the length of time that it takes for a wafer (or wafer lot) to pass through the manufacturing process. Modern cycle times typically take from a couple of weeks to several months. (Hansen, et al., 1997).
- Parametric and functional test results.

The single most important measure of IC manufacturing success is yield (Cunningham, J. 1990). Each process step has an associated yield measure. These can be characterized as

- Y_{line}: line yield is defined as the ratio of the number of wafers in to the number out. Line yield is typically measured after functional test.
- Y_{die}: die yield is defined as the ratio of good die-per-wafer at die test to total die-per-wafer. Die yield is measured after functional test.
- $Y_{assembly}$: assembly yield is defined as the ratio of ICs that survive packaging to the total number of die-per-wafer. Assembly yield is measured after the assembly (package) step.
- $Y_{package\_test}$: package test yield is defined as the ratio of number of ICs that pass package test to the total number of ICs packaged. Package test yield is measured at the conclusion of package test.

Given these definitions, overall yield is defined as follows (Cunningham, J. 1990; Van Zant, 1997): $Y_{Overall} \equiv Y_{line}Y_{die}Y_{assembly}Y_{package\_test}$

The key goal in optimizing a production line is to simultaneously maximize yield and minimize cycle time. The standard approach to implementing a management-by-exception policy to achieve this goal is via statistical process control (SPC). Under an SPC approach, statistical limits for manufacturing parameters are determined and observed performance is compared to these limits. There are two types of limits: control and specification. Control limits delineate the normal operating range of process parameters and are usually set by analysis of manufacturing data. Specification limits denote levels at which the work-in-progress (WIP) is considered unacceptable. There are numerous ways that these limits can be determined, but they are typically derived from statistical analysis of process data, for example ± 2 standard deviations around the sample mean. Another set of parameter limits is the set of engineering limits that are defined by the design engineers. These engineering limits are often used to define the specification limits. It should be obvious that control limits place tighter constraints on the process than specification limits.

As indicated by the manufacturing process description in the previous section, product testing plays a significant role in IC manufacturing, providing the data that informs key decisions in process control and analysis. The acquisition of wafer data comes at some expense. Some analysts estimate that the cost of testing comprises greater than 50% of the product price (Dislis, et al., 1993). Others claim that test dominates the cost of IC manufacturing (Kumar and Erjavic, 1993). This is due, in part, to the fact that dice are created en masse (i.e., as wafers), but tested individually. Others note that wafer test is the bottleneck on production and that reducing testing can result in increased production starts and greater throughput (Longtin, et al., 1996).

There are several aspects to the expense of IC testing. First, IC testers are expensive; many cost several million dollars each. Second, testing is expensive, requiring personnel and production time. Third, testing can cause damage to devices, both physical (e.g., probes) and electrical (e.g., faulty test signals). Since testing is both expensive and potentially damaging, it is important to optimize the number and the types of tests performed. This process is assisted by intelligent analysis of test results so that maximum benefit is derived from this resource. The key to interpreting and exploiting test results is accurate models of process performance. It is important to be able to predict correct performance and to detect deviations. Accurate models allow engineers and managers to plan, monitor, and control manufacturing operations.

2.4 Models of Wafer Test Data

Mathematical models of process performance play several important roles in IC manufacturing and considerable research has been devoted to developing accurate models.

There are three major applications of mathematical models of wafer test results:

1. Yield prediction and cost estimation. Yield models can be used to predict the manufacturing costs for other products or products under consideration.

(Cunningham, J. 1990). Accurate estimates of yields and costs allow planners to make decisions about expanding manufacturing operations or initiating new projects. The costs of changing or creating a wafer fabrication line make reliable forecasts necessary.

2. Yield analysis and feedback to fabrication. Test results can be compared to predictions from models and discrepancies can focus attention on processing problems. The models developed for yield analysis often include some representation of spatial information, since such information is useful for determining root causes of process problems. This class of models includes techniques from spatial statistics, (e.g., cluster analysis, (Cunningham, S., 1995a; Hansen, et al., 1997)) and artificial intelligence (AI) (e.g., artificial neural nets (Zhang and Milor, 1993; Koppenhoefer, et al., 1997; Zinke, et al., 1997), fuzzy classifiers (Luria, 1993), and the k-nearest-neighbor algorithm (Tobin, 1998)). The spatial statistics models are employed primarily to decide if test results indicate significant spatial clustering. The AI models are typically classifiers, and their purpose is to determine whether the test data exhibits patterns associated with known problems. These models are used as part of a post-test analysis of process data.

3. Process control and product disposition. Actual yields can be compared to predictions from yield models and the results can guide decisions about processing of the current parts under test. The models employed in this application are usually simple lot-level and wafer-level summary statistics. They typically provide estimates of the expected number of good or defective parts. Until recently, models developed for this application ignored spatial information. They essentially assumed that wafer defects were uniformly distributed across the wafer surface. Some newer models include spatial analysis, but they typically employ this information only to compute summary statistics, i.e., the spatial information is used to make more accurate yield estimates. A more thorough review of process control models is presented below.

The key distinction between yield analysis and process control is that yield analysis is focused on understanding the process in order to effect changes to future production, whereas process control is concerned with making decisions about the current work in progress (WIP). This corresponds to the difference in focus between statistical quality control (SQC) and statistical process control (SPC) (Zorich, 1991). The yield analysis models are typically employed off-line after testing. The control models are on-line and operate in real time. In practice these two tasks are closely related and the distinction between them is often blurred. Accordingly, "SPC" and "SQC" are usually treated as synonyms. A sample of the models commonly applied to process control is reviewed next.

SPC systems employ a variety of relatively simple statistical models in the construction of process control limits. The most common is standard deviations around the mean of process parameters, including various yield measures. Assuming that the process is stationary, then successive sample means will be normally distributed. Thus control limits are established by collecting a sample of test data and computing confidence intervals of $\pm 2\sigma$ or $\pm 3\sigma$ around the sample mean. Process data is then collected and compared to these limits in real time. Often the process data statistics are charted and monitored for both trends and exceptions. Other common forms of statistical models employed in SPC systems include the range, the standard deviation, the variances, and various counts, e.g., the number and rate of defective parts.

Slightly more complex models are employed in acceptance sampling. In acceptance sampling, yield models are used to derive screening policies for determining whether products pass to the next processing step (Albin and Friedman, 1989; Longtin, Wein, and Welsh, 1996, Tomlinson, 1997). The standard application employs a Poisson distribution with mean set to np, where n is the number of parts and p is the fraction of parts that are defective (Albin and Friedman, 1989). This approximates a binomial distribution with parameters n and p. A fixed number of parts is examined (the sample size) from a finite population. If the number of defective parts exceeds a critical level determined by the Poisson distribution then the population of parts is rejected. This approach is commonly applied to screening wafer lots. In this case a few wafers are examined to determine the disposition for the entire wafer lot. If the examined wafers contain too many bad dice, then either the entire lot is rejected or more testing is performed. Otherwise it is passed on to the next processing step (Albin and Friedman, 1989). Acceptance sampling is motivated by the desire to minimize testing and is based on the realization that no money can be made by testing bad parts.

A number of researchers have noted the limitations of the Poisson distribution in semiconductor yield modeling (Albin and Friedman, 1989; Cunninham, J. 1989, Cunningham, S. 1995a, Lontin, et al., 1996). First, as chip area increases, the Poisson model becomes less accurate than other models, e.g., the negative binomial. Also, it assumes a uniform distribution of failures and therefore doesn't capture the spatial distribution of wafer defects. Thus, the Poisson model is only useful in estimating overall wafer yields, not in predicting where those yields will occur on the wafers.

Another criticism of the Poisson distribution is that it doesn't provide any support for more intelligent test policies. It can be employed to determine how many wafers per lot to test, and how many dice per wafer to test, but it can not answer the question of which dice should be tested. Lately some researchers have begun to consider wafer-testing policies that attempt to exploit the spatial distribution of wafer test results. Such policies require a more explicit treatment of spatial dependencies than that provided by the Poisson distribution. Lontin (Longtin, et al., 1996) considered eight different screening strategies that attempt to exploit yield nonuniformities. Each strategy provided a policy for deciding which dice to test on a wafer. Their strategies included exhaustive (test every chip), radial (concentrate on the center of the wafer), checkerboard (every other chip), and a variety of other partial test policies. Each policy was designed to exploit a particular type of spatial distribution of wafer defects. They propose two models of yield, the Markov random field and the Bayesian gamma-gamma and note that these models avoid the simplifying independence assumptions of the Poisson, Bernoulli, and binomial random variables (Lontin, et al., 1996; Ou and Wein, 1996). A final interesting aspect of the work performed by Longtin is their attention to the relationship between testing, product starts, throughput, and profits. They argue that wafer test is often a process bottleneck, and in such operations a decrease in testing allows an increase in wafer starts and therefore an improvement in overall throughput and profits (Longtin, et al., 1996).

2.5 Die-Level Functional Test

DLFT is an important and complex process and provides the application focus of the research described in this thesis. This section describes the conventional approach to DLFT in more detail.

Die-level functional tests (DLFT) measure the operational integrity of the individual die. While they are still in wafer form, signals are fed to the individual dice, and the outputs are checked to verify that the dice are operating correctly. Tests are performed across a range of operating conditions for the device under test (DUT). For example, dice are tested under conditions of low voltage, nominal voltage, and high voltage to ensure robust performance across the spectrum of possible voltage levels. In functional testing, each die is subjected to a sequence of tests. This testing sequence is often terminated at the conclusion of the first failed test. A die that passes the entire sequence of tests is deemed good.

The resulting distribution of functional die test results provides a basis for making decisions regarding the following:

- wafer disposition (ship, scrap, retest, hold for yield analysis)
- tester diagnosis and maintenance (preventive maintenance., alignment problems, failures)
- manufacturing process corrections/improvements

Thus functional testing performs several roles. First, as the basis for the disposition decision for the device tested, e.g., ship or scrap. Second, for feedback about the processes that created the device. Third, as the basis for other models, e.g., yield models.

Testing is expensive and testing can cause defects, either physical (through probing) or electrical (through faulty test signals). Thus process engineers strive for "just enough test." Functional test determines which dice to package, and, since packaging is expensive, it is important to avoid poor package decisions. Functional tests are also expensive, so optimizing the test and package decisions is a non-trivial problem. A closer examination of functional testing is presented next.

There are three key decisions involved in die-level functional test:

1. Which dice to test?
2. Which wafers to hold, which to send to package?
3. On the wafers sent to package, which dice to package?

The conventional approach to these decisions can be described as follows:

1. Exhaustive test, i.e., test all dice on each wafer.
2. Hold wafers that exceed SPC limits; send others to package.
3. On wafers sent to package, package all dice that passed die-level functional test.

This approach to functional testing is depicted in figure 2.

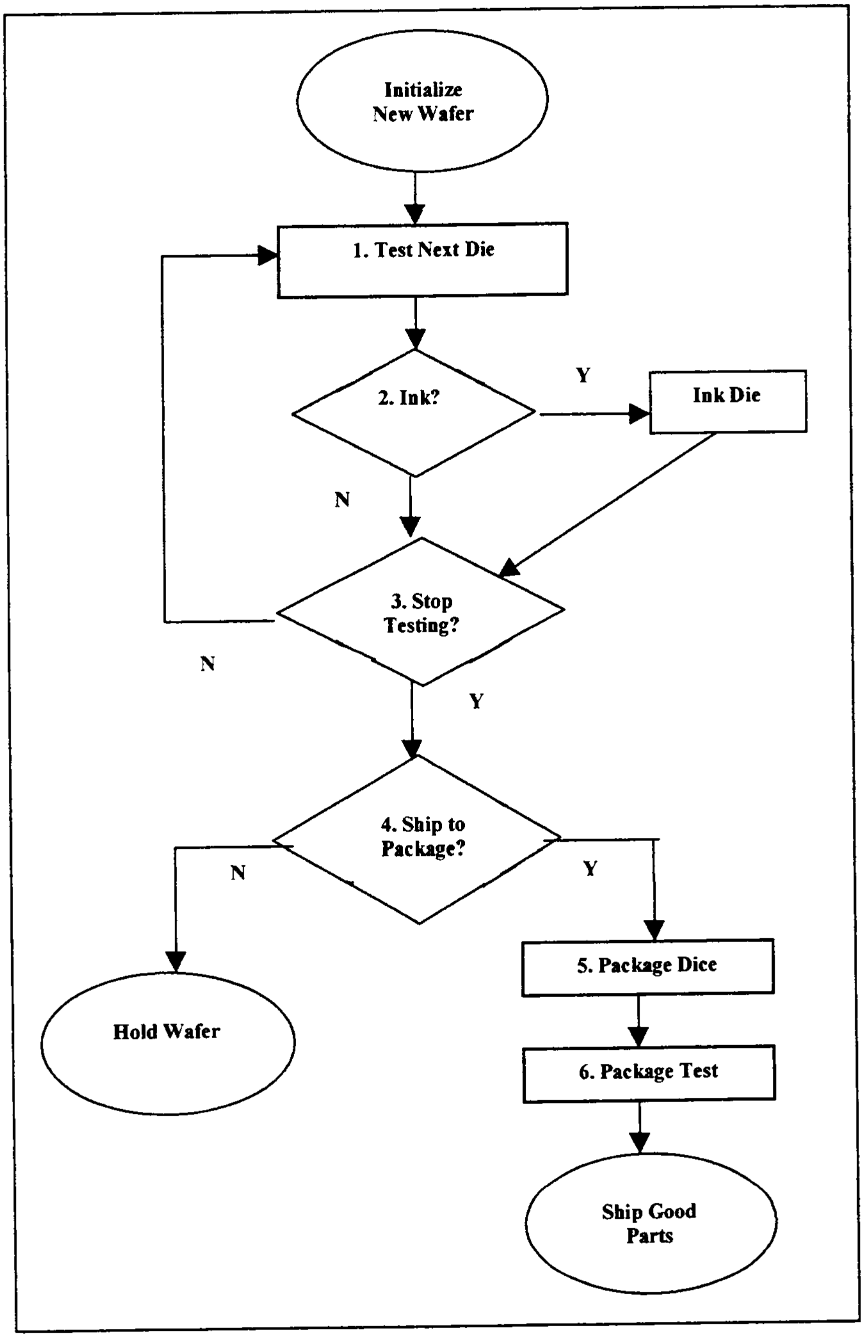


Figure 2: Exhaustive Test Flowchart

Brief descriptions of the key elements of figure 2 are presented next:

1. Test Next Die: dice are tested sequentially in an exhaustive serpentine (sweep) pattern.
2. Ink Die: a die that fails functional test is marked with an ink dot ("inked").
3. Stop Testing: testing continues until the wafer is finished. There are no intra-wafer interrupts except for emergencies, for example symptoms of tester problems.
4. Ship to Package: basic SPC (statistical process control) rules determine wafer disposition.
5. Package: all unmarked (uninked) dice on accepted wafers are packaged.
6. Package Test: all packaged dice are tested and sorted prior to shipment.

Functional test results can be summarized as wafer maps. This makes it easier to perform visual inspections of test results to detect spatial patterns of failures. Figure 3 presents a sample of four wafer maps. Red (dark) squares indicate failed dice, green (light) squares indicate good dice.

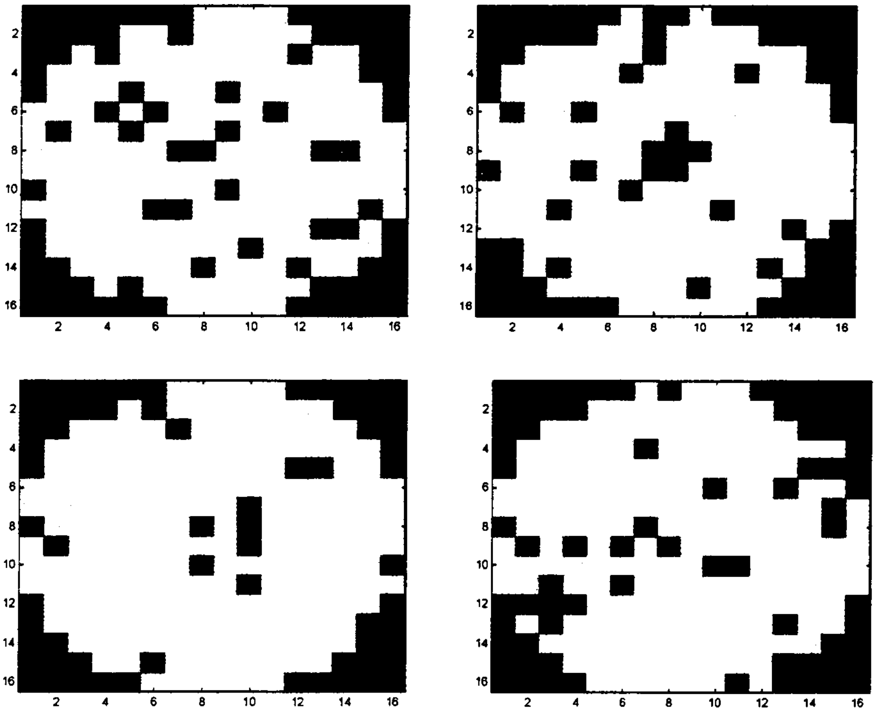


Figure 3: Sample Wafer Maps

2.6 Wafer Test Data Set

For this study a data set containing the functional test results from 50 lots was provided by Hewlett-Packard Company. The data originated from a stable run of a mature product. Each wafer had a 6" diameter and contained 209 dice. Each lot contained 24 wafers for a total of 1200 wafers. As part of the analysis, summary statistics were computed, including the mean and standard deviation for the number of good dice for each lot. These statistics are presented in figure 4 and figure 5, respectively, and indicate that these test results were from a relatively stable manufacturing line.

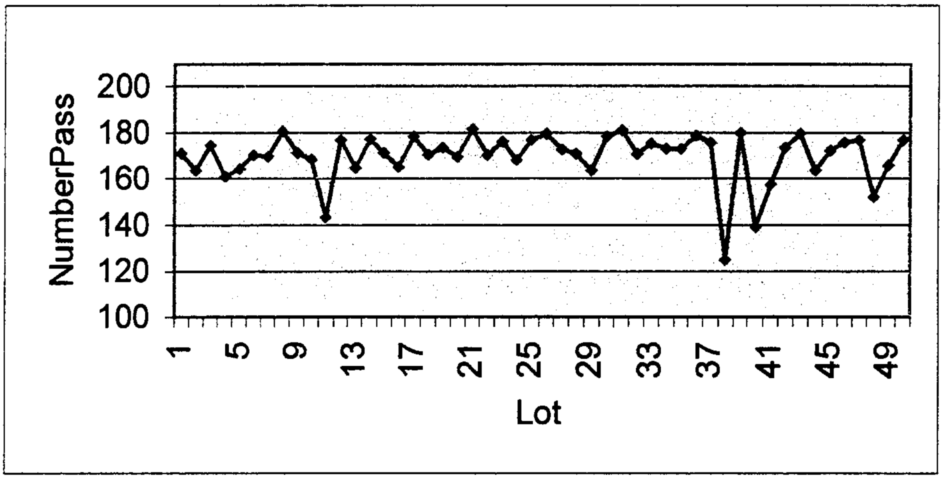


Figure 4: Wafer Data Average Yield

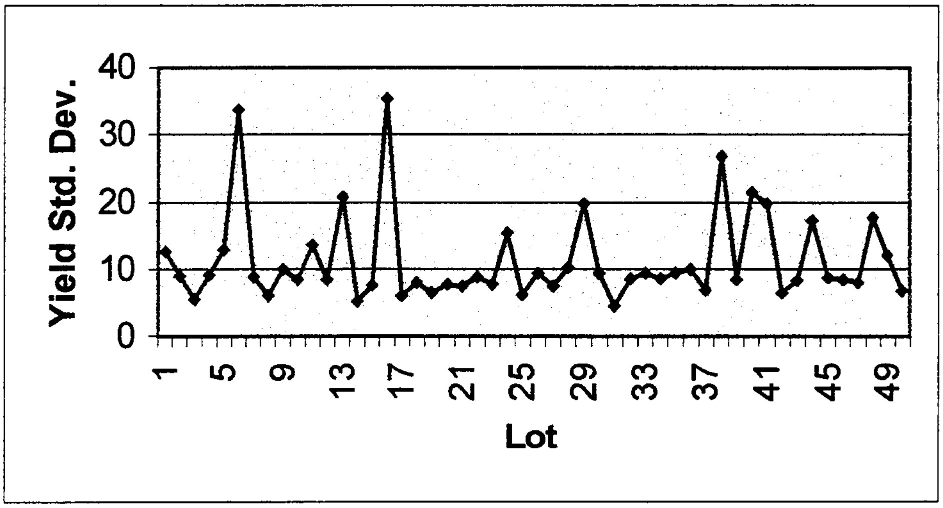


Figure 5: Wafer Data Average Standard Deviations

In addition to the summary yield statistics presented above, another interesting characteristic of the test results is the degree of spatial clustering of defects. Hansen, Friedman, and Nair (1997) developed their T-statistic to measure this aspect of wafer test data. Their test statistic is a sum of two weighted 'join-count' statistics; one counts the number of defective neighbors of defective dice and the other counts the number of nondefective neighbors of nondefective dice. For a given yield level, the test statistic will be large if either the defective or the nondefective dice tend to be spatially clustered. They prove that this statistic is approximately normally distributed with the corresponding moments. Figure 6 depicts a 90% confidence interval for various yield levels with the computed T-statistics for the 1200 wafers.

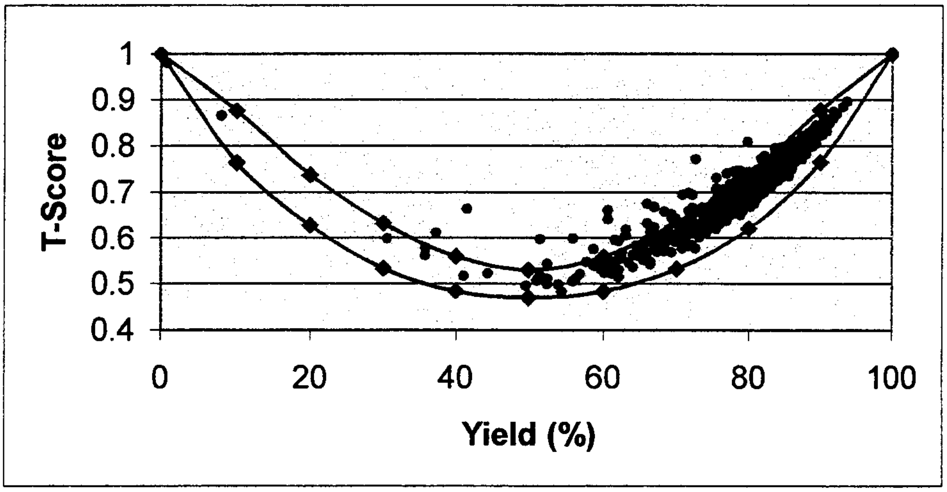


Figure 6: Wafer Data T-Statistics

There are 43 points outside the confidence limits, or approximately 3.6% exceptions. Thus, spatial clustering of the defective dice is not a significant issue with this data set. This means that there were few clusters of failed dice, however the T-statistic does not detect other types of spatial regularities, for example, dice around the edge of the wafers often fail at a higher rate. Thus although this data set does not contain significant spatial clustering as measured by the T-statistic, it may contain other types of defect patterns.

2.7 Testing Policy Alternatives

There are three general types of policies for determining which dice undergo die-level functional test: exhaustive test, no test, and selective test. The advantages and disadvantages of these types of policies are discussed in this section.

The conventional approach to die-level functional testing is exhaustive test, in which every die on every wafer is tested. There are two significant advantages to the exhaustive test policy. First, the test results provide information about the quality of the manufacturing process. Second, the exhaustive test approach produces accurate package and disposition decisions. Since the functional test results are known for each die, bad dice can be identified and rejected prior to the package process. Also, wafers that contain a significant number of bad dice can be identified and held for yield analysis. Thus, the exhaustive test policy provides timely process feedback and supports accurate disposition and packaging decisions. There are two significant disadvantages to the exhaustive test policy. First, die-level functional tests are expensive due to the costs of the testing machines, the associated personnel, and the production time. Second, die-level functional tests can damage the dice, either through the physical act of probing to establish electrical contact, or through faulty test signals.

A second type of testing policy is one in which no die-level functional testing is performed, i.e., a test-none policy. Under a test-none policy all wafers are shipped to package and all dice are packaged. The advantage of a test-none policy is that no resources are expended on die-level functional test. Thus, the resources normally associated with testers, personnel, and production time are saved. When the yield is high and the manufacturing process is stable then the test-none approach can produce profits that exceed the profits of the exhaustive test approach. However, since yield is never perfect, the test-none policy results in some bad dice being packaged. Final package test prevents any bad die from being shipped to a customer, so product quality is assured. However, if the cost of packaging is high, then a relatively few bad dice will result in lost profits. Thus, there are two disadvantages to the test-none policy. First, the test-none policy results in some bad dice being packaged. Second, and perhaps more significant, is that the test-none policy provides no feedback on the quality of the manufacturing process. Under a test-none policy, process problems are only detected at final package test. Given a global value delivery system, for example, one in which wafer fabrication occurs in Corvallis, OR and packaging occurs in Singapore, this delay in feedback would be risky. Without die-level functional test, an out-of-control process would continue to generate bad wafers for an intolerably long time.

A third type of testing policy is a selective-test policy, in which only some of the dice are tested during die-level functional test. A selective-test policy attempts to achieve the benefits of the exhaustive-test and test-none policies, while avoiding their disadvantages. The idea is to test only to the degree required to make accurate package and disposition decisions, and to detect process problems. The advantages of a selective-test policy are that it conserves testing resources, and provides adequate process feedback. There are two possible disadvantages. First, since the selective-test policy makes package decisions without complete functional test results, there is some possibility of incorrect package decisions. Second, a selective-test policy is more complicated than either the exhaustive-test or test-none policies. This dissertation explores the development and evaluation of a selective-test policy for die-level functional test.

2.8 Summary

Semiconductor manufacturing is a complex operation in which IC testing plays a significant role. One type of IC test is die-level functional test (DLFT), which measures the operational characteristics of individual dice. The conventional approach to DLFT involves testing every die on every wafer. These tests serve two purposes. First, they determine which dice to package. Second, they provide process monitoring and feedback information. However, functional testing is expensive (testers, personnel, production time), can induce problems, and may be unnecessary, since all packaged dice undergo package testing. In cases where the product is mature and the process is stable, simply packaging all dice and eliminating final wafer testing would lead to increased profits. The problem with this approach is that there is no guarantee that a stable process will remain stable and, without DLFT, changes in yield are only discovered at package test. The selective test approach provides a middle path between exhaustive test and no test. The idea is to test only a sample of dice from each wafer and then use these results to make package decisions for all dice on the wafer. Thus testing is minimized, yet process problems and bad wafers are caught prior to wafer assembly. The selective test approach is the subject of chapter 3.

3. DECISION-THEORETIC WAFER TEST

An alternative to the exhaustive test policy for wafer test is a selective test policy. Under a selective test policy, wafers are tested only to the degree needed to make good package decisions. The assumption is that die defects exhibit a spatial distribution that can be predicted by testing only a sample of the dice on any wafer. This assumption is exploited by the testing policy in order to make package decisions for all dice on a wafer while only testing a sample of the dice on that wafer. Thus, the decision to package a die is based on the *expected* utility of the package test result for that die, rather than on the observed functional test result for that die. In other words, a die is packaged if it is expected to pass package test and produce a profit regardless of whether it was functionally tested while in wafer form. Since all packaged dice are functionally tested after packaging, there is no chance that a bad die will be shipped to a customer. Thus the selective test approach attempts to eliminate unnecessary functional tests.

The selective test approach requires a mapping from the functional test results of some dice to the package test results of other dice on the same wafer. Since that relationship is non-deterministic, this mapping will be accomplished with a stochastic model. The selective test approach also requires methods for evaluating testing alternatives with respect to their effects on expected profits. Since this is a problem of decision making under uncertainty, techniques from statistical decision theory will be employed. Thus uncertainty will be represented by probability theory, and decisions will be based on maximizing expected utility. These assumptions lead to the formulation of the wafer test problem as a type of partially observable Markov decision model (POMDP). The details of that formulation are the focus of this chapter.

In general terms, a wafer test controller based on the selective test approach can be described as follows. The system begins with a prior probability model of wafer test results and a utility model over processing costs. The system chooses the best die to test according to a value-of-information metric. This die is tested, and the results are incorporated into the system's model. The system then repeats this test-and-update cycle until the expected value of further testing is non-positive. At that stage, the system makes an individual package decision for each of the dice on the wafer, marking those that are rejected. Once this is complete, the system determines wafer disposition, either ship to package or hold for analysis. For each of the wafers that make it to package, the unmarked dice are packaged and sent to package test. Those that survive package test are shipped to customers.

The remainder of this chapter the organized into the following sections:

3.1 Structuring the Problem: Wafer Test Control as Staged Decisions. The selective test approach is decomposed into a series of decisions corresponding to the major control points in the process. This structure is analogous to the task structure of the exhaustive test approach presented in chapter 2.

3.2 Decision Theory for the Wafer Test Problem. This section reviews the relevant concepts of statistical decision theory and operations research.

3.3 Belief Networks, Inference Algorithms and Influence Diagrams. This section describes the application of probabilistic inference techniques to the formulation and solution of the wafer test control decisions.

3.4 Summary. The development of decision models for the wafer test problem is summarized.

3.1 Structuring the Problem: Wafer Test Control as Staged Decisions

The selective test policy can be decomposed into a series of decisions and structured into a real-time control procedure. The main body of this procedure is a loop over die test selection and test execution steps. Upon termination, the system makes packaging and wafer disposition decisions. This control structure is depicted in flowchart format in figure 7.

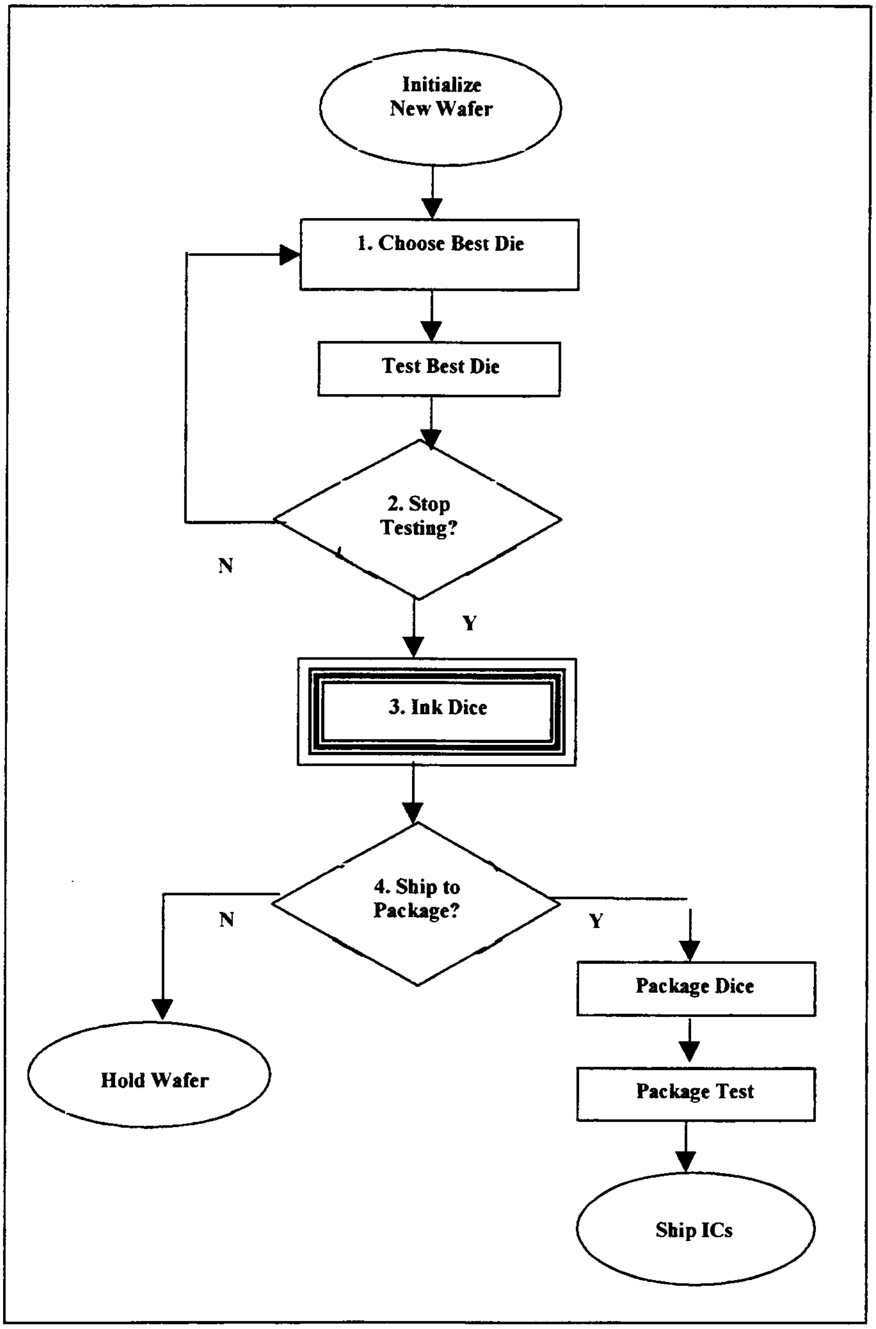


Figure 7: Selective Test Flowchart

In the flowchart presented in figure 7, numbered boxes indicate key decisions. Note that the multiple borders around the Ink Dice decision denote that the inking decision is actually *n* separate decisions, one for each die on the wafer. Since the decision to ink any single die is independent of the decision to ink any other die, these decisions are represented as distinct decisions (rather than a single decision to ink the wafer). This independence of inking decisions will play a critical role in managing the complexity of control decisions, and therefore it is introduced at this point and maintained throughout the development.

The key decisions from the selective test flowchart are described in general terms below. The details of how these decisions are actually made are deferred until section 3.3.

1. Best Test: Choose the best die to test.

   Among the untested dice on the wafer, select the die that is predicted to provide the most useful information for making package decisions for all dice on the wafer. Since wafers exhibit variability, it is reasonable to expect that the optimal sequence of dice tested will differ across wafers. On some wafers, certain dice will provide the most useful information; on other wafers other dice will provide more useful information. Thus an optimal policy should incorporate the flexibility of selecting any die to test from among the remaining untested dice at any point during testing.

2. Test Termination: Continue testing of the current wafer or stop testing and proceed to the package decisions.

   For each wafer, selectively test dice on that wafer until there is no expected profit in testing any of the untested dice. At that point, stop testing, make package decisions, and initiate testing of the next wafer. Given the variability among wafers, it is reasonable to expect that the optimal number of tests will vary across wafers. Therefore an optimal policy should incorporate the flexibility of performing varying degrees of testing.

3. Inking: For each die on the wafer, decide if it should be packaged or scrapped.

   Since it is possible for any subset of dice to be rejected, the optimal policy should incorporate the flexibility of inking any subset of dice. Individual inking decisions are independent of one another (the decision to ink any single die has no effect on the inking decisions of the other dice). Therefore for each wafer there are *n* individual inking decisions to determine which dice to package and which to reject.

4. Wafer Disposition: Decide if the wafer should be sent to package or held for analysis.

   There is a shipping and handling cost incurred for each wafer sent from the fabrication plant to wafer assembly to be packaged. Only wafers whose expected profits exceed these costs should be packaged. Therefore an optimal test policy should provide estimates of net profit value and be able to weigh these against the shipping and handling costs. Wafers that are not sent to package are assumed to be defective and are held for yield analysis or recycled.

The next section presents a formal statement of the wafer test decisions in terms of expected utility.

3.2 Decision Theory for the Wafer Test Problem

Statistical decision theory is a formal model of rational decision making under uncertainty and provides a mathematics of uncertainty, a logic of prediction and decision making, and real-valued utility measures for expressing preferences and ranking alternatives (Savage, 1954; Raiffa, 1968; Berger, 1985, North, 1968). Statistical decision theory is founded on the notion of lotteries and preferences among lotteries, but for applications, the key concept is expected utility, which is defined as the product of utility and probability. In an uncertain world, expected utility provides a principled basis for rational behavior.

Statistical decision theory provides techniques for representing decision problems and performing expected utility calculations. It assumes that a decision problem can be described in terms of a set of states, a set of actions, a transition function that maps states and actions to other states, and a utility function over states. From this representation, the expected utility of actions can be computed. According to the principle of rationality, a rational agent will act in such a way as to maximize its expected utility. Thus a rational agent operating in an uncertain world will choose actions consistent with those specified by statistical decision theory. A rational decision will produce optimum expected utility; therefore an optimal decision policy is a decision policy that specifies a rational course of action. Statistical decision theory is concerned with the development of optimal decision policies.

It is often noted that decision theory is a normative model of decision making and a distinction is drawn between the formal rationality of decision theory and the normal behavior of human decision-makers (Simon, 1983). It is well established that human decision-making exhibits systematic deviations from normative behavior (so-called "cognitive illusions" to borrow Pearl's phrase; Pearl, 1988, Tversky and Kahneman, 1981; Einhorn and Hogarth, 1981; Shepard, 1964). This effect is quite apparent in probability assessment tasks, where alternative framings of a single problem elicit contradictory responses. The cognitive limitations of human reasoning and the imperfections of short-term memory in particular, are the usual explanations for the irrationality of human decision-makers. Decision theory is offered as a model of ideal decision-making, a standard to be emulated, and a tool to compensate for the frailties of human cognition. In these roles, decision theory has found broad acceptance and wide application. However, with the advent of computerized decision-making and the development within the Artificial Intelligence community of the notion of the autonomous agent, decision theory has found new applications and new practitioners. In addition to assisting human decision-makers, decision theory provides a foundation for constructing automated decision-making systems (Holtzman, 1988; Horvitz, Breese, and Henrion, 1991; Pearl, 1988). The application of decision theory to the wafer test problem follows this line of development.

Statistical decision theory distinguishes two types of actions: those that affect the state of the world and those that provide information about the state of the world. These latter actions correspond to information gathering acts, for example, observations, tests, and experiments. The value of an information-gathering act is measured as the difference between (1) the expected utility that results from performing the information-gathering act and then following an optimal decision policy, and (2) the expected utility of following the optimal decision policy alone. When value of information (VOI) is measured as immediate gain, i.e., only a single information act is considered, then it is referred to as myopic VOI. When it is assumed that the result of the information-gathering act will be known with certainty, then VOI is sometimes referred to as the value of perfect information (VPI). For some problems all aspects of the problem states are observable, in other problems there exist aspects that are unobservable.

A sequential decision problem is a decision problem whose solution requires a sequence of decisions to reach an overall goal. A stochastic sequential decision problem with observable states is called a Markov decision problem (MDP) (Russell and Norvig, 1995). Exact solutions for MDPs are provided by the methods of value iteration and policy iteration, both of which were developed by Howard (1960) and are related to dynamic programming and Bellman's principle of optimality (Dean and Wellman, 1991). This principle states that "An optimal policy has the property that, whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision" (Bellman, 1957; White, 1978). Thus the formula for computing the optimal policy can be expressed as a set of recurrence relations in which the maximum expected value of a state is determined by a linear combination of the current state value and the maximum expected value achievable in the remaining states.

When a stochastic sequential decision problem contains unobservable states, it is called a partially observable Markov Decision problem (POMDP) (Russell and Norvig, 1995). Solutions to POMDPs are particularly complex due to the combination of uncertainty and unobservability. The standard approach to solving a POMDP is to first create an equivalent MDP and then apply value iteration or policy iteration (Smallwood and Sondik, 1973; Monahan, 1982). The translation from POMDP to MDP involves encoding the state uncertainty as a model variable with continuous parameters. To avoid a potentially infinite state space, the range of these parameters is discretized. Unfortunately the computational complexity of exact solutions remains prohibitive except for small problems (Littman, et al., 1995). Several approximation techniques have been proposed, however complexity issues prohibit even these methods from scaling to large problems (Cassandra, Kaelbling, and Littman, 1994; Parr and Russell, 1995). In practical applications, the solutions to POMDPs generally rely on simplifying assumptions about the complexity of model interactions and the length of the decision sequence. In the wafer test problem, several assumptions are employed to keep the complexity manageable. These assumptions and the problem formulation are described in following sections.

3.3 Belief Networks, Inference Algorithms and Influence Diagrams

Belief networks provide a succinct representation for probabilistic models and inference algorithms provide computational answers to questions of interest (Pearl, 1988; Charniak, 1991; Oliver and Smith, 1990). The key idea is to take advantage of independence relations within the probabilistic model in order to factor the model into more primitive elements. Then probability theory is employed to compute other values of interest.

A belief network has a graphical representation in terms of a directed acyclic graph (DAG). In this representation, nodes correspond to variables and arcs between nodes correspond to conditioning relations. At the mathematical level, a belief network is a collection of marginal and conditional prior probability distributions. A marginal distribution is defined for each source node (unconditioned variable) and a conditional distribution is defined for each of the other nodes. These distributions can be either discrete or continuous, and a single model can contain a mixture of both. From a computational perspective, probabilistic inference is the calculation of prior or posterior probability distributions over a set of random variables. A fully specified belief network implicitly defines a coherent (complete and correct) probability model. From such a network, any prior or posterior marginal, conditional, or joint probability distribution can be computed by means of probabilistic inference algorithms.

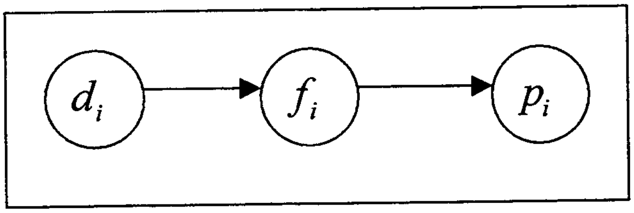


Figure 8: Single Die Belief Net

Figure 8 depicts a simple belief network for modeling the test results for a single die. It consists of three random variables: d_i, f_i, and p_i. These variables are described below.

d_i represents to the true state of die *i*. There are two possible die states: good and bad. $P(d_i)$ denotes the corresponding marginal probability distribution over die states.

f_i represents to the functional test result for die *i*. There are two possible test results: pass and fail. $P(f_i|d_i)$ denotes the corresponding conditional distibution over the functional test results given the true die state.

p_i represents the package test result for die i. There are two possible test results: pass and fail. $P(p_i|f_i)$ denotes the corresponding conditional distribution over the package test results given the functional test result.

There are several general approaches to developing probabilistic inference algorithms for computing probability values from belief networks. For small models and fixed probability queries, often the simplest and most efficient approach is the direct application of probability theory, in particular Bayes' Theorem and the product rule. This is the approach taken in this study. More general solutions are provided by graph-theoretic and algebraic approaches. Examples of the graph-theoretic approach include belief propagation (Pearl, 1988), clustering (Lauritzen and Spiegelhalter, 1988) and cutset conditioning (Horvitz, Suermondt, and Cooper, G. 1989). The principal algebraic approach is symbolic probabilistic inference (SPI) (Shachter, D'Ambrosio, and DelFavero, 1990 ; Li, and D'Ambrosio, 1992). There are also numerous approximation algorithms, including simulation approaches (Shwe and Cooper, 1990; Henrion, 1988) and search-based methods (D'Ambrosio, 1992; Henrion, 1991). At the logical level, the probability calculations performed by the probabilistic inference algorithms correspond to various inference tasks, such as predictive (causal), diagnostic (abductive), and intercausal reasoning. They also provide a basis for statistical decision making (Shachter and Peot, 1992; Cooper, 1988).

For the die test belief network depicted in figure 8, the package test probabilities can be computed by the following formula: $P(p_i) = \sum_{d_i, f_i} P(d_i)P(f_i|d_i)P(p_i|f_i)$.

The influence diagram formalism is a graphical knowledge representation for decision problems and is closely related to the belief network representation (Howard and Matheson, 1989, Oliver and Smith, 1990; Horvitz, Breese, and Henrion, 1991, Shachter, R., 1990). Influence diagrams extend the belief network representation by adding decision and value nodes, and additional interpretations for the arcs associated with these nodes. The decision nodes represent decisions, and the value nodes represent the utility function. In the graphical representation, boxes represent decisions, and diamonds represent value functions. Arcs into decision nodes imply prior knowledge. Arcs into value nodes indicate preference dependencies. Solving an influence diagram means identifying values of the decision variables that maximize the expected value of the value nodes.

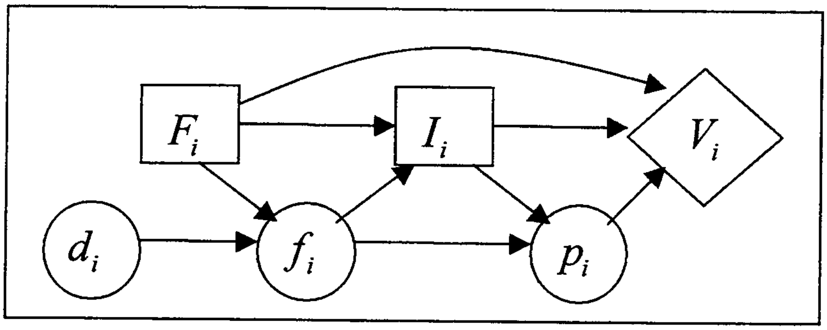


Figure 9: Single Die Influence Diagram

Figure 9 presents a graphical representation of the testing and packaging decisions for a single die. This influence diagram extends the belief network of figure 8 by adding decision nodes and a value node. These new model elements are described in more detail below.

F_i is a decision variable and corresponds to the decision to functionally test die *i*. There are two possible decision alternatives to the functional test decision: test and not-test.

I_i is a decision variable and corresponds to the decision to ink die *i*. There are two possible decision alternatives to the ink decision: ink and not-ink. Recall that inking a die means that the die will be scrapped. On wafers that are sent to the package process, all dice that are not inked are packaged and then packaged tested.

V_i represents the value function for the decision model. It incorporates the associated costs of functional testing and inking (i.e., packaging and package testing), and the die value (i.e., the single die selling price). Thus there are four parameters to the value function:

$c_f \equiv$ the cost of a single functional test

$c_k \equiv$ the cost of packaging a single die

$c_p \equiv$ the cost of package testing a single die

$v \equiv$ the value of a single packaged die

These value function parameters are provided by the manufacturer. Note that a die that is functionally tested incurs the functional test cost. A die that is not inked is packaged and thus incurs the packaging and package testing costs. A die that is inked is neither packaged nor packaged tested, and therefore has no package test result. A die that passes package test can be sold and thus accrues the single package value.

Given these parameters, the value function for the single-die influence diagram can be defined as follows:

$$V_i(F_i, I_i, p_i) \equiv \begin{array}{cccc} F_i & I_i & p_i & V_i \\ \end{array} \left[\begin{array}{ccc|c} \text{not-test} & \text{not-ink} & \text{fail} & -c_k - c_p \\ \text{not-test} & \text{not-ink} & \text{pass} & -c_k - c_p + v \\ \text{not-test} & \text{ink} & \text{n/a} & 0 \\ \text{test} & \text{not-ink} & \text{fail} & -c_f - c_k - c_p \\ \text{test} & \text{not-ink} & \text{pass} & -c_f - c_k - c_p + v \\ \text{test} & \text{ink} & \text{n/a} & -c_f \end{array}\right]$$

The following formula can be evaluated to determine the expected utility of functionally testing die *i*:

$$EU(F_i = test) =$$
$$Max_{I_i} \sum_{p_i} \left[\left(\sum_{d_i, f_i} P(d_i) P(f_i | d_i F_i = test) P(p_i | f_i, I_i) \right) V_i(F_i = test, I_i, p_i) \right]$$

Similarly, the expected utility of not functionally testing die *i* can be computed, and the testing alternative that results in the maximum expected utility is deemed best (i.e., the rational choice). Thus, this procedure provides a method for deciding whether a single die should be tested, and it also provides a numeric value of the expected utility of this decision.

Recall that there are two decisions in the single-die influence diagram: the functional test decision and the ink decision. The decisions are ordered so that the ink decision follows the test decision. Thus the result of the functional test decision is known at the time that the ink decision is made, and if the decision was to test, then the functional test result is also known. The following formula can be evaluated to compute the expected utility of inking die *i* given that this die was functionally tested and passed:

$$EU(I_i = ink | F_i = test, f_i = pass) = \sum_{p_i} [P(p_i | f_i = pass, I_i = ink) \; V_i(F_i = test, I_i = ink, p_i)]$$

The expected utility for the other cases can be computed similarly. Thus the influence diagram of figure 9 provides a model for making testing and inking decisions for a single die. This model can be extended to provide a method for making decisions about multiple dice, which is what the selective test approach requires. First, notice that the influence diagram of figure 9 can be simplified by making the following assumption: functional tests provide an accurate measure of true die state. In other words, if a die is good, then it passes functional test; if a die is bad, then it fails functional test. Since functional tests are the sole indicator of die quality, this assumption is reasonable. This means that $P(f_i) = P(f_i | d_i)$, and the influence diagram of figure 9 can be simplified to that depicted in figure 10.

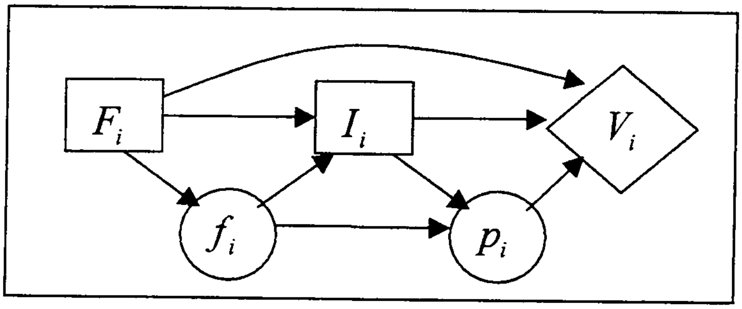


Figure 10: Simplified Single Die Influence Diagram

This results in a simplification of the formula for computing the expected utility of testing a single die:

$$EU(F_i = test) = Max_{I_i} \sum_{p_i} \left[\left(\sum_{f_i} P(f_i | F_i = test) P(p_i | f_i, I_i) \right) V_i(F_i = test, I_i, p_i) \right]$$

The formulas for computing the expected utility of package test decisions are simplified in a similar manner.

The single-die influence diagram of figure 10 provides a basis for developing an influence diagram for the entire wafer. A wafer test influence diagram is first developed under an assumption that die test results are independent. Then this wafer test influence diagram is modified to model the assumption that die test results are conditionally independent.

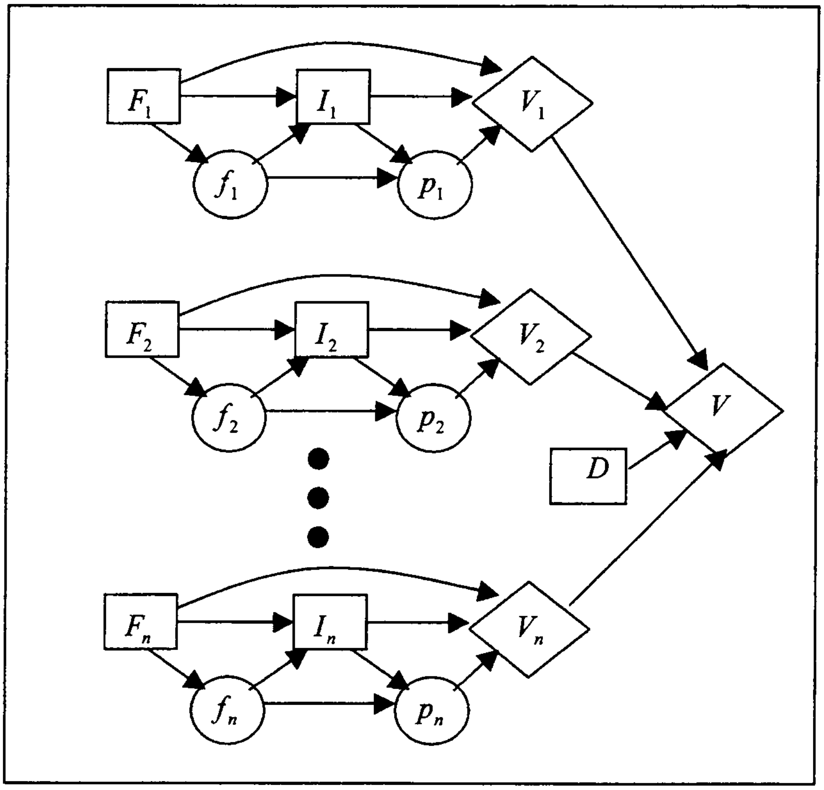


Figure 11: Wafer Test Influence Diagram

The influence diagram in figure 11 represents the test and ink decisions for an entire wafer. For each die on the wafer, there is a test and an ink decision. In addition to the test and ink decisions, there is also a wafer disposition decision, *D*, which represents the decision to either ship the wafer to package or to scrap the wafer. The influence diagram of figure 11 also conatains a value node that incorporates all of the costs and values associated with the wafer. This value function is defined as the sum of the values for the individual die decisions minus the wafer handling cost; i.e., the cost of shipping and cutting (sawing) the wafer. This handling cost depends on whether the wafer was shipped to package or scrapped. If the wafer was shipped, then it incurs a fixed cost; otherwise there is no handling cost. The handling cost is an additional parameter that is provided by the manufacturer.

The influence diagram in figure 11 can be evaluated to provide answers to the wafer test decisions. Recall that the selective test approach is an iterative decision cycle as depicted in the flowchart in figure 7. Thus, rather than deciding a priori which dice to test, the system chooses the single best die to test next. The result from this test is then incorporated into the decision to choose the next best test. This cycle is repeated until it is determined that further testing is unprofitable, and at that time a decision is made to either ship the wafer to package or to scrap the wafer. The selective test approach can be implemented by repeatedly evaluating and updating the influence diagram of figure 11. This process is described in general terms below. In these descriptions, each of the key decisions in the selective test approach is expressed in terms of expected utility. The details are presented in the following sections.

Best test: What is the best die to test next?

Select the test that results in the maximum expected utility that would be achieved if the wafer were to be packaged after that test. Thus the best test is determined by a single step lookahead followed by an evaluation of the package decisions based on the expected test results. For each of the untested dice, the probabilities of the functional test results for that die are computed. Then, based on these functional test probabilities, package decisions are made for all dice on the wafer and the overall expected utility is computed. The die whose corresponding test yields the maximum expected utility is selected as the next best die to test.

Test termination: Stop testing or continue?

The decision to stop testing or to continue with the next test is determined by myopic (one-step) value of information (VOI). Myopic VOI is defined as the difference between the expected utility of stopping and packaging and the expected utility of performing the next best test and then stopping and packaging. The decision to stop testing is made when there is no expected value of testing further, i.e., when the myopic value of information is non-positive.

Inking: Which dice should be packaged?

The decision of which dice to package and which to reject is based on the expected utility of each individual die. An independent inking decision is made for each die on the wafer. These decisions are based on the expected utility of inking the die. For each die, if the expected utility of inking the die is greater than the expected utility of not inking the die, then the die is inked.

Wafer disposition: What happens with a wafer once it is through DLFT?

The decision to ship a wafer to package or to hold the wafer for analysis or recycling is based on the expected utility of shipping vs. the expected utility of holding. If the expected utility of shipping the wafer exceeds the expected utility of holding the wafer then the wafer is shipped.

The next section extends the wafer test influence diagram to model conditional independence of wafer test results and presents the details on the probability and expected utility computations.

3.3.1 Representing Conditional Independence of Die Test Results

For the wafer test decisions, expected utility depends on an estimate of the package test results given a set of functional test results. Thus, the key probability values that are needed are the probabilities of package test results conditioned on a set of functional test results. If it is assumed that the relationship between functional test results and package test results is the same for all dice, i.e., that there is a uniform probability of damaging dice during packaging, then the critical probability values that are needed for the expected utility calculations are the probabilities of unobserved functional test results conditioned on the set of observed functional test results. In other words, the key probability calculation is $P(f_i|f_j = r_j,..,f_m = r_m)$ where f_i is the unobserved test result for die d_i and $\left\{f_j = r_j,..,f_m = r_m\right\}$ is the set of observed test results for dice $\left\{d_j,..,d_m\right\}$. $\left\{f_j = r_j\right\}$ denotes the proposition that the functional test result for die *j* is r_j. In order to create a stochastic model to provide these probability values, some assumptions about the relationship among the dice must be made. The models developed up to this point have all assumed that functional test results are independent. This is a common modeling assumption and greatly simplifies the modeling requirements. However it is widely recognized that functional test results are not independent and that the information ignored by this assumption is valuable in yield predictions (Albin and Friedman, 1989; Cunningham, J., 1990; Longtin, et al., 1996). A more accurate assumption is that the functional test results are correlated, because the physical processes that create defects operate over regions of the wafer or even the entire wafer. For example, a drip or scratch may damage several adjacent die, and a bad processing step may damage all of the dice on a wafer. For this study, we will model these correlations using a finite mixture model. According to this model, there are *K* different "classes" or "families" of wafers. Within each wafer class, each die has its own probability of being defective. When a new wafer is processed, this model assumes that the defects on the wafer were generated according to the probability distribution of one of the *K* classes. The identity of this class is never known (or observed), but there is a probability distribution that specifies the probability that the wafer is generated from each class. This is called a "mixture model", because the probability that a die is defective is a mixture of the probability that it defective according to each of the different wafer class models. This model can be represented as a Naïve Bayes network.

A naïve Bayes' belief network is a relatively simple stochastic model that has proven to be useful in diagnostic systems (Henrion, 1990) and learning and discovery systems (Dietterich, 1997, Cheeseman, Self, Kelly, Taylor, and Stutz, 1988). The naïve

Bayes' belief network also provides a convenient representation for latent-class and finite-mixture models. Figure 12 presents the graphical representation of a naïve Bayes' belief network for computing the wafer test probabilities.

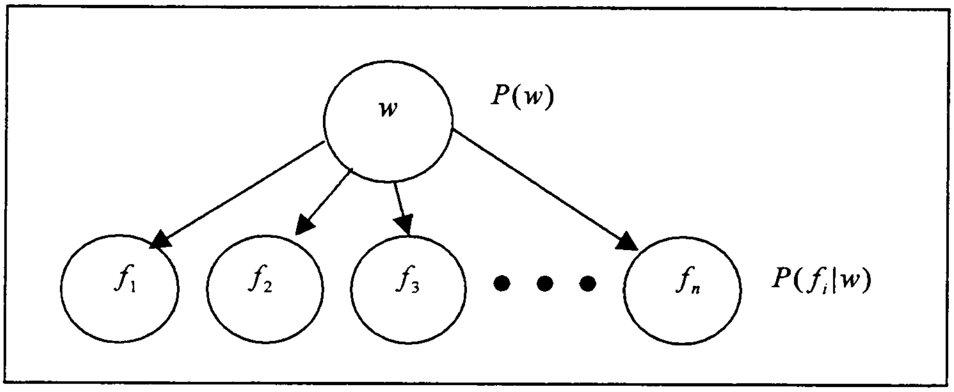


Figure 12: Naïve Bayes' Belief Network for the Wafer Test Probabilities

In this belief net, the root node represents the possible wafer types (or wafer classes), and the leaf nodes correspond to the functional test results for the dice on the wafer. There is a one-to-one correspondence between the leaves of the model and the dice on the wafer. The mathematical definition of this model includes a single marginal distribution over the possible wafer classes, $P(w)$, and a distinct conditional distribution for each of the functional test results, $P(f_i|w), i = 1..n$. This model encodes the assumption that the functional test results are conditionally independent of one another given the wafer class. This model can also represent the assumption that functional test results are independent of one another simply by restricting the number of wafer classes to one.

For this study, the wafer classes are assumed to be abstract and unobservable. This means that the wafers can be partitioned into a number of groups, where members of a group share some stochastic pattern of test results. Each group corresponds to a distinct wafer class. Although the stochastic patterning of the wafers may have some physical basis in the fabrication process steps, this basis is not explicitly identified. Instead wafer classes are given abstract names, and their probability distributions are determined by machine learning techniques that were designed for incomplete-data problems. In other words, the wafer classes are defined by statistical analysis of wafer test data. These techniques are described in chapter 4.

Among the useful probabilities that can be computed from this belief network are the probabilities that are needed for test selection, i.e., the marginal posterior probabilities for functional test results. A direct method for computing the marginal posterior probabilities of die test results can be derived from basic probability theory. For example, the following formula computes the probability of the functional test results for die j given a set of observed tests, $E = \{f_k, .., f_m\}$:

$$P(f_j) = \alpha \sum_{w} \left[P(w) P(f_j | w) \prod_{f_k \in E} P(f_k = r_k | w) \right]$$

Note that although this probability calculation is straightforward, it is not particularly efficient for inference tasks in which the probability values for numerous untested dice are needed, since each calculation recomputes the product of the observed tests. This approach is also inefficient for incremental inference tasks where evidence accumulates over time. In this case the evidence must be saved and considered anew at each probability calculation. As evidence accumulates the complexity of the probability calculation increases. Bayesian belief updating provides an alternative approach for incremental inference tasks (Pearl, 1988; Lauritzen and Spiegelhalter, 1988).

3.3.2 Evidence Processing and Belief Updating

With belief updating, the probability model is updated after each observed test result. This corresponds more closely to the task requirements of the wafer test problem in which observations are monotonic (test results accumulate during the testing of a wafer) and control decisions are made after each test. Under belief updating, the model is explicitly modified to reflect the effects of the new evidence. This addresses both of the shortcomings of the approach mentioned above. First, evidence is incorporated into the model, so it is no longer necessary to save evidence. Second, the complexity of the probability calculations remains constant.

Bayesian belief updating is based on the application of Bayes' rule. The following formula is Bayes' rule for the wafer test model:

$$P(w|f_i = r) = \frac{P(w)P(f_i = r|w)}{P(f_i = r)}$$

This formula describes how to compute the posterior probability of the wafer class given a single functional test result. This provides the basis for a recursive updating formula.

Let *E* represent the test results observed up to this point in testing. Then the impact of the next test result can be computed by the following formula.

$$P(w|E, f_i = r) = \frac{P(w|E)P(f_i = r|w, E)}{P(f_i = r|E)}$$

Since the functional test results are conditionally independent with respect to the wafer class the following identity holds:

$$P(f_i = r|w, E) = P(f_i = r|w)$$

This identity simplifies the updating formula to the following.

$$P(w|E, f_i = r) = \frac{P(w|E)P(f_i = r|w)}{P(f_i = r|E)}$$

Since the denominator only serves to normalize the result, the computation of this term can be performed by normalizing the numerator. This yields an efficient method for incrementally incorporating functional test results.

Thus after each observed test result, the wafer class distribution is updated according to this formula. There is no need to explicitly save evidence, and the complexity of the probability calculations remains constant with respect to the number of dice tested.

Given the belief updating method, incremental evidence and query processing is straightforward. For example, consider the process of incorporating a single functional test result and then computing the functional test probabilities of a single untested die. Let $(f_j = Pass)$ represent the proposition that die j passed functional test. Then update the wafer class distribution with this result, and then compute the functional test probabilities for die k.

Update wafer class distribution: $P(w|f_j = Pass) = \alpha P(w) P(f_j = Pass|W)$

Compute the functional test probabilities for die k:

$$P(f_k) = \sum_w P(w|f_j = Pass) P(f_k|w)$$

3.3.3 Representing Package Test Results

The extension of the belief network from the previous section to incorporate package test results is straightforward. Figure 13 depicts this modified belief network.

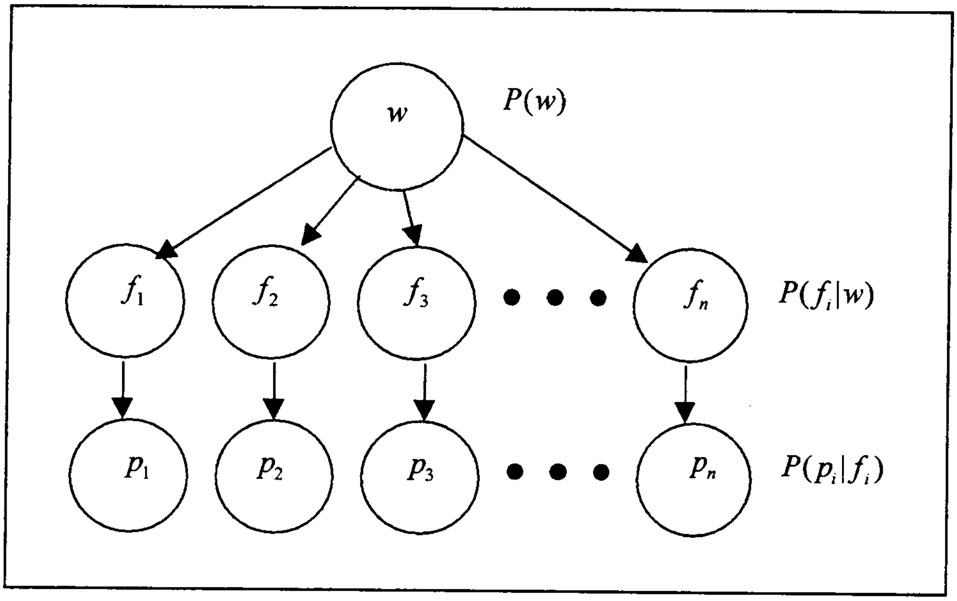


Figure 13: Belief Network with Package Test Results

Since it was assumed that there is a fixed probability of damaging any die during the package process, the addition of the package test nodes requires the specification of only a two additional parameters, $P(p_i = pass|f_i = pass)$ and $P(p_i = pass|f_i = fail)$. The conditional probabilities for cases where $(p_i = fail)$ are simply the complement to the cases where $(p_i = pass)$.

Since typical cost models for IC manufacturing condition IC value on package test results, the belief network depicted in figure 13 provides an appropriate basis for the wafer test decisions.

The belief nets and the inference algorithms presented in this section can be employed to compute the probabilities needed in the expected utility equations. The next section provides the details of how these probabilities are combined with the utility model to solve the wafer test decisions.

3.3.4 Evaluating the Wafer Test Influence Diagram

Figure 14 depicts an influence diagram for the wafer test decisions in which the functional test results are modeled as being conditionally independent. This influence diagram can be evaluated to provide solutions to the wafer test decisions. First, consider the best test decision.

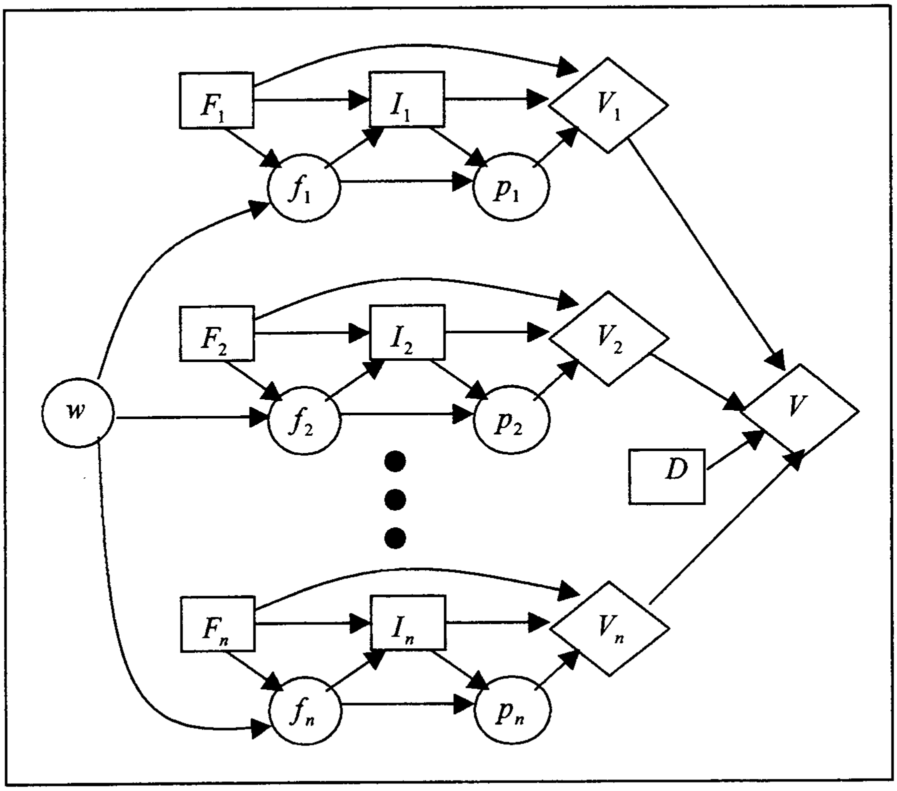


Figure 14: CI Wafer Test Influence Diagram

In this problem, a single functional test is followed by n package decisions (one for each of the dice). The value of a functional test decision is determined by the results of the package tests for all dice, combined with the utility model over test and inking costs and package values. The best functional test is the test that results in the highest expected value over all inking decisions. The evaluation procedure is described next.

At any time during testing, there exists a set of untested dice and a set of dice that have been tested. The evaluation of the best next test model involves considering each of the untested dice and computing the expected utility that would result from testing that die and then making the package decisions for all dice on the wafer. The die for which testing yields the maximum expected utility is the best die to test. For example, the expected utility of testing a single die can be computed as follows.

First, compute the expected utility that would result if this die was tested and it passed. This involves computing the expected utility of inking each of the dice on the wafer, conditioned on the tested die passing functional test. The expected utility of inking a single die depends on the probability of the package test results for that die. The package test probabilities for a single die conditioned on an observed functional test result for another die can be computed as follows.

Let d_j represent the tested die with result $f_j = pass$. Let d_k be the die for which the expected utility of testing is desired. Then the package test probabilities for d_k can be computed as follows:

$$P(p_k|f_j = pass) = \frac{\sum_{w,f_k} P(w)P(f_j = pass|w)P(f_k|w)P(p_k|f_k)}{P(f_j = pass)}$$

Given the package probabilities for d_k, the expected utilities are computed as the product of the package probabilities and the associated package value. This includes the value of the package, the cost of packaging, and the cost of package test.

This procedure can be repeated to compute the expected package utilities for all dice conditioned on the result $f_j = pass$. The sum of these utilities is the expected utility of testing d_j and observing result $f_j = pass$. This value can be denoted by $EU(F_j = test, f_j = pass)$. Similarly the expected utility of testing d_j and observing result $f_j = fail$ can be computed. This yields the value $EU(F_j = test, f_j = fail)$.

Then, given these expected utility values, the overall expected utility of testing d_j is computed as the weighted sum of the probabilities of the test results and the expected utility values minus the cost of testing d_j (i.e., the fixed cost for performing one functional test):

$$EU(F_j = test) = \sum_w \left\{P(w)P(f_j = pass|w)EU(F_j = test, f_j = pass)\right\} +$$
$$\sum_w \left\{P(w)P(f_j = fail|w)EU(F_j = test, f_j = fail)\right\} - c_f$$

To choose the best test, this procedure is repeated for each of the untested dice. The best test is the test that results in the maximum expected utility.

There are several key assumptions that keep the best test computation tractable.

1 The single-step lookahead assumes that only one additional functional test will be performed and then the packaging decisions will be made. This approximation is a form of greedy search and requires experimental verification.

2 The inking decisions for the dice are independent. This assumption is not an approximation, since this is how actual inking is performed. In this case it is just a fortuitous circumstance.

3 Testing actions do not change the state of the wafer; they only provide information about the wafer state. This is an approximation, because in rare cases, the testing procedure may damage a die. But this possibility is ignored in this analysis.

The test stopping and wafer disposition decisions are relatively simple given the expected utility calculations that were performed during the course of determining the best die to test. Testing continues as long as there is one functional test that has a higher expected utility estimate than the expected utility of stopping. Computing the expected utility of stopping involves the following steps:

1. Compute the package test probabilities for each of the dice. For any die this is accomplished by $P(p_i) = \sum_{w,f_i} P(w)P(f_i|w)P(p_i|f_i)$.

2. Multiply these probability values by the appropriate utility parameters. This yields two expected utility measures for each die. These correspond to the options to ink and to not ink.
3. For each die choose the inking action with the highest expected utility.
4. Sum all of these values and subtract the wafer handling costs.

The resulting value is the expected utility that would be realized if testing was stopped and packaging was performed with the current information. The difference between the expected utility associated with the best test and the current expected utility is the value of information. If this value is non-positive, then testing is terminated. Notice that in computing the expected utility of stopping, the inking decisions for all dice are made. Thus the only decision left is the wafer disposition decision, and this is trivial given the expected utility values computed during the test stopping decision. If the expected utility of stopping minus the wafer handling cost is positive, then the wafer is shipped.

3.3.5 Inference with Ensembles of Models

The modeling and inference methods presented so far assume a single model. An alternative is to employ an ensemble of wafer models. The motivation for an ensemble approach is that for some problems it is better to combine the results from several models than it is to rely on a single model. The primary requirement for this approach to be beneficial is that the models must have uncorrelated errors. By combining models that have uncorrelated errors, it is sometimes possible to produce better results than those produced by any one of the component models. For some problems it may be impossible or impractical to generate a single model with acceptable performance. In these cases a practical alternative may be to generate an ensemble of lower-quality models whose combined performance is acceptable (Smyth and Wolpert, 1997).

For the wafer test problem, the ensemble approach involves generating a suite of parameterized belief nets. Given an ensemble of this form, inference is performed by having each of the models in the ensemble vote on the result. The combined votes from the models serves as the basis for the wafer test decisions. In the wafer test problem, models vote on probability values for inference queries. Each model votes with equal weight.

As an example, consider the formula for computing the package test probabilities for a single die.

Let M represent the ensemble of models.
Let m_j represent a single model in the ensemble.
Let s represent the size of the ensemble, $s = |M|$.
All probability distributions are now explicitly conditioned on a model variable, e.g., $P(w|m_j)$ represents the wafer class distribution associated with model j.

Then the formula for the package test probabilities is

$$P(p_i) = \frac{1}{S} \sum_{w,f_i} \sum_{m_j \in M} P(w|m_j) P(f_i|w,m_j) P(p_i|f_i,m_j)$$

The other inference formulas are modified in a similar manner to work with an ensemble of models.

3.4 Summary

In this chapter, principles from statistical decision theory were employed to formulate the wafer test problem as a collection of maximum expected utility decisions. Probabilistic inference techniques were applied to derive computable solutions to these decisions. This included belief nets for representing stochastic models, influence diagrams for representing decisions, and inference algorithms for computing probabilities and expected utilities. Taken together, these elements define a selective test approach to the optimal test problem. The next chapter describes the machine learning methods that were developed to acquire the parameters to the belief network distributions.

4. LEARNING METHODS FOR STOCHASTIC MODELS

The decision models that were developed for the wafer test problem are composed of two major components, a stochastic model of wafer test results and a utility model over processing costs. The parameters to the utility model are provided by the manufacturer. The stochastic model parameters must be generated. In this chapter the methods that were developed to generate the stochastic model parameters are described and results are presented that show how well these methods perform.

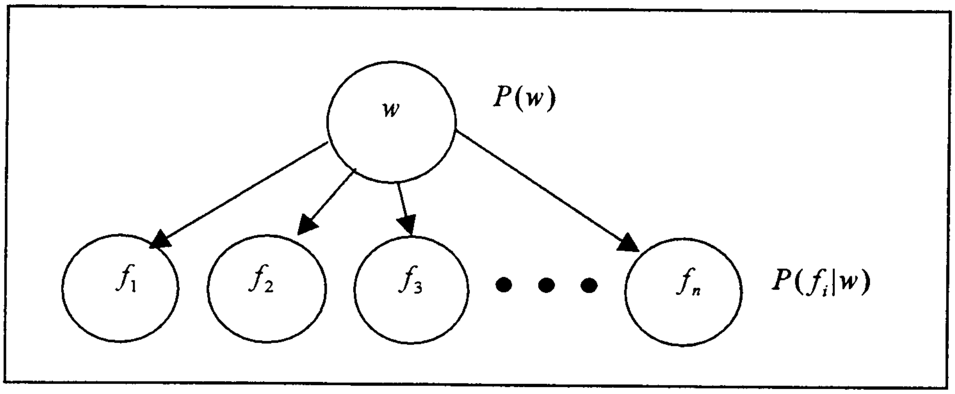


Figure 15: Wafer Test Belief Network

Figure 15 depicts a belief network representation of the stochastic model of wafer test results. The root node corresponds to the wafer class. Each of the leaf nodes corresponds to the functional test of a specific die. There is a probability distribution associated with each node, and the entries to these distributions are the parameter values that must be determined. In this study, the distribution entries are generated by analysis of historical test data with machine learning algorithms. This historical test data contains the observed results from the functional tests of numerous wafers, however it does not contain the wafer class information. Thus the parameter estimation problem is one that contains missing values, and the stochastic model is one that has a hidden, or latent, variable (the wafer class, w). This latent variable makes the parameter estimation problem more challenging. In this study, data-augmentation algorithms are applied to the parameter estimation problem for the wafer class models, specifically, the EM algorithm and the Gibbs sampling algorithm.

The remainder of this chapter is organized into the following sections:

> 4.1 Measuring Model Quality. The sample likelihood and negative log likelihood measures are defined and the rationale for their application is discussed.
>
> 4.2 Incomplete-Data Problems and Data-Augmentation Algorithms. A brief overview of incomplete-data problems and data-augmentation algorithms is presented.
>
> 4.3 The EM Algorithm for the Wafer Test Models. The EM algorithm is introduced and developed for estimating the parameters to the wafer test model.
>
> 4.4 The Gibbs Sampling Algorithm for the Wafer Test Models. The Gibbs sampling algorithm is introduced and developed for estimating the parameters to the wafer test model.
>
> 4.5 Experimental Tests. The EM and Gibbs sampling algorithms are tested on the parameter estimation task for the wafer test models. Results are presented and performance is summarized.

4.1 Measuring Model Quality

The general approach to solving the wafer test problem is to train a stochastic model on historical wafer test data, then combine the trained model with the utility model to produce a parameterized decision model. This decision model is then embedded within a selective test control structure and testing proceeds. Performance of the entire system is measured in terms of net profit, testing accuracy, and the system's responsiveness to the wafers tested. Thus, these test performance measures will provide the ultimate evaluation of the quality of the learned models. During the development phase, an estimate of these performance measures is needed. For decision problems, the obvious candidate is expected utility, however in practice surrogate measures such as mean square error, cross entropy, classification rates, Bayes Factors and model likelihood are often employed (Heckerman, Geiger, and Chickering, 1995; Kass and Raferty, 1994). There are two motivations for these surrogate measures. First, utility measures are difficult to construct. Second, it is often convenient to develop the trained models apart from the rest of the decision structure. For the wafer test problem, the expected utility calculations require a relatively complex computation. Therefore the performance measure for the parameter estimation algorithms is initially provided by a model likelihood score. The testing performance of the models is evaluated in a separate set of tests.

Model likelihood provides a relative probabilistic measure of how well the model fits the data (Madigan, et al. 1994; Buntine, 1994; Buntine, 1996; Dietterich, 1997; Heckerman, 1997). Model likelihood is defined as follows.

Notation:
$\mathbf{D} \equiv$ data sample $\equiv$ training set
$D_j \in \mathbf{D} \equiv$ single training example, (i.e., test results for each die on the training wafer)
$S_m \equiv$ model structure
$\Theta_m \equiv$ model parameters

For the wafer test problem, S_m, refers to the belief network in figure 15, and Θ_m refers to the entries in the corresonding probability distributions. The training set is the historical wafer test data.

Given this notation and assuming that the training examples are independent and identically distributed, then the model likelihood can be defined by the following formula.

Model likelihood: $P(\mathbf{D}|S_m,\Theta_m) = \prod_{D_i \in \mathbf{D}} P(D_i|S_m,\Theta_m)$

This provides a measure of how well the model predicts the data. In practice, logs are employed to keep the values within reasonable limits and the resulting value is negated to yield positive values. This final measure is referred to as the negative log likelihood and is computed as follows:

$$NL(\Theta_m) \equiv -\log(P(\mathbf{D}|S_m,\Theta_m)) = -\sum_{D_i \in \mathbf{D}} \log(P(D_i|S_m,\Theta_m))$$

This is the form of the likelihood measure that is used to measure model quality in this study. Thus, the objective of the learning algorithms is to find a set of model parameters that minimizes the negative log likelihood score.

Within the machine learning framework, parameter estimation often involves two data samples assumed to be drawn from the same population. One data sample is employed to train the model; the other is used to measure performance. This approach is referred to as holdout-validation and its purpose is to ensure that the trained models generalize and haven't been overfit to the training data (Smyth, et al., 1998; Russell and Norvig, 1995). In this study, holdout-validation is employed as part of the model development strategy.

In separating the measure of model quality from actual task performance, in this case wafer tests, an interesting question arises: how do likelihood measures relate to actual testing performance? One aspect of this study is the evaluation of learned models with respect to utility, that is, within the context of a decision-making task. In the wafer test problem, the utility is measured as net value and the units are dollars. The relationship between model likelihood and wafer test profits is one of the issues explored in chapter 5.

4.2 Incomplete-Data Problems and Data Augmentation Algorithms

An incomplete-data problem is a problem for which the data values for some variables are unavailable, either because these values are missing or because some model variables correspond to non-observable entities. In the wafer test problem, the wafer class values are missing, and the reason that they are missing is that wafer classes are unobservable. If the wafer class values were available, then the maximum likelihood estimates of the model parameters would be simple to determine. Their absence makes the parameter estimation problem more difficult and necessitates a more complex solution. A general approach to solving incomplete-data problems is provided by data-augmentation algorithms (Tanner, 1991; Tanner and Wong, 1987; McLachlan and Krishnan, 1997; Neal, 1993; Dietterich, 1997).

The basic idea behind data-augmentation algorithms is that incomplete-data problems can be translated into complete-data problems by augmenting the observed data with estimates of the missing values. The resulting problem can then be solved by conventional methods for complete-data problems. More formally, Tanner (1991) states the Principle of Data Augmentation as follows:

> Augment the observed data Y with latent data Z so that the augmented posterior distribution $P(\Theta|Y,Z)$ is "simple". Make use of this simplicity in maximizing/marginalizing/calculating/sampling the observed posterior $P(\Theta|Y)$.

The class of data-augmentation algorithms includes the EM algorithm, the Gibbs Sampling algorithm, Tanner's Data Augmentation algorithm, and the Sampling-Importance Resampling (SIR) algorithm. Tanner (1991) and McLachlan and Krishnan (1997) discuss the relationships among these algorithms in detail.

For this study, the EM algorithm and the Gibbs Sampling algorithm are developed to generate parameters for the wafer test models. From a machine learning perspective, the application of data-augmentation algorithms to wafer test parameter estimation is an example of unsupervised machine learning.

4.3 The EM Algorithm for the Wafer Test Models

The Expectation Maximization (EM) algorithm is a general-purpose iterative algorithm for maximum likelihood estimation (MLE) (Dempster, Laird, and Rubin, 1976; McLachlan and Krishnan, 1997). The EM algorithm is a common statistical procedure that has been applied to a variety of statistical models including regression, finite mixtures, hidden Markov, and latent class models. For many problems, the EM algorithm provides accurate and efficient solutions and is straightforward to implement.

There are two steps to the EM algorithm, the expectation step (E-step) and the maximization step (M-step). In the E-step, the algorithm computes the expected sufficient statistics of the hidden variables. In the M-step, the algorithm computes the maximum likelihood estimates for the model parameters. In data-augmentation terms, the E-step augments the observed data with the expected sufficient statistics of the missing values, and the M-step solves the resulting complete-data problem. The EM algorithm iterates over these two steps until the parameters converge. Within a Bayesian framework, maximum *a posteriori* (MAP) estimates are possible with the EM algorithm (McLachlan and Krishnan, 1997; Cheeseman, et al., 1988). The extension to MAP estimates is handled by the inclusion of a prior density in the M-step of the EM algorithm. The basic operations are the same for both types of estimates.

The primary criterion for the applicability of the EM algorithm is that there exists a mapping from the incomplete-data problem to a complete-data problem for which MLE is feasible. The E-step takes expectations over the complete-data conditional distributions and the M-step performs the complete-data maximum likelihood (ML) estimation. For members of the exponential family of distributions, and for discrete multinomial distributions in particular, these operations often have closed-form solutions. Under these conditions, the EM algorithm produces reliable global convergence with monotonic increases in likelihood (McLachlan and Krishnan, 1997). These characteristics make the EM algorithm an attractive candidate for solving the parameter estimation problem for the wafer test models.

With incomplete data problems, the model variables can be partitioned into those that are observable and those that are not. Let O represent the set of observed variables, and let U represent the set of unobservable variables. Since the wafer test models assume a fixed model structure, i.e., the naïve Bayes' belief network of figure 13, the model structure parameter is omitted from the following formulas.

The goal is to generate a set of model parameters that minimizes the negative log likelihood of the observed variables. This value is computed as follows:

$$NL(\Theta_m) = -\log P(O|\Theta_m) = -\log \sum_U (P(U,O|\Theta_m))$$

With these definitions, the EM algorithm can be described as follows.

EM Algorithm:

```
Guess at initial parameters $\Theta_{\mathrm{m}}{}^{(0)}$
E-step:  compute $P(U|O,\Theta_{\mathrm{m}}{}^{(t)})$
M-step:  set $\Theta_{\mathrm{m}}{}^{(t+1)}$ to maximize $E[P(U,O|\Theta_m^{(t+1)})]$
```

The EM algorithm begins with a guess at the initial model parameters, $\Theta_m^{(0)}$, and then iterates between the E-step and the M-step. On each iteration, the parameters are updated in the M-step, $\Theta_m^{(t)} \to \Theta_m^{(t+1)}$. The EM algorithm terminates at convergence, i.e., when $NL(\Theta_m^{(t)})$ reaches a local minimum.

A detailed description of the EM algorithm for the wafer test models requires additional notation. Let $D_i \in \mathbf{D}$ represent a complete set of test values for a single wafer, (i.e., one test result for each die on the wafer). Let $d_{i_j} \in D_i$ represent the result of testing die *j* on wafer *i*. These wafer test result values provide an instantiation for each of the functional test variables in the model. The wafer classes for each of the training examples are unknown. Thus the E-step of the EM algorithm augments each example with a conditional distribution over wafer classes. For each training example $D_i \in \mathbf{D}$, compute the following wafer class distribution:

$$\begin{aligned} P(w_i|D_i,\Theta_m) &= \frac{P(D_i|w_i,\Theta_m)P(w_i|\Theta_m)}{P(D_i|\Theta_m)} \\ &= \alpha P(D_i|w_i,\Theta_m)P(w_i|\Theta_m) \\ &= \alpha P(w_i|\Theta_m)\prod_j P(d_{i_j}|w_i,\Theta_m) \end{aligned}$$

Where α is a normalization operator.

The M-step updates the wafer class and functional test distributions. First, update the wafer class distribution:

$$P(w|\Theta_m) = \alpha\sum_i P(w_i|D_i,\Theta_m)$$

Second, update the functional test distributions. For each functional test, compute the following distribution:

$$P(f_j|w,\Theta_m) = \frac{P(f_j,w|\Theta_m)}{P(w|\Theta_m)} = \frac{P(w|\Theta_m)P(f_j|\Theta_m)}{P(w|\Theta_m)}$$

Note that $P(f_j|\Theta_m)$ is either 0 or 1 depending on the test result for f_j. So this computation is equivalent to the following:

Let *a* be defined as the sum of the $P(w_i)$ for each wafer *i* where $f_j = d$.

Let *b* be defined as the sum of the $P(w_i)$ for all wafers.

Then $P(f_j|w,\Theta_m) = \frac{a}{b}$.

This completes the development of the EM algorithm for estimating parameters to the wafer test models. Experiments exploring the performance of this algorithm are described in section 4.5.1. The next section presents a similar development of the Gibbs sampling algorithm for the parameter estimation task.

4.4 The Gibbs Sampling Algorithm for the Wafer Test Models

The Gibbs sampling algorithm presents an alternative method for generating the parameters to the wafer test models. Gibbs sampling is a Markov chain Monte Carlo method that is useful for generating random samples from joint distributions that are difficult to sample directly (Geman and Geman, 1984; Gefland and Smith, 1990; Buntine, 1996; Neal, 1993, Dietterich, 1997). The key insight is that knowledge of conditional distributions is sufficient to determine a joint distribution (Casela and George, 1992). The Gibbs sampling algorithm exploits this idea by sampling conditional distributions to provide values for complex densities. Thus, the Gibbs sampling algorithm is most applicable to density estimation problems where the conditional distributions are easy to sample (Neal, 1993, McLachlan and Krishnan, 1997). Like the EM algorithm, the Gibbs sampling algorithm is a practical approach and has been applied to numerous problems, including generalized linear models, mixture models, logistic regression, and latent-class models (Casela and George, 1992; McLachlan and Krishnan, 1997). There is a close relationship between the EM and Gibbs sampling algorithms, and it is commonly noted that the EM algorithm is an approximation to the Gibbs sampling algorithm (Buntine, 1991; Dietterich, 1997). The EM algorithm is often more efficient (i.e., faster), but less accurate.

In general terms, the Gibbs sampling algorithm can be described as follows. Assume that there is a set of random variables, $\mathbf{X} = \{X_1, X_2, \ldots, X_k\}$, whose joint distribution is of interest, $P(\mathbf{X}) = P(X_1, X_2, \ldots, X_n)$. Assume further that the conditional distributions for each of the X_i are known, $P(X_i | \mathbf{X} - \{X_i\})$. Then start with arbitrary initial values for each of the X_i, $\mathbf{x}_0 = \{x_{1,0}, x_{2,0}, \ldots, x_{n,0}\}$, i.e.,

$X_1 = x_{1,0}, X_2 = x_{2,0}, \ldots, X_n = x_{n,0}$. Next, for each variable X_i sample a new value $x_{i,1}$ from the following conditional distribution:

$P(X_i | X_1 = x_{i,1}, \ldots, X_{i-1} = x_{i-1,1}, X_{i+1} = x_{i+1,1}, \ldots, X_k = x_{k,1})$

This yields a new sample $\mathbf{x}_1 = \{x_{1,1}, x_{2,1}, \ldots, x_{n,1}\}$. As long as distribution entries are bounded away from 0, iterations of this procedure will produce an empirical distribution of the $\mathbf{x}_i$ that converges to the desired joint distribution. In order to measure the parameters of the joint distribution, the Gibbs sampling algorithm is employed to generate a large sample of values. These values are used to compute the parameters of the joint distribution. Hence, Gibbs sampling does not directly generate the parameters of interest, but only a sample drawn from the distribution.

For the wafer test problem, the Gibbs sampling algorithm augments each training example with a specific wafer class assignment, rather than a distribution over wafer classes as in EM. The following description is based on the development by Dietterich (1997).

Let Θ_m represent the set of model parameters. Let w represent the unobservable wafer class variable. Using Gibbs sampling, we will generate samples from the joint distribution. From these samples, we can extract an ensemble of values for the parameters, Θ_m. We will then apply the ensemble techniques described in the previous chapter to make probabilistic inferences. The Gibbs sampling process works as follows: Begin by choosing initial random values for the model parameters, Θ_m. Then iterate over the following two steps until convergence.

1. Augment each of the training examples, $D_i \in \mathbf{D}$, with a wafer class assignment drawn randomly according to $P(w|\Theta_m, D_i)$. Since the current values of Θ_m and the observed values for D_i are known, computing $P(w|\Theta_m, D_i)$ is straightforward. This yields a class assignment (an instantiation of the wafer class variable, w) for each $D_i \in \mathbf{D}$. Let $\{c_1, c_2, \ldots, c_n\}$ represent these class assignments.

2. Generate new values for the parameters by drawing them from the appropriate conditional distribution.

The model parameters can be partitioned into those associated with the wafer class distribution and those associated with the functional test distributions, $\Theta_m = \{\Theta_{m_w} \cup \Theta_{m_f}\}$.

First update the parameters for the wafer class distribution:

Let Θ_{m_w} represent the parameters for the wafer class distribution, $P(w)$, and let q_c be the distribution entry corresponding to the probability of wafer class *c*. Let n_c represent the number of training examples currently assigned to class *c*. Then assuming uniform priors over q_c, the posterior probability is a beta distribution with parameters $(n_c+1, n-n_c+1)$. Thus $\beta(n_c+1, n-n_c+1)$ can be sampled to provide a value for q_c. Various algorithms exist for sampling from the beta distribution (Devroye, 1986). This process is performed for each of the wafer classes.

Next, update the parameters to the functional test distributions:

Let Θ_{m_f} represent the parameters to the conditional distributions for functional test results, $P(f_i|w), i = 1,\ldots,n$. Let q_{jvc} be the parameter in Θ_{m_f} that represents the probability that the *jth* feature (test result) will have value v when generated from wafer class *c*, i.e., $q_{jvc} \equiv P(f_j = v|w = c)$. Let n_{jvc} represent the number of training examples with a class assignment of *c* in which the *jth* value is *v*. Assuming uniform priors for $P_{q_{jvc}}$ then $P_{q_{jvc}}$ is distributed as a beta distribution with parameters $(n_{jvc}+1,\ n_c - n_{jvc}+1)$. Thus $\beta(n_{jvc}+1,\ n_c - n_{jvc}+1)$ can be sampled to provide the parameters for $P(f_i|w)$. This procedure is repeated to generate the parameters for all functional tests and all wafer classes. This completes one iteration of the Gibbs sampling algorithm and results in a new set of parameters for the wafer test model.

Successive iterations of this procedure produce additional sets of parameters and the collection of parameters generated over a number is iterations is referred to as the Gibbs sequence.

Unlike the EM algorithm, the Gibbs algorithm does not converge to a single parameter vector. This is a reflection of the Monte Carlo aspect of the Gibbs sampling algorithm in which class assignments for the training samples are made probabilistically. In practice, rather than selecting the parameters generated on the last iteration, a collection of parameters is chosen from the Gibbs sequence. This results in an ensemble of parameters, which define an ensemble of models. In notation, an ensemble of size s is denoted by $M = \{\Theta_{m_1}, \Theta_{m_2}, \ldots, \Theta_{m_s}\}$.

There are a number of strategies for selecting the Gibbs ensemble (Casela and George, 1992; Tanner, 1991; Dietterich, 1997; Neal 1993). The general approach is to allow the Gibbs sampling algorithm to iterate for a fixed number of steps and then to collect elements from the Gibbs sequence at regular intervals. For example, in the Markov random field application of Longtin, et al. (1996) 10,000 initial iterations were performed and then every *nth* element of the Gibbs sequence after that point was collected, where *n* was on the order of 1000. For the wafer test models a more modest strategy was employed. In this strategy, only 30-50 initial iterations were performed and then every element of the Gibbs sequence after that point was collected until the desired ensemble size was reached. In general this is not a recommended approach, since for some problems it is possible to get trapped in a nonoptimal subspace (Tanner, 1991). However for the wafer test problem, empirical tests failed to elicit this bad behavior. The performance measure for a Gibbs ensemble on the learning task is the negative log likelihood of the ensemble. Assuming that M represents an ensemble of models and that there are s models, i.e., $s = |M|$, then the negative log likelihood measure for M is defined as follows. Let $P(O, U, \Theta_m)$ be the joint distribution of the observables, the unobservables, and the model parameters.

$$
\begin{aligned}
P(O) &= \sum_{U,\Theta_m} P(O,U,\Theta_m) = \\
&\sum_{U,\Theta_m} P(O|U,\Theta_m)P(U|\Theta_m)P(\Theta_m) = \\
&\frac{1}{S}\sum_{U,\Theta_m} P(O|U,\Theta_m)P(U|\Theta_m)
\end{aligned}
$$

For an entire data sample, this becomes $\sum_i -\log\left[\frac{1}{S}\sum_{U,\Theta_m} P(O_i|U_i,\Theta_m)P(U_i|\Theta_m)\right]$. This is just the usual likelihood computation for model Θ, but it will involve a product over all the dice, i.e., $P(O|U,\Theta_m) = \prod_j P(f_j|U,\Theta_m)$. This formulation captures the assumption that each model in the ensemble votes with equal weight and corresponds to the realization that the Gibbs algorithm generates models in proportion to their likelihood. More likely models occur more often in the Gibbs sequence (Dietterich, 1997).

4.5 Learning Experiments

In order to determine how well the learning algorithms performed on the parameter estimation task and also to identify the best models to employ in the selective test policy, a number of learning experiments were performed. Each test consisted of instantiating the naïve Bayes' network model of wafer test results with a specific number of wafer classes. The model parameters were then initialized to random values. Next each model was trained on a historic wafer data set that contained the functional test results from 600 wafers. After training, each model was then evaluated on a separate data set that contained the functional test results from an additional 600 wafers. This data set is referred to as the "validation" set. The results from the learning trials on the training and validation data sets are used to make design decisions about model structure and training policy. A third data set is employed in the wafer test simulations in the following chapters. This data set is referred to as the "test" data set. In this chapter the performance on the training and validation data sets is measured in terms of negative log likelihood. The results are presented in the sections below.

4.5.1 EM Tests

In order to explore the behavior of the EM algorithm on the task of learning parameters for the wafer test models, a number of tests were performed. For these tests, eight models were constructed by instantiating the naïve Bayes' network with different wafer class dimensions. Wafer class distributions of the following sizes were considered: 1,2,4,8,12,16,20,24.

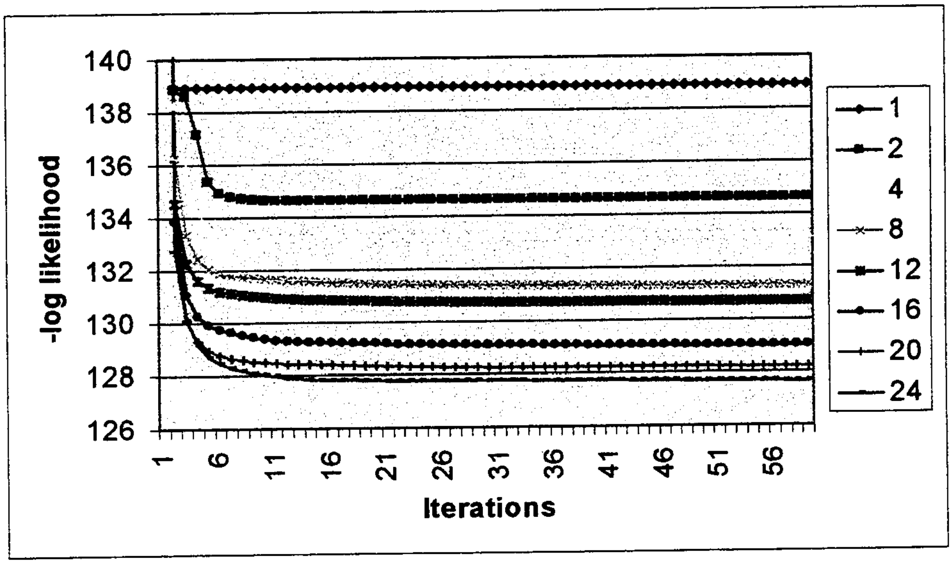


Figure 16: EM Learning Curves for Training Data

Figure 16 presents the results from the EM learning trails on the training data set. The results show that as the number of wafer classes increases, the negative log likelihood decreases, indicating a better fit to the data. There is a perfect negative relationship between the number of wafer classes and the negative log likelihood score on the training data. The learning curve for the 1-class model indicates a significantly poorer fit to the training data. Recall that the 1-class model effectively encodes an assumption of independence among the dice, whereas the *n*-class models encode an assumption of conditional independence. These learning curves suggest that conditional independence provides a significant improvement in modeling accuracy for the wafer test data.

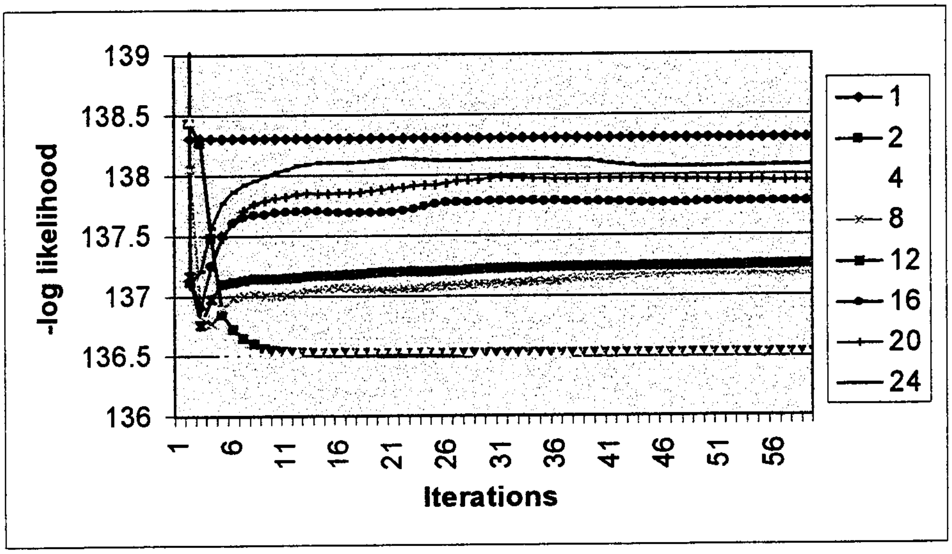


Figure 17: EM Learning Curves for Validation Data

Figure 17 presents the results from the EM learning trials on the validation data set. This indicates that the best fit to the validation data was achieved with four wafer classes. For models with more than four classes, the EM algorithm overfit the training data as indicated by the higher negative log likelihood score. Note that the single-class model performed poorly on the validation set as well. This confirms that the additional expressiveness provided by the conditional independence assumption of the *n*-class models results in better overall predictions for the wafer class data.

In order to determine the structure of the wafer class distributions that were generated by the EM algorithm, models with four and eight wafer classes were trained 100 times on the training set of 600 wafers. After each training trial, the wafer class probabilities were sorted from lowest to highest value. The sorted values from all trials were averaged together. Thus, the probability for class 1 is the average prior probability value of the least likely wafer class. The probability for class 2 is the average prior probability value of the second least likely wafer class. The probabilities for the other classes are defined similarly. These results are presented in the figures 18 and 19, respectively. The actual average probability values are listed across the top of the chart, e.g., .05347 is average prior probability value for the least likely wafer class in the 4-class model.

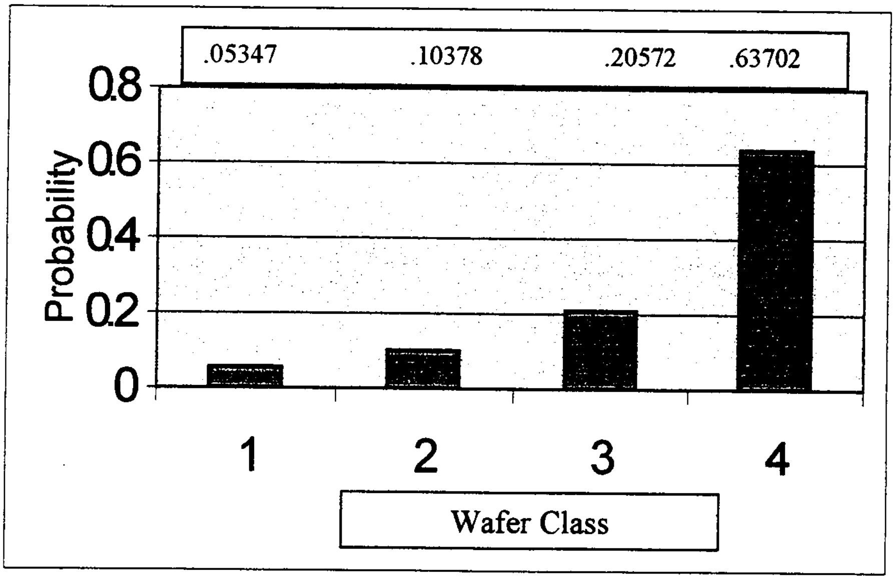


Figure 18: EM Wafer Class Distribution 4 Classes

Figure 18 depicts the average wafer class distribution over 100 training sessions of the EM algorithm with a model that contained four wafer classes. The results show that one wafer class dominated the probability distribution, accounting for over 63% of the mass.

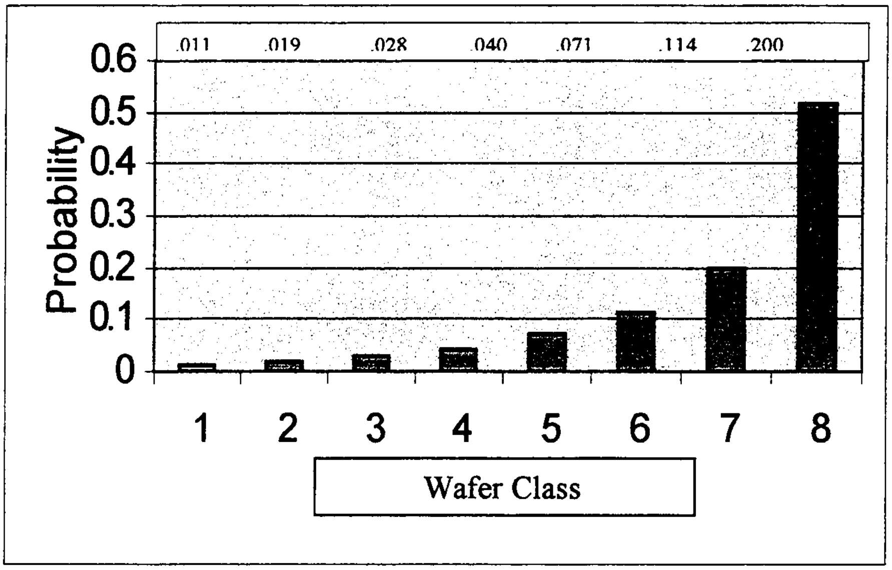


Figure 19: EM Wafer Class Distribution 8 Classes

Figure 19 depicts the average wafer class distribution over 100 training sessions of the EM algorithm with a model that contained eight wafer classes. As with the 4-class model, the results show that one wafer class dominated the probability distribution, in this case accounting for approximately 52% of the mass. This suggests that most of the training examples shared a common stochastic pattern, however this doesn't suggest that a single-class model is an adequate representation of wafer test results. In fact, even though these tests demonstrate that a single class typically dominates the resulting wafer class distributions, they also demonstrate that significant mass is associated with the non-dominant classes. Approximately 37% of the mass of the wafer class distribution for the 4-class models and 48% of the mass of the wafer class distribution for the 8-class models were allocated to the non-dominant classes. As indicated by the likelihood scores, the additional wafer classes provide a significant improvement in predictive performance. The fact that the wafer class distributions were skewed towards one class is only of marginal interest or significance.

There are several important results from the EM tests. First, the EM algorithm works. The learning curves indicate that the EM algorithm was able to extract a signal from the wafer test data. Second, the best performance, as determined by holdout negative log likelihood, was achieved with four wafer classes. Third, the EM algorithm was efficient. Convergence was achieved within 30 iterations. These results suggest that the EM algorithm provides an appropriate method for generating parameters to the wafer test models. The performance of the learned models on wafer tests is explored in chapter 5.

The next section presents the results from similar training experiments with the Gibbs sampling algorithm.

4.5.2 Gibbs Sampling Tests

For the Gibbs sampling algorithm, two sets of training experiments were performed. The tests of the first set were similar to those performed with the EM algorithm. In these tests, six individual models were trained, where each model was instantiated with a different number of wafer classes. Models with 1, 2, 4, 8, 16 and 20 classes were created. Each model was created by running the Gibbs sampling algorithm for a fixed number of iterations, and then the next parameter vector was retained for testing and the process was halted. For ensembles with multiple models, this process was modified so that the last *m* models were retained. Performance was measured in terms of negative log likelihood on the training and validation data sets. For the second set of tests, the effects of ensemble size were investigated. In these tests, all models were instantiated with four wafer classes and ensembles of various sizes were created. As before, performance was measured in terms of negative log likelihood on the training and validation data sets. The results from both sets of tests are presented in learning curves and these are described next.

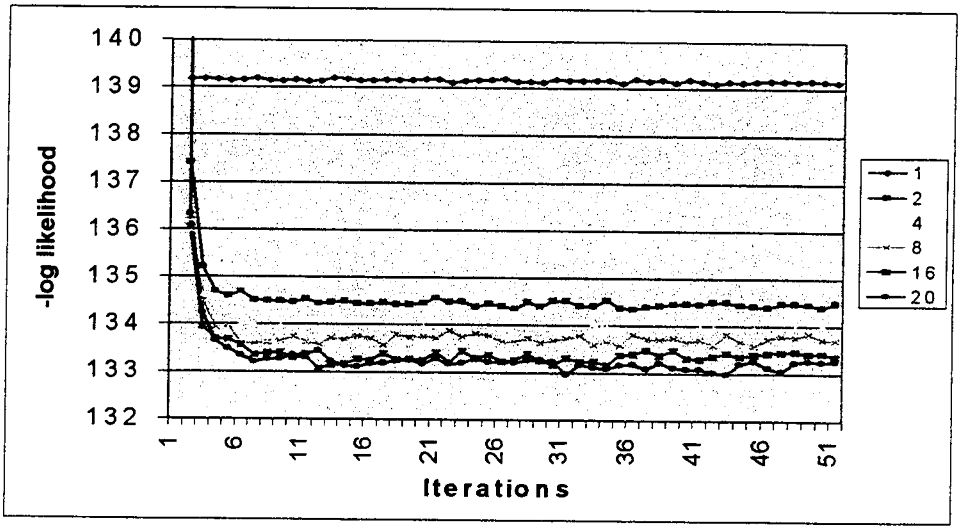


Figure 20: Gibbs Learning Curves for Training Data

Figure 20 presents the learning curves for six models trained with the Gibbs algorithm. Individual curves correspond to models with different size wafer distributions, 1, 2, 4, 8, 16 and 20. Performance is measured as negative log likelihood of the training data. There are several points to note. First, as with the EM algorithm, the models converge quickly. Second, the single-class model performed significantly poorer than the other models. This also corresponds to the EM results. Third, convergence was not monotonic. This is the expected behavior for the Gibbs sampling algorithm. Fourth, the final values for the Gibbs-trained models were not as low as the values for the EM-trained models. For example, the lowest negative likelihood score for any of the Gibbs-trained model on this task was 132.99 with a model containing 20 classes. The EM-trained model with 20 classes achieved a value of 128.22 on the identical task. Recall that lower values indicate better fit. Thus for a single model, the EM algorithm produces parameters that fit the training data better than those produced by the Gibbs sampling algorithm. The next graph considers the performance of the same Gibbs-trained models on the validation data set.

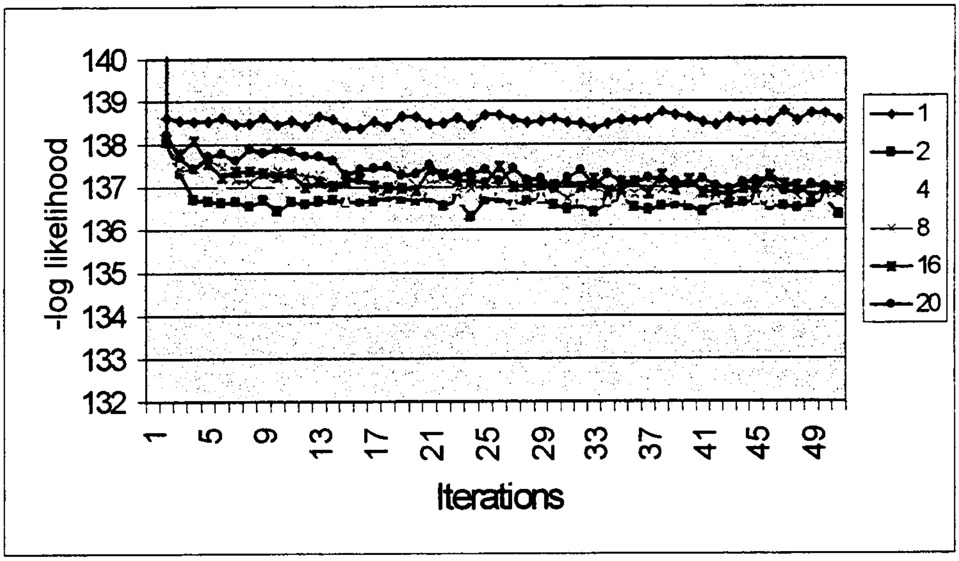


Figure 21: Gibbs Learning Curves for Validation Data

Figure 21 presents the results of the Gibbs-trained models on the validation data set. As with the EM-trained models, the validation set values exceed the training set values. However, unlike the test set values for the EM-trained models, the test set values of the Gibbs-trained models exhibit less overfitting. The best test set likelihood was achieved with 2-class model, with similar values produced with the 4-class model. As with the EM algorithm, increasing the number of wafer classes beyond eight resulted in poorer performance; however the effect is not as significant. It is also interesting to note that the best performance of a Gibbs-trained model on the test set was a negative likelihood score of 136.29. This score was achieved with a model that contained 2 classes. The best score on this task for an EM-trained model was 136.48, with a model that contained 4 classes. Thus, the Gibbs algorithm produced a single model that outperformed the best EM-trained model on test set fit, although the differences in negative log likelihood are negligible. Considering that the Gibbs algorithm is performing a randomized search in parameter space it is not surprising that it occasionally hits upon a particularly good fit. In general, determining the optimal stopping point in the Gibbs sequence from consideration of training scores is impossible. In practice, a sample of models from the Gibbs sequence is chosen and these models are combined to produce a composite model that outperforms each of the individual models. This is the idea behind ensemble methods. It is interesting, and somewhat counter-intuitive, to note that averaging models can produce above-average results. The next set of tests demonstrates this behavior by constructing a Gibbs ensemble for the wafer test models.

A Gibbs ensemble for the wafer test model was created by the following method. Thirty initial iterations of the Gibbs algorithm were performed. Then an additional 30 iterations were performed and on each of these iterations the parameters were collected. This resulted in an ensemble of 30 models. Performance was measured on each iteration by means of the negative log likelihood score. For each of the first 30 iterations this process produced two likelihood scores, one for the training set and one for the validation set. For iterations 31 through 60 there were four performance measures per iteration:

1. single model training set negative log likelihood
2. single model test set negative log likelihood
3. ensemble training set negative log likelihood
4. ensemble test set negative log likelihood

These performance measures are presented in figure 22.

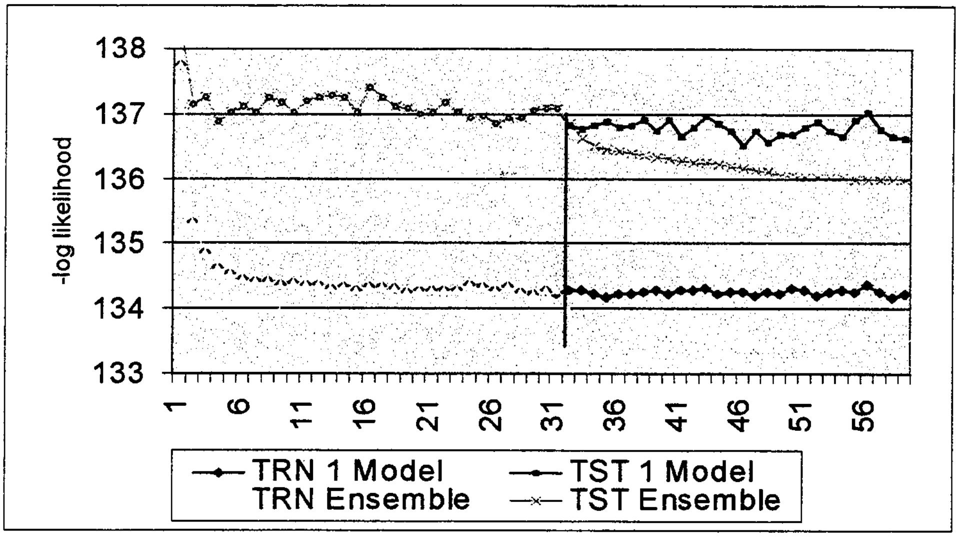


Figure 22: Gibbs Ensemble Learning Curves

Figure 20 presents the results from the ensemble tests. Notice that the performance of the ensemble is better than any single model on both the training and validation sets. Also the performance on the test set is better than that realized with the EM-trained models. The actual likelihood scores are presented in table 1.

	EM	Best Single Gibbs	Gibbs Ensemble
Training Score	127.68	134.20	133.20
Validation Score	136.48	136.29	135.99

Table 1: EM vs. Gibbs Scores

These scores suggest that the best EM algorithm, as judged by test set negative log likelihood, overfit the training data. The Gibbs ensemble outperformed the best

Gibbs model on both training and validation sets, and outperformed the EM algorithm on the validation set.

The results presented in figure 22 suggest that larger ensembles produce better results. The next set of tests explores this behavior further by generating ensembles containing from one to eighty models. As before, the number of wafer classes was held constant at four. The negative log likelihood scores for the test and training data sets are presented in the next two figures.

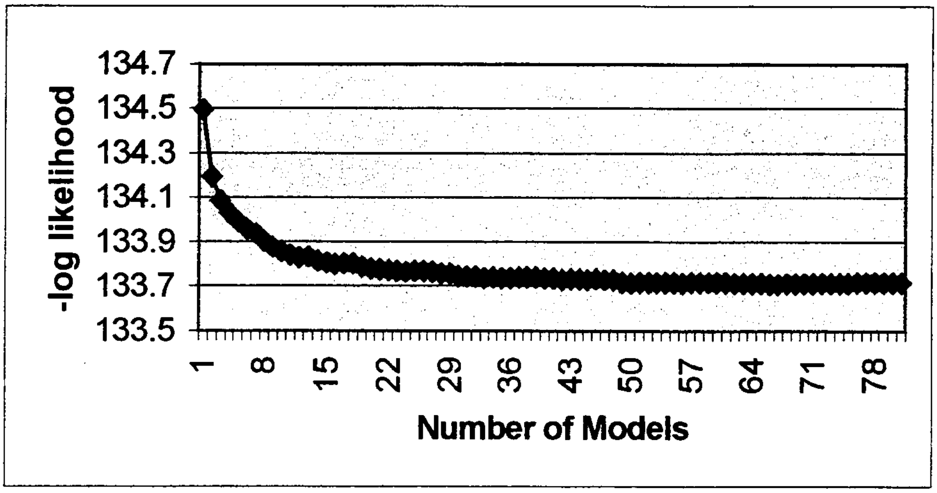


Figure 23: Gibbs Ensemble: 80 Models: Training Data

Figure 23 presents the negative log likelihood scores from the Gibbs ensembles on the training data set. The results show convergence to a minimum value of 133.72 with about 50 models. With 30 models the negative log likelihood score is within .01 of this minimum value.

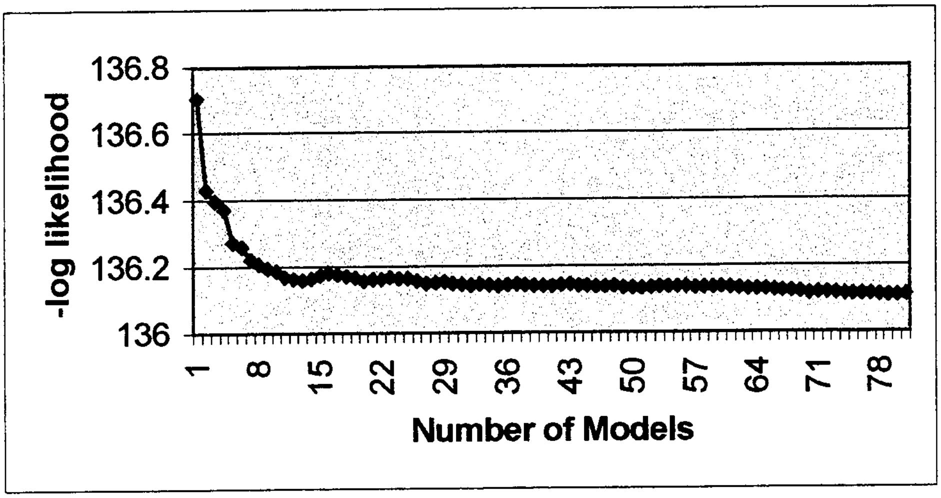


Figure 24: Gibbs Ensemble: 80 Models: Validation Data

Figure 24 presents the negative log likelihood scores for the ensemble models on the validation data set. These results show that a minimum value of 136.11 was reached with ensembles containing more than 70 models. With 10 models the likelihood score was 136.17. These results suggest that improved negative log likelihood scores can be attained by increasing the size of the Gibbs ensemble, however the actual utility of these improved scores is unclear.

4.6 Summary

In this chapter, the EM algorithm and the Gibbs sampling algorithm were applied to the parameter estimation task for the wafer test models. Performance was measured as cross-validated negative log likelihood scores. The effects of varying the wafer class dimension were investigated, and for the Gibbs sampling algorithm the effects of varying the ensemble size were examined. There are several significant results from these tests. First, both the EM algorithm and the Gibbs sampling algorithm worked. Both were able to generate parameters for the wafer class model that resulted in improved negative log likelihood scores. For the EM algorithm, the best performance was achieved with four wafer classes. Additional classes resulted in overfitting of the training data set. A single-class model performed poorly, suggesting that the conditional independence assumption embodied in the multiple-class models is of some benefit. For a single Gibbs-trained model, the results were similar to those of the EM-trained models, although the *2*-class model performed slightly better than the *4*-class model. Experiments with Gibbs ensembles indicate that negative log likelihood scores can be improved by combining models. Training tests were performed in which ensembles were generated that contained from one to eighty models. In general, performance improved as the number of models was increased, although the size of performance improvements diminished as the ensemble grew beyond 10 models. For both the EM algorithm and the Gibbs sampling algorithm, convergence was relatively fast requiring approximately 30 iterations for a single model.

The next chapter explores the question of whether these learned models are actually useful in wafer test. It also explores the adequacy of negative log likelihood as a predictor of expected utility for the wafer test problem.

5. EXPERIMENTAL WAFER TESTS

A number of experiments were performed to determine how well the learned models would perform when embedded within the decision-theoretic test structure. A variety of stochastic models were trained on wafer test data, and the performance of these models was measured on various testing scenarios. These experiments were designed to address the following questions:

1. What is the best stochastic model and how well does this model compare to the exhaustive test approach and to an optimal testing policy? Also, how well does model likelihood predict testing profit? Various EM and Gibbs Sampling models are trained, and the testing performance of these models is compared to that of the exhaustive test and optimal test policies.

2. How well does myopic value of information (VOI) perform in deciding when to stop testing? Myopic VOI stopping is compared to optimal stopping.

3. How much training data is needed for reasonable performance? The effects of varying the size of the training data sets are measured.

4. How does the system respond to abnormal wafers? Is the approach sensitive to process problems? Individual wafers representing normal and abnormal conditions are tested and system performance is evaluated.

5. How robust is the system with respect to changes in utility parameters? Do changes in utility parameters result in rational responses from the system? Various utility parameters (functional test costs and package costs) are varied, and performance is evaluated.

6. How well does this approach generalize to wafers not considered in the development phase of this research? Testing performance is measured on a separate data set that was not analyzed during system development.

Each of these questions is addressed in a separate section in this chapter.

5.1 Model Performance

In order to evaluate the performance of the proposed approach, four types of testing policies were compared through simulated wafer testing. These approaches include the current exhaustive test approach, an optimal testing policy, and selective test approaches based on stochastic models trained by EM and Gibbs Sampling.

Wafer test policies:

1. Exhaustive testing – This corresponds to the current testing policy. All dice on all wafers are tested, all dice that pass functional test are packaged, and all dice that are packaged are package tested.

2. Oracle – This is the ultimate optimal test policy. It assumes knowledge of which dice would pass functional test without performing any functional tests. The purpose of the oracle models (sometimes described as being omniscient) is to provide a measure of optimal performance. Under the oracle testing policy no functional tests are performed, but all dice that would have passed functional test are packaged, and all packaged dice are package tested.

3. Selective sampling with EM models – Test and package decisions are based on expected utility calculations from a single stochastic model trained with the EM algorithm. Myopic value of information (VOI) is employed to decide when to stop testing. All dice with non-negative expected utility are packaged. All dice that are

packaged are package tested. A variety of EM-trained models are generated by varying the number of wafer classes.

4. Selective sampling with Gibbs ensembles – Test and package decisions are based on expected utility calculations from an ensemble of stochastic models trained by Gibbs Sampling. Myopic value of information (VOI) is employed to decide when to stop testing. All dice with non-negative expected utility are packaged. All dice that are packaged are package tested. Ensembles of varying sizes are considered.

The key performance measure of a testing policy is net profit as determined by the profit model presented in figure 25. Secondary measures include the number of tests performed, the number of dice packaged, the number of true positives (good dice that are packaged), true negatives (bad dice that are rejected), false positives (bad dice that are packaged), and false negatives (good dice that are rejected).

$W_j \equiv$ wafer$_j$

$N_d(W_j) \equiv$ number of dice on wafer$_j$
$N_f(W_j) \equiv$ number of functional tests performed on wafer$_j$
$N_{f_p}(W_j) \equiv$ number of passing functional tests on wafer$_j$
$N_p(W_j) \equiv$ number of package tests performed on wafer$_j$
$N_k(W_j) \equiv$ number of dice packaged from wafer$_j$

$Y(W_j) \equiv$ package yield of wafer$_j$
$c_f \equiv$ cost of single functional test
$c_k \equiv$ cost of packaging a single die
$c_p \equiv$ cost of single package test
$c_h \equiv$ single wafer handling cost (shipping and sawing)
$v_k \equiv$ value of single good package

value of package yield $\equiv V(Y(W_j)) = Y(W_j) * v_k$
total cost of wafer functional tests $\equiv C_f = N_f(W_j) * c_f$
total cost of packaging $\equiv C_k = (N_k(W_j) * c_k) - c_h$
total cost of package tests $\equiv C_p = N_p(W_j) * c_p$
wafer profit $\equiv V(Y(W_j)) - C_f(W_j) - C_k(W_j) - C_p(W_j)$

Figure 25: Wafer Test Profit Model

5.1.1 The Best EM Model

In order to determine which EM-trained model performs best in wafer testing, a variety of EM-trained models were generated by varying the number of wafer classes. Models with 1, 2, 4, 8, 12, 16, and 24 wafer classes were created. These models were trained on 25 lots (600 wafers) of wafer test data. The trained models were embedded in the decision-theoretic selective test structure, and performance was measured on simulated testing of 25 lots (600 wafers) of wafer test data. The training and testing data sets were distinct. These data sets were the same ones that were used in chapter4. The results from these tests are presented in table 2.

Number Classes	Learning -log Likelihood	Testing -log Likelihood	Profit	Number Tests	Number Packaged	Number False Positives	Number True Negatives
1	133.6546	138.9136	1228787	7200	122227	19939	3173
2	134.6209	136.5204	1229337	8093	121685	19397	3715
4	133.2200	136.4924	1229417	8137	121635	19347	3765
8	131.5449	136.8713	1229531	8210	121560	19272	3840
12	130.6172	137.2648	1229403	8014	121682	19394	3718
16	129.6295	137.5456	1229368	7989	121706	19418	3694
24	127.8068	138.2524	1229457	7828	121720	19432	3680

Table 2: EM Tests: Results

Selected results are considered in more detail below.

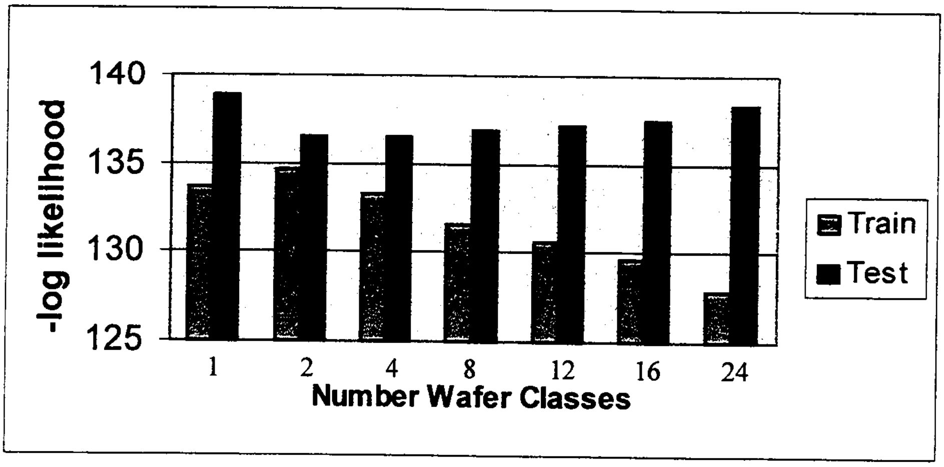


Figure 26: EM Tests: Learning Curves

Figure 26 depicts the learning curves for the EM model as the number of wafer classes is increased. These results were originally presented in chapter 4. As expected, an increase in the number of wafer classes results in a better fit to the training data and thus a lower negative log likelihood rating. The only exception to this trend is that the negative log likelihood value on the training set increases slightly when moving from one to two classes. However, after this point the negative log likelihood values decrease monotonically as the number of wafer classes is increased. The test set negative log likelihood values indicate the best fit of the test set occurs with four wafer classes. As the number of wafer classes is increased past four, the values indicate overfitting, i.e., the negative log likelihood values for the training set decrease as the values for the test set increase.

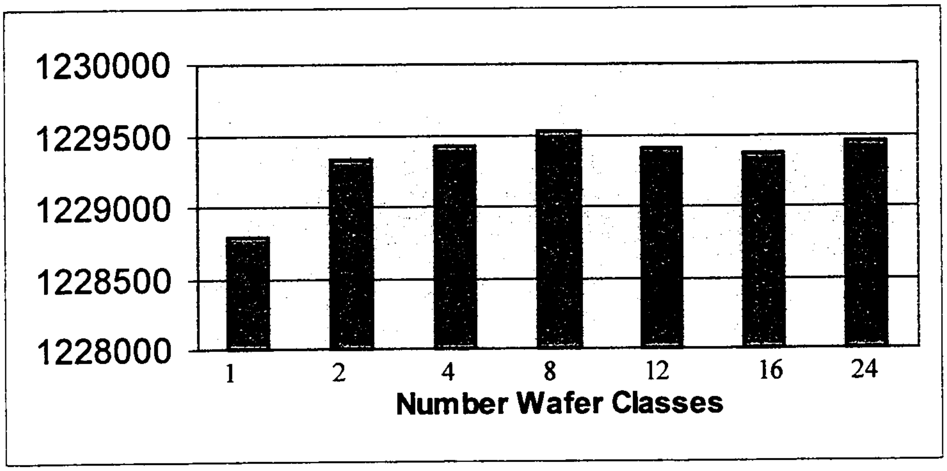


Figure 27: EM Tests: Profit

Figure 27 shows the relationship between total net profit over the test set wafers and the number of wafer classes in the model. There is a weak relationship between the number of wafer classes and profit, but the differences in profit are relatively small. The difference between the best model (8 classes, $1229531 profit) and the poorest (1 class, $1228787) is only $774. So the poorest model produces over 99% of the profits produced by the best model.

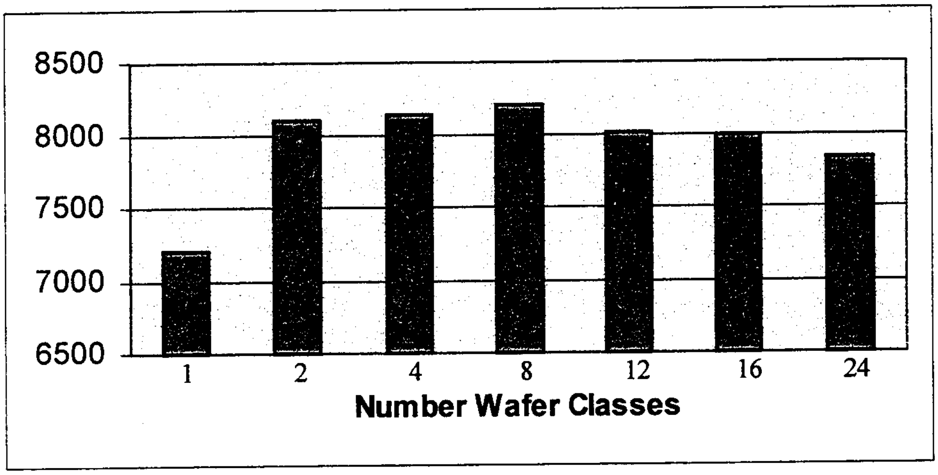


Figure 28: EM Tests: Number Tests

Figure 28 shows the relationship between the total number of functional tests performed on the test set and the number of wafer classes. In general a lower negative log likelihood score on the test set (i.e., a better fit) corresponds to a higher number of tests. The actual correlation coefficient between test set negative log likelihood and the number of functional tests is -.8996.

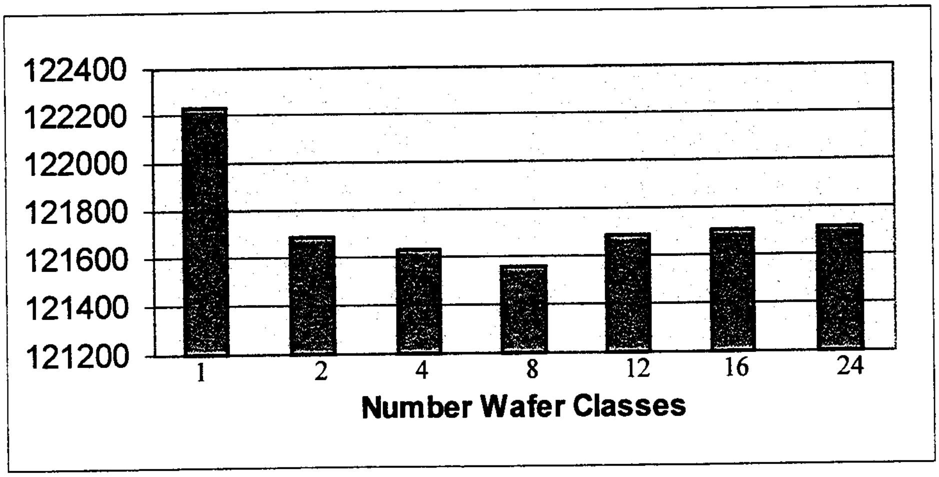


Figure 29: EM Tests: Number Packages

Figure 29 summarizes the relationship between the number of wafer classes in the models and the number of dice packaged from the test set. The number of packages is directly related to the number of functional tests that were performed, more testing leads to fewer packages. The explanation of this behavior is that during the extra testing, more bad dice were discovered, and this resulted in few dice being packaged. This relationship can also be seen in the graphs of true negatives and false positives in which more tests correspond to more true negatives and fewer false positives.

An interesting issue is the ability of the model to predict good and bad dice as measured in terms of true positives, true negatives, false positives, and false negatives. In general, the system is optimistic in that it tends to assume that dice are good rather than bad. With the default utility parameters, the system never misses a good die in its package decisions. Therefore the number of true positives is always equal to the number of good dice. Correspondingly, the number of false negatives is always zero. Thus the only measures that vary are the number of true negatives (i.e., the number of bad dice rejected) and the number of false positives (i.e., the number of bad dice packaged).

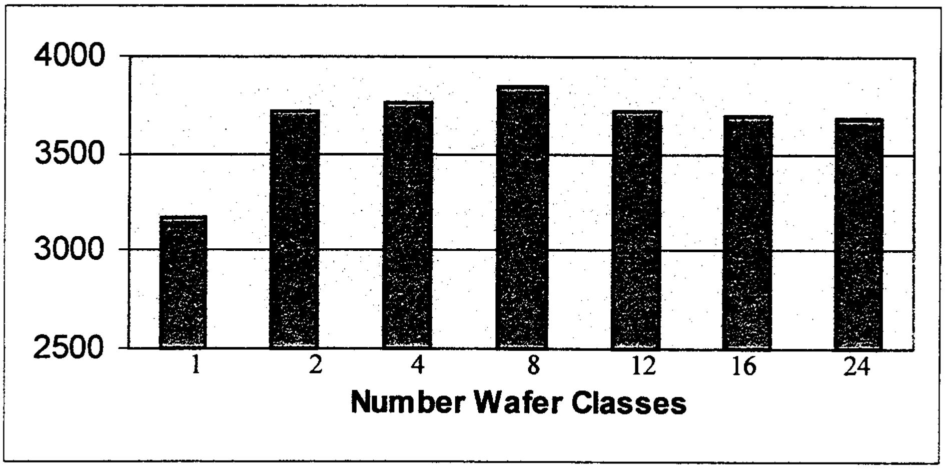


Figure 30: EM Tests: Number True Negatives

Figure 30 shows the relationship between the number of wafer classes and the number of true negatives from the test wafers. This is the number of bad dice that were rejected by the system.

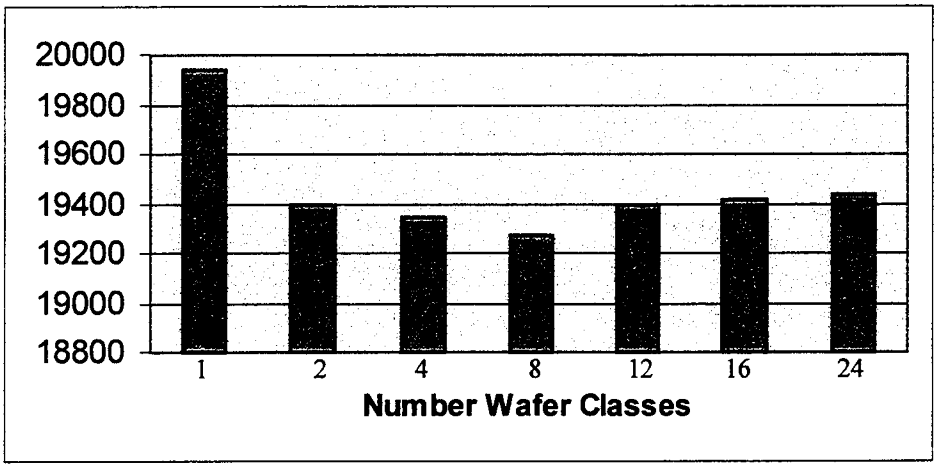


Figure 31: EM Tests: Number False Positives

Figure 31 shows the relationship between the number of wafer classes and the number of false positives. This shows that false positives are the complement to true negatives - as the number of true negatives increases, the number of false positives decreases. The strength of this relationship can be seen in the table of correlation coefficients presented in table 3.

	Test -log				False	True
	likelihood	Profit	Tests	Packages	Positives	Negatives
Test -log likelihood	1	-0.6768	-0.8996	0.8061	0.8061	-0.8061
Profit	-0.6768	1	0.9224	-0.9797	-0.9797	0.9797
Tests	-0.8996	0.9224	1	-0.9811	-0.9811	0.9811
Packages	0.8061	-0.9797	-0.9811	1	1	-1
False Positives	0.8061	-0.9797	-0.9811	1	1	1
True Negatives	-0.8061	0.9797	0.9811	-1	1	1

Table 3: EM Tests: Correlation Coefficients

To summarize the EM tests, likelihood and profit are related, but not directly correlated. The EM-trained model with four wafer classes had the lowest negative log likelihood score on the test data set, but the model with eight wafer classes produced the most profit. Testing and packaging are inversely related. In general the more the system tests, the less it packages. This reflects the fact that the system is initially optimistic about the quality of the wafer and tends to package all dice until it acquires some evidence that dice are not good. Further testing generally reveals some bad dice, which the system then rejects. Interestingly, this extra testing to detect bad dice and avoid false positives doesn't always pay off in terms of net profit.

5.1.2 The Best Gibbs Sampling Model

Tests similar to those performed on the EM-trained models were performed for models trained via Gibbs Sampling. For these tests the number of classes was fixed at four, and the effect of varying the ensemble size was investigated. Results from ensembles of size 1, 2, 4, 8, and 12 are reported. As with the EM tests, the models were trained on a data set consisting of 25 lots (600 wafers) and tested on a separate data set of 25 lots (600 wafers). Performance measures include training and testing set negative log likelihoods, the net profit, the number of tests, the number of packages, the number of false positives, and the number of true negatives. The results are summarized in table 4.

Number Models	Learning -log Likelihood	Testing -log Likelihood	Profit	Number Tests	Number Packages	Number False Positives	Number True Negatives
1	134.3082	136.687	1229476	7785	121726	19438	3674
2	133.9960	136.4806	1227836	11916	121078	18790	4322
4	133.8065	136.3636	1228155	11499	121075	18787	4325
8	133.7356	136.2936	1228073	11998	120945	18657	4455
12	133.7400	136.1679	1228640	11392	120895	18607	4505

Table 4: Gibbs Tests: Results

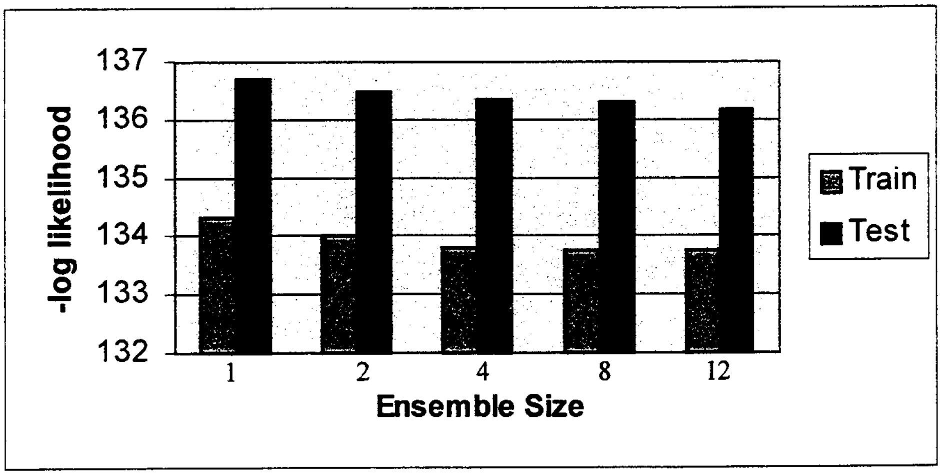


Figure 32: Gibbs Tests: Learning Curves

The learning curves for the Gibbs-trained models are shown in figure 32. Both the training and testing measures (negative log likelihood) decrease as the ensemble size increase, although differences are small. There is no evidence of overfitting.

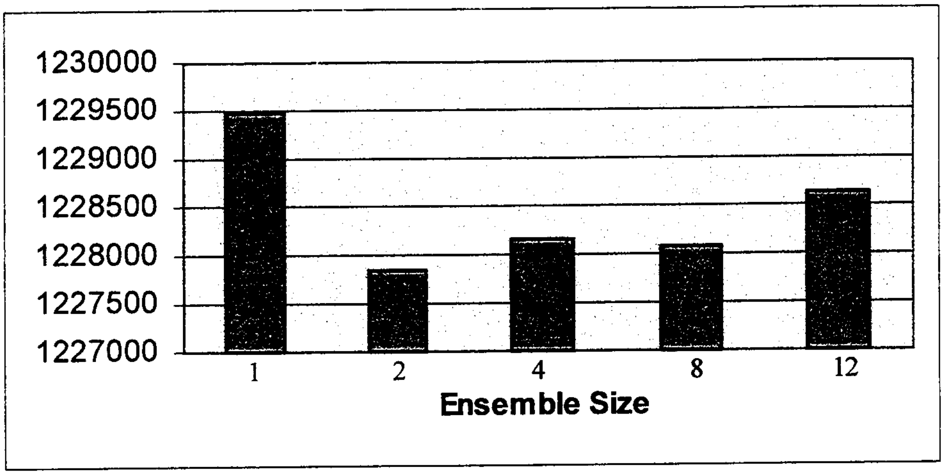


Figure 33: Gibbs Tests: Profit

The relationship between ensemble size and net profits is presented in figure 33. This figure shows that the maximum profit was realized with the ensemble of size one. The second highest profit was attained with the largest ensemble considered which included 12 models. This shows that for this task, ensemble test likelihood is not a good predictor of testing profit. In fact the correlation coefficient between negative log likelihood and profit is .4904 (see table 6 below).

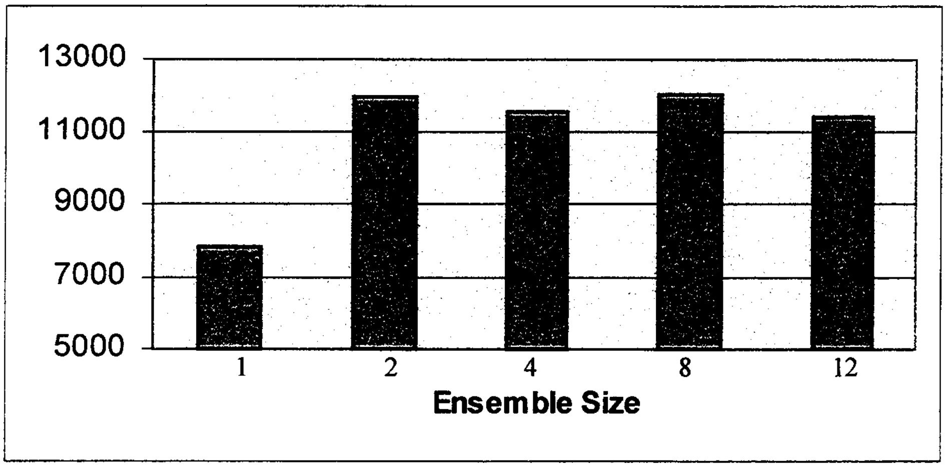


Figure 34: Gibbs Tests: Number Tests

The relationship between ensemble size and the number of functional tests is presented in figure 34. This shows that there is a distinct shift in performance when increasing the ensemble size beyond one. An ensemble of size one results in fewer tests than ensembles containing multiple models. For ensembles containing more than one model the number of tests appears to be unrelated to the number of models.

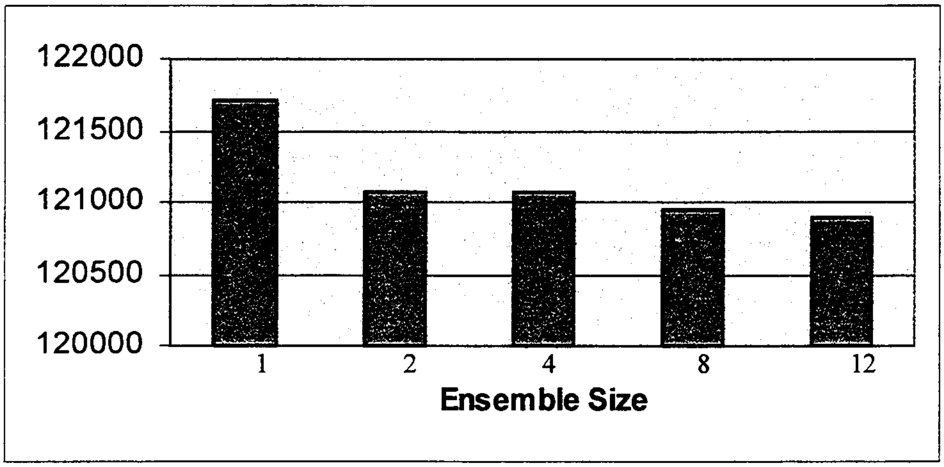


Figure 35: Gibbs Tests: Number Packages

The relationship between ensemble size and the number of packages is depicted in figure 35. In general, the larger ensembles produced fewer packages. As with the testing behavior, there appears to be a distinct shift in performance when moving to ensembles containing multiple models. A single model tests less and packages more than any of the larger ensembles.

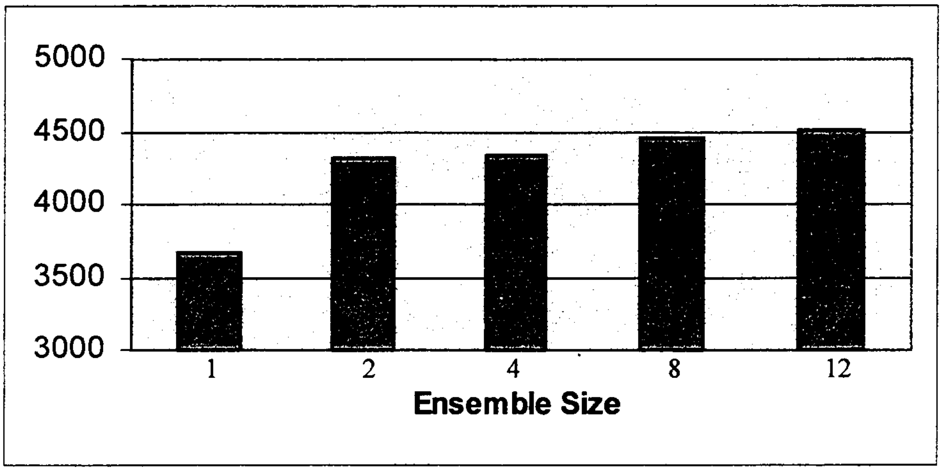


Figure 36: Gibbs Tests: Number True Negatives

Figure 36 presents the relationship between the ensemble size and the number of true negatives. From this graph, it is clear that the extra testing performed by the larger ensembles results in the discovery of more bad dice which are then rejected. By testing less and packaging more the single model rejects fewer bad dice (i.e., true negatives).

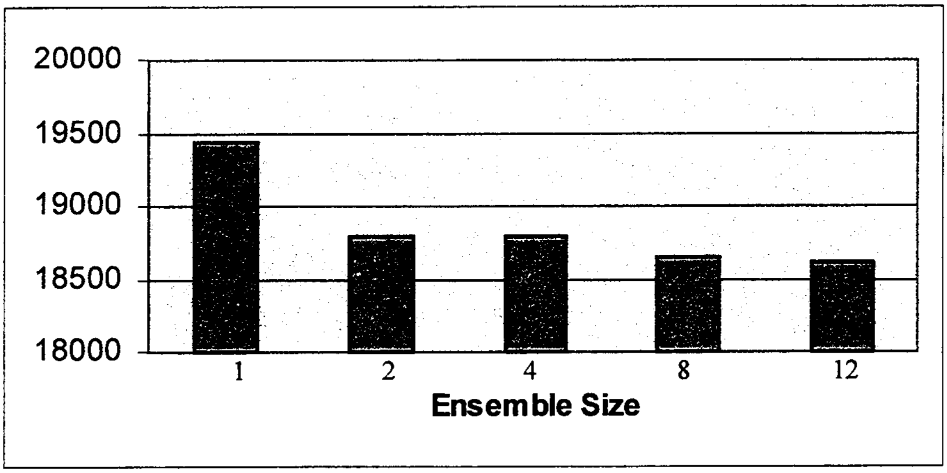


Figure 37: Gibbs Tests: Number False Positives

The relationship between ensemble size and the number of false positives (i.e., package bad die) is presented in figure 37. As with the EM model, the false positives are simply the complement to the true negatives. This graph also displays a distinct performance difference when the ensemble size is increased to more than one model. The single model tests less and packages more than the other models. The result is that the single model packages more bad dice (i.e., false positives). It is interesting that this testing behavior produces more profit than is realized by the more extensive testing performed by the larger ensembles. This demonstrates that there are times when the cost of extra information (i.e., additional tests) exceeds the value of the improved package decisions. This also calls into question the quality of the myopic VOI estimations for determining test stopping with ensembles of stochastic models.

The correlation coefficients for the various performance measures are presented in table 5.

	Test -log likelihood	Profit	Tests	Packages	False Positives	True Negatives
Test –log likelihood	1	0.4904	-0.7628	0.9203	0.9203	-0.9203
Profit	0.4904	1	-0.9364	0.7859	0.7859	-0.7859
Tests	-0.7628	-0.9364	1	-0.9529	-0.9529	0.9529
Packages	0.9203	0.7859	-0.9529	1	1	-1
False Positives	0.9203	0.7859	-0.9529	1	1	-1
True Negatives	-0.9203	-0.7859	0.9529	-1	-1	1

Table 5: Gibbs Tests: Correlation Coefficients

To summarize the Gibbs tests, the most salient result is that a single Gibbs model outperforms the larger ensembles, even though the larger ensembles fit the test data better (as measured by negative log likelihood). The larger ensembles tend to test more than the single model and they package less. They discover more bad dice and avoid more false positives, but the extra cost of testing outweighs the gain in profit. These results demonstrate that the relation between likelihood and performance is more complicated when considering ensembles.

5.1.3 Model Comparisons

In the previous two sections, model performance was compared between models trained according to the same learning algorithms. EM-trained models were compared to other EM-trained models, Gibbs-trained models were compared to other Gibbs-trained models. In these section, the best of the selective test models are compared to one another and to several non-VOI approaches. In all, five approaches are compared:

1. Exhaustive test, package all dice that pass functional test. This approach corresponds to the current manufacturing practice.
2. Test none; package all dice. Under this approach die-level functional test is eliminated. All dice are packaged then package tested.

3. Selective test with Gibbs-trained model. For these tests a single Gibbs-trained model was employed.
4. Selective test with EM-trained model.
5. Test none; package all good only. This approach assumes an oracle that can determine which dice are good without testing. Since oracles for this task do not exist, the purpose of this approach is to provide a benchmark for optimal performance.

All approaches are evaluated on the identical task of testing 25 wafer lots (600 wafers). Each wafer contains 209 dice. Summary statistics are provided in table 6.

Total Number Good Dice	102288
Total Number Dice	125400
Yield	0.8157

Table 6: Test Wafers Statistics

The results from the model tests are presented in table 7.

	Total Profit	Number Tested	Number Packaged
Exhaustive Test	1184550	125400	102288
No test, Package All	1226598	0	125400
Gibbs	1229414	7313	121911
EM	1229531	8210	121560
Oracle	1278600	0	102288

Table 7: Model Comparisons: Results

The profits that were realized under the various testing policies are summarized in figure 38. The first result to consider is that the current approach of exhaustive test produces the least profit. The oracle produces the most profit, of course. The no-test approach produces slightly less profit than the selective test approach using either the Gibbs or the EM.

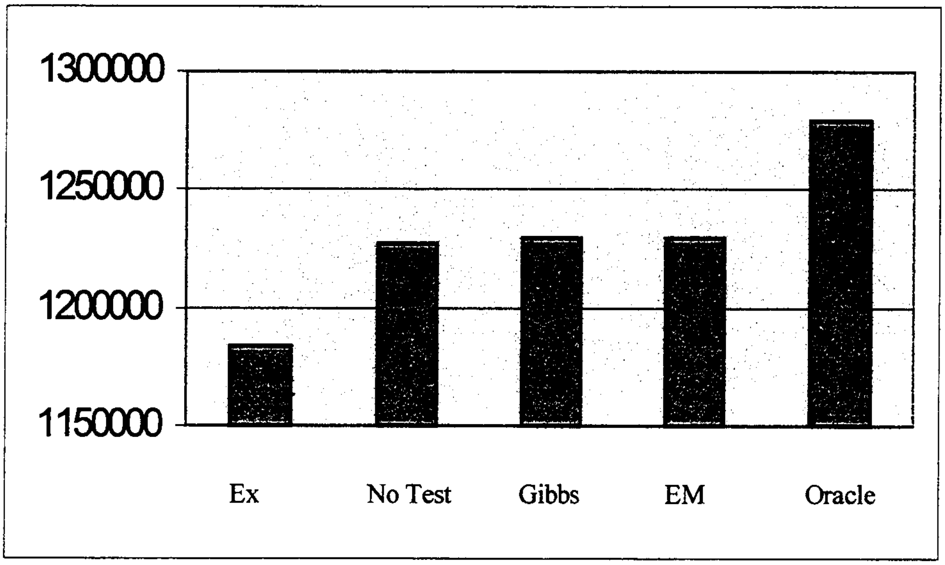


Figure 38: Model Comparisons: Profit

The number of tests performed by the various approaches is not surprising. The exhaustive test approach tests all 125400 dice. The two test-none approaches (no test and oracle) test zero dice. The two selective test approaches (Gibbs-trained and EM-trained) test a fraction of the dice, 5.85% for the Gibbs-trained, 6.65% for the EM-trained.

The number of dice packaged according to the various approaches is shown in figure 39. The exhaustive-test approach and the oracle-based approach both perform perfect packaging. These approaches package all 102288 good dice or 81.57% of the total. The no-test approach packages all 125400 dice. The two selective test approaches package almost all of the dice. The Gibbs-trained model packaged 121911 dice, or 97.21% of the total. The EM-trained model packaged 121560 dice, or 96.94% of the total.

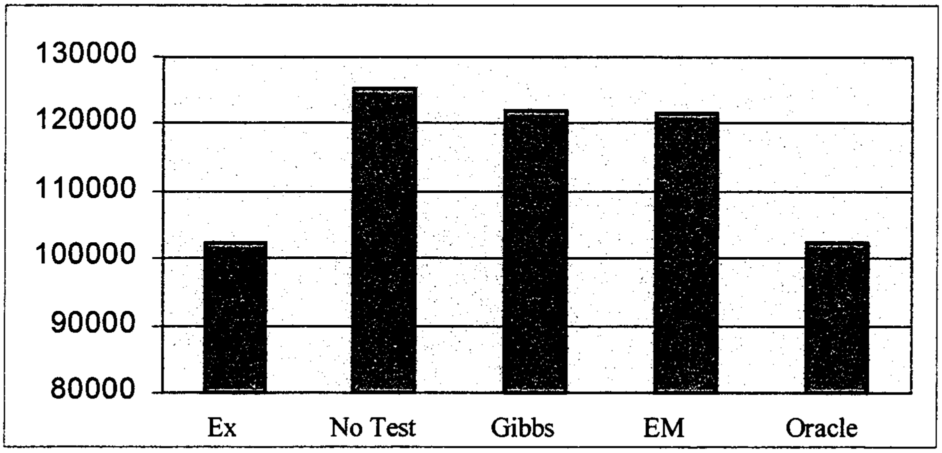


Figure 39: Model Comparisons: Number Packages

To summarize the model comparisons, with the exception of the Oracle model, the EM-trained model produced the most profit over the testing scenario. The Gibbs-trained model produced slightly less. The No-Test policy produced slightly less profit than the Gibbs model. All approaches produced more profit than the current exhaustive test approach. Comparing the EM-trained model and the Gibbs-trained model the EM-trained model tested slightly more (8210 vs. 7313) and packaged slightly less (121560 vs. 121911), but in general the performance of these two approaches is very similar. Both the EM-trained model and the Gibbs-trained model produced over 96% of the profit realized by the Oracle model. The No-Test policy also produced profits in this range, so a reasonable question is what advantage the selective test approaches have over the No-Test policy. The answer to this question is that on wafers with high yield there may be little benefit, and in fact the optimal policy for high-yield wafers is probably to package all dice without performing any die-level functional tests. The problem with this approach is that process problems will not be detected until package test results become available. This is an especially serious concern, since often the packaging process occurs at a location far from the wafer fabrication plant. For example, it is common practice to send wafers that are fabricated in the United States to Asia for packaging. In these cases the delay in feedback and the costs of shipping and handling make the Test-None policy risky. One of the benefits of the selective test approach is that for good wafers it can produce profits comparable to those produced with a Test-None policy, but it can also detect abnormal wafer and process problems while the wafers are still at the wafer fabrication plant. This ability to adjust testing behavior to wafer characteristics is explored in more detail in section 5.4.

Finally, another benefit of the selective test approach over the exhaustive test approach is that since the system tests only a fraction of the dice, the resources that are allocated to the exhaustive testing policy can be reallocated. This means that fewer testers are required as well as the necessary support for these machines. Alternatively, in cases where functional testing is the process bottleneck, a decrease in number of dice tested per wafer means that more wafers can be tested, thus wafer starts, throughput, and profits per unit time can be increased. Thus there are two deficiencies in the profit model utilized in this study (and in typical industry planning): (1) the model doesn't account for the effects of testing load on wafer starts and throughput, and (2) the model doesn't place any value on the diagnostic value of test results. If these factors are considered, then the selective test approach is even more attractive.

5.2 Evaluation of the Myopic VOI Stopping Rule

The selective test approach relies on myopic value of information (VOI) computation to decide when to terminate testing. In order to determine how well this heuristic performs in testing, experiments were performed in which myopic VOI stopping was compared to the optimal stopping point. To determine the optimal stopping point, the selective test policy was modified to continue testing past the past the point of non-positive VOI until all dice were tested. Thus the optimal stopping point is determined by looking back over the history of testing decisions to the point at which profit would have been maximized had the system stopped at that point. Profit includes the costs for functional tests up to that point and the rewards obtained by making package decisions at that point. Under this new policy only the stopping decision was modified, the best test decision and the inking (packaging) decisions were the same.

For these tests a single stochastic model with four wafer classes was trained via the EM algorithm on 600 wafers. The learned model was embedded within the selective test structure and the testing performance was measured on a distinct set of 600 wafers. The results from these tests are summarized in table 8. The results from the exhaustive test policy are provided for comparison.

	Profit	Number Tests	Number Packages
Exhaustive Test	1184550	125400	102288
VOI Selective Test	1229531	8210	121560
Optimal Stopping	1231970	11206	119264

Table 8: Optimal Stopping: Results

The first result to consider is the net profit realized by the various approaches. This information is presented in figure 40.

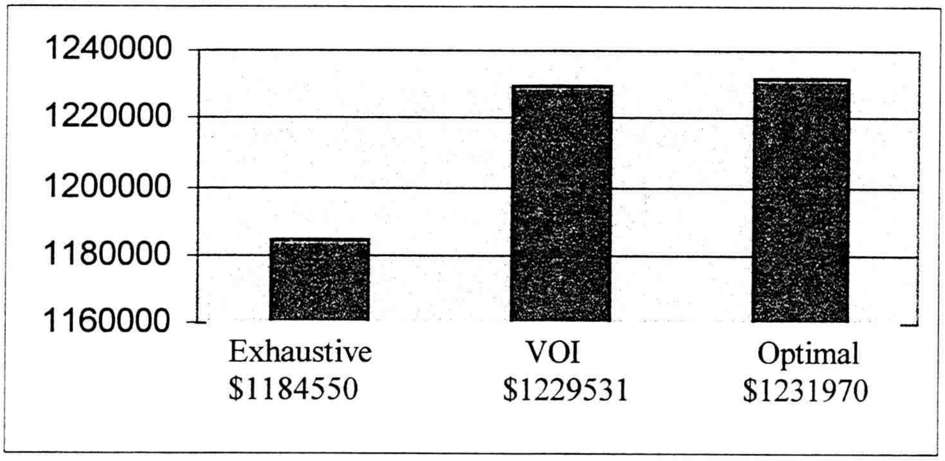


Figure 40: Optimal Stopping Tests: Profits

Notice that the Optimal Stopping policy produces only slightly more profit than the myopic VOI approach. In fact, the myopic VOI approach realizes over 99% of the profit achieved by Optimal Stopping

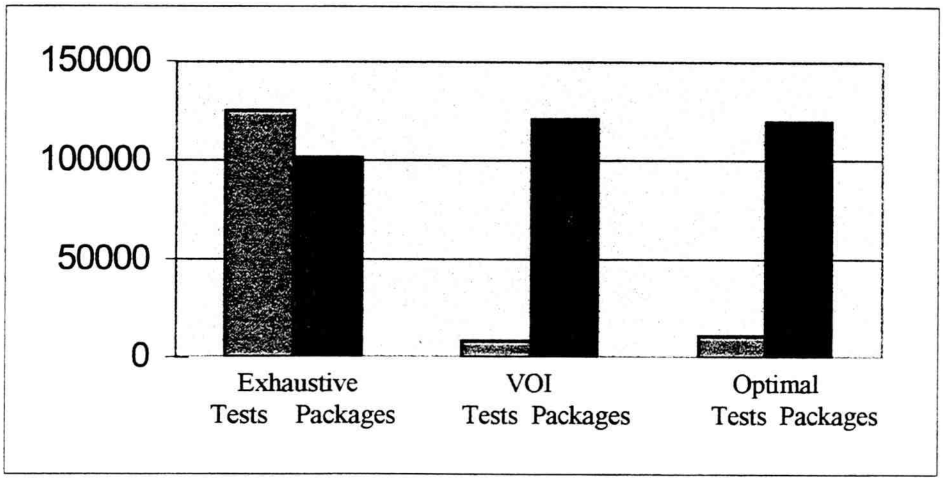


Figure 41: Optimal Stopping Tests: Number Tests and Packages

The next result to consider is the number of tests and the number of packages from each of the policies. These results are presented in figure 41. The Optimal Stopping approach performs more functional tests (11206 vs. 8210) and produces slightly fewer packages (119264 vs. 121560) than the myopic VOI approach.

To summarize the optimal stopping experiments, myopic VOI Stopping tests less than Optimal Stopping (performing only about 73% as many tests) and packages more (about 2% more packages). These extra packages are false positives. So the Optimal Stopping approach spends a bit more on functional testing in order to reduce the number of bad dice packaged. But these tests are not uniformly distributed across the wafers. Even though the overall number of tests performed by the myopic VOI Stopping approach is less than the number of tests performed by the Optimal Stopping approach, the number of wafers on which Optimal Stopping performed more testing than myopic VOI is only 262 out of 600 (44%). So on the majority of wafers, myopic VOI Stopping tested more than Optimal Stopping. Thus simply testing beyond the point of non-positive VOI will not improve performance. Optimal stopping is better at recognizing good wafers early, and in deciding how to test bad wafers to a degree needed to make good package decisions. Despite the differences, myopic VOI Stopping performs very well and realizes over 99% of the profit achieved by Optimal Stopping.

5.3 Training Policies

Up to this point the training set has been fixed at 50 lots (600 wafers). While this is not an unreasonable number from a process perspective, it is interesting to ask whether the same performance can be obtained with a smaller training set. It is also of interest to see how performance is affected by training sets that are much smaller than 50 lots.

To investigate this behavior a single stochastic model with four wafer classes was trained with the EM algorithm on training sets of various sizes, from 10 to 500 wafers. The resulting models were then embedded in the selective test structure and performance was measured on simulated testing of 600 wafers. Each experiment was performed twice with distinct training sets to increase the generality of the results. This provided two measures for each sample size, and these results were averaged together to yield a single measure. Table 9 presents the learning results for both the training and the testing sets in terms of negative log likelihood.

Size	Training	Testing
10	96.89	151.02
20	106.88	147.29
40	114.53	144.51
60	123.18	142.69
80	128.16	141.10
100	128.76	140.42
200	125.67	139.63
300	132.29	137.78
400	135.12	136.58
500	133.14	136.46

Table 9: Training Policies: Learning Results in –log likelihoods

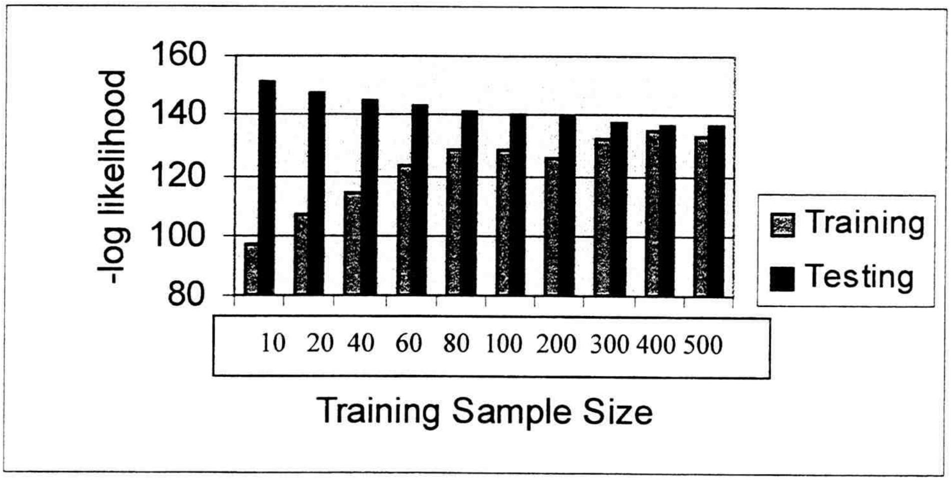


Figure 42: Training Policies: Learning Curves

The relationship between training sample size and negative log likelihood for the training and testing sets is presented in figure 42. The results show that there is considerable overfitting for the small sample sizes. In these cases the training set measure is very low, but the test set measure is high. As the training sample size is increased, the test negative log likelihood decreases and the training negative log likelihood increases, indicating a better fit of the testing data and a worse fit on the training data. With a training set size of about 300, the likelihood measures converge to where the ratio of training set likelihood to test set likelihood is greater than .95. The profits that are realized by testing 600 wafers with the learned models are presented in table 10.

Size	Profit
10	1221635
20	1225978
40	1226158
60	1226536
80	1226607
100	1226856
200	1226994
300	1227141
400	1227449
500	1227412

Table 10: Training Policies: Profits Results

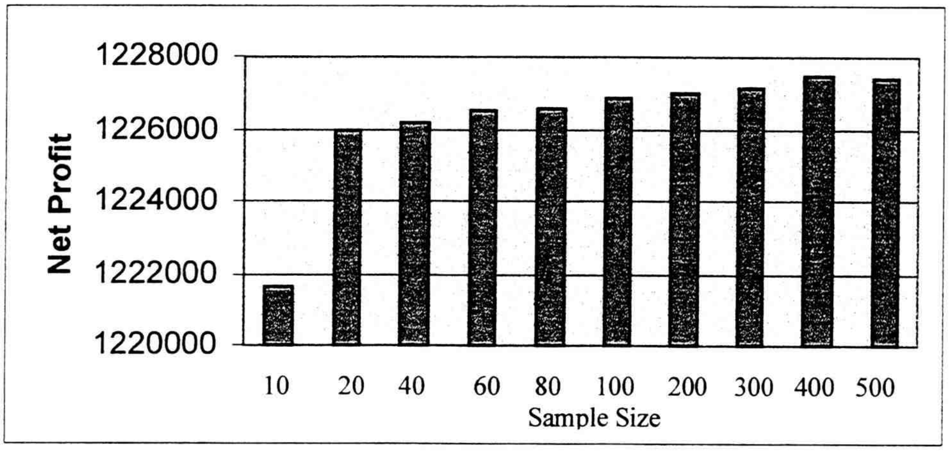


Figure 43: Training Policies: Profits

Figure 43 depicts the relationship between training sample size and net profits. This shows a direct relationship between the amount of training data and the profit: more data means more profit. This relationship can also be observed in figure 44, which plots profits vs. test set negative log likelihood.

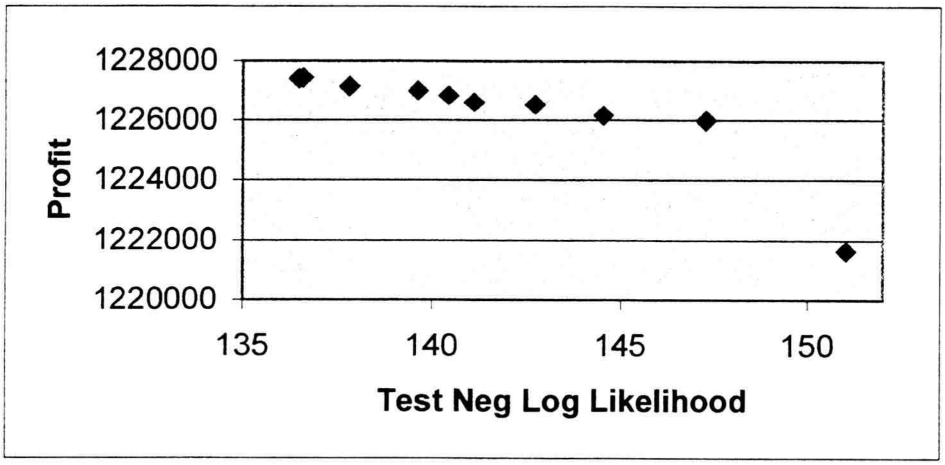


Figure 44: Training Policies: Profit vs. Test Negative Log Likelihood

To summarize the results of varying the training sample size, profits can be realized with a small training set and additional training examples increase profits. It is interesting to note that once the training set is larger than 80 wafers then the profit from the selective test policy exceeds that achieved by the test-none policy. Since training with the EM algorithm is relatively fast, these results suggest that for this task, a training sample of 300 wafers is reasonable.

5.4 Detection of Process Problems

An interesting question for any testing policy is how it responds to abnormal wafers. Although the selective test approach was targeted towards a stable mature product, process problems are not uncommon and abnormal wafer test results are often the first symptoms of such problems. In the presence of process problems, it is questionable whether the correct test behavior is to continue selective testing. Within the context of mature product testing, perhaps the most reasonable response to an abnormal wafer is to exhaustively test the wafer and to raise an alert to process operators. If this is the case, then the important question for the selective test approach is not how much profit it can make, but how well it can recognize abnormal wafers.

To explore this behavior, six wafers are considered in detail. The first two wafers are typical 'good' wafers with yields over 80%. The next two wafers exhibit extremely low yield, less than 10% each. These wafers are considered 'bad' wafers. The final two wafers exhibit yields in between at 66% and 65%. These wafers are considered 'mediocre' wafers. The last wafer exhibits significant spatial clustering as measured by the T-statistic (described in chapter 2). A single stochastic model with four wafer classes was trained with the EM algorithm on the test results from 600 wafers. This model was then embedded in the selective test structure and testing of the six wafers described above was performed. The results from these tests are presented in table 11.

Wafer	#Goods	Yield	# Tests	# Packages	Current Profit	VOI Profit
1	178	0.85	9	205	2068.25	2158.25
2	180	0.86	9	206	2093.25	2185.50
3	17	0.08	202	25	55.75	43.75
4	2	0.01	202	10	-131.75	-143.75
5	138	0.66	57	182	1568.25	1583.25
6	118	0.56	13	202	1318.25	1277.00

Table 11: Detecting Process Problems: Wafer Summary

In table 11, current profit is the profit realized under the exhaustive test policy. VOI profit is the profit realized under the VOI selective policy.

To better document the testing behavior, wafer test maps are presented for each wafer. These maps show the actual good dice, the dice that were tested according to the selective test policy, and the dice that were packaged according to this policy. In each map green (light) encodes true, so green represents good dice, tested dice, and packaged dice. Red (dark) encodes bad dice, untested dice, and unpackaged dice.

Good Wafers: yield > .85

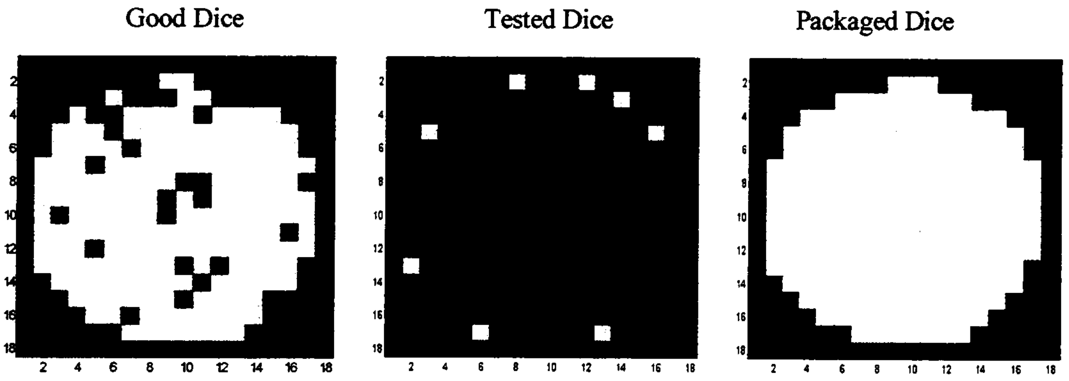

Good Dice Tested Dice Packaged Dice

Figure 45: Detecting Process Problems: Wafer 1

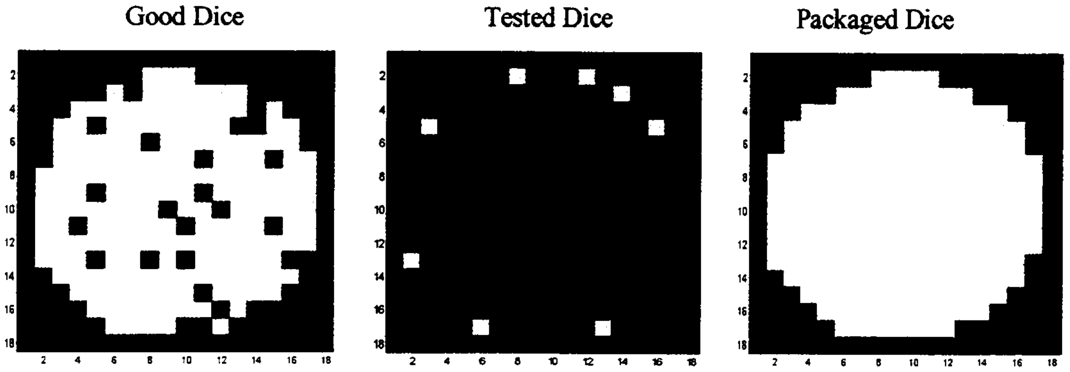
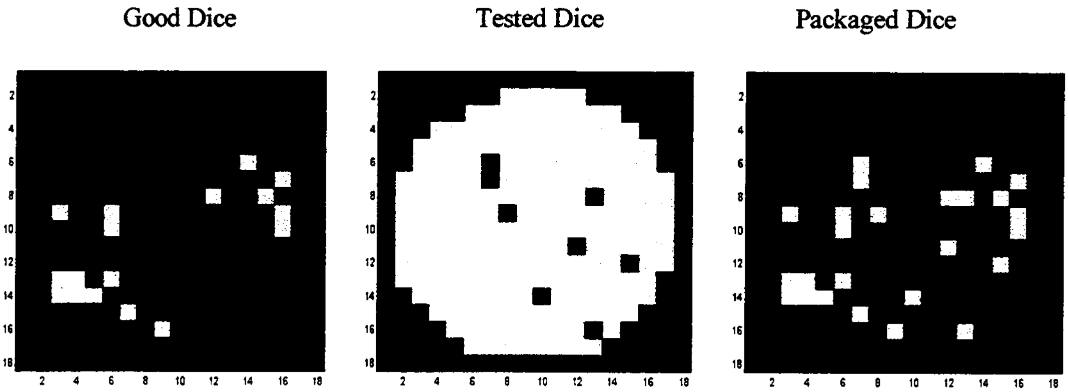

Good Dice Tested Dice Packaged Dice

Figure 46: Detecting Process Problems: Wafer 2

Summary of good wafer tests: For the typical 'good' wafer, the system performs minimal testing. In these cases it tested only around the outside edges where bad dice are most likely. The system then decides to package almost all of the dice on the wafers. This testing and packaging behavior produces a net profit over the exhaustive test policy and the test-none policy.

Bad Wafers: yield < .10

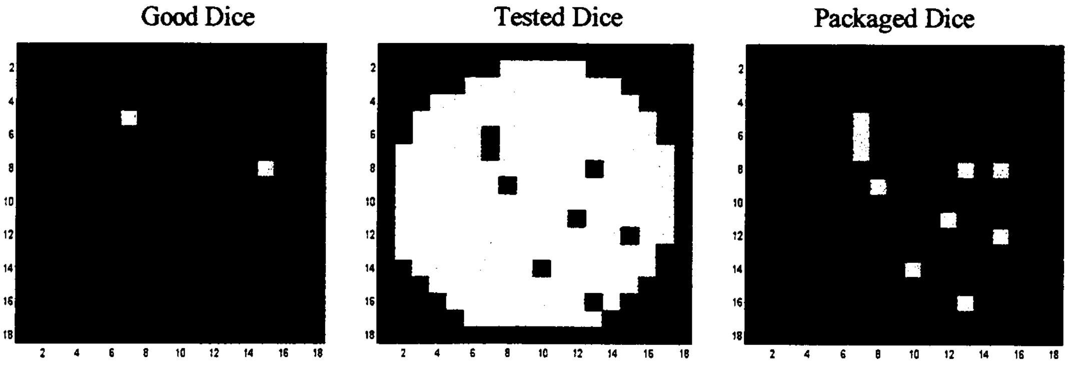


Good Dice

Tested Dice

Packaged Dice

Figure 45: Detecting Process Problems: Wafer 3



Good Dice

Tested Dice

Packaged Dice

Figure 46: Detecting Process Problems: Wafer 4

Summary of bad wafer tests: For bad wafers, the system typically tests most dice on the wafers. It then packages only a few dice. If actually implemented in a production environment, these wafers would most likely be held for analysis rather than sent to package. The important point is that the system was able to recognize the bad wafers and then decided to test to confirm that they were bad. The testing and packaging decisions for bad wafers are distinctly different than those performed on good wafers.

Mediocre Wafers: .50 < yield < .70

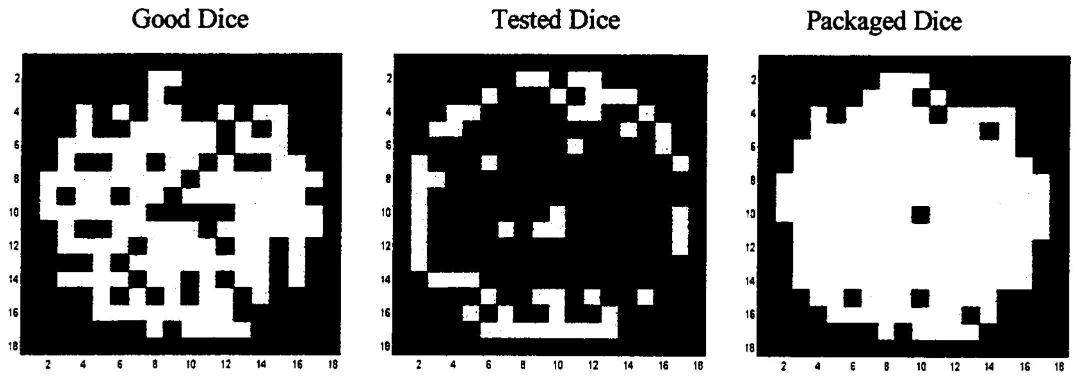


Figure 47: Detecting Process Problems: Wafer 5

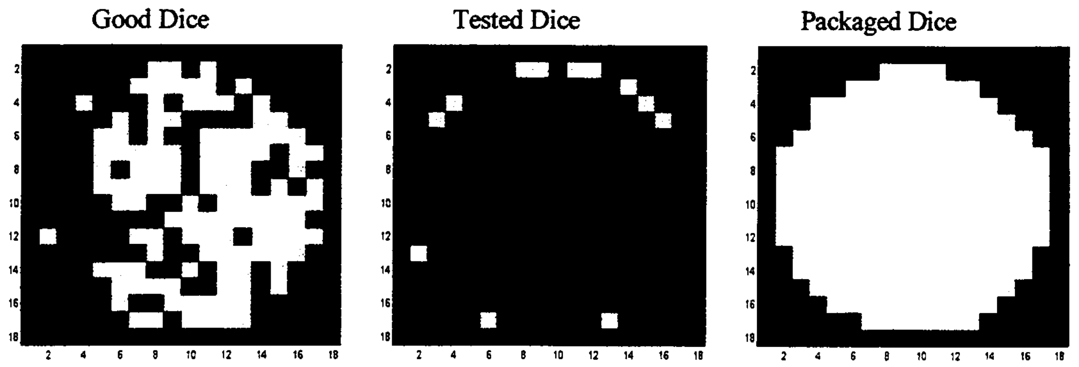


Figure 48: Detecting Process Problems: Wafer 6

Summary of mediocre wafer tests: the system performs more testing than on the good wafers and less than on the bad wafers. So, the degree of testing is sensitive to the yield. For wafer 5 the system did a reasonable job with the test and package decisions and produced a net profit over the exhaustive test policy. On wafer 6, the system was overly optimistic in its assessment. This is most likely due to the fact that the failures on wafer 6 exhibited an unusual pattern of spatial clustering. Note that wafer 6 was determined to have significant spatial clustering according to the T-statistic, and the few wafers in the training set exhibited this behavior (see T-statistic summary in chapter 2).

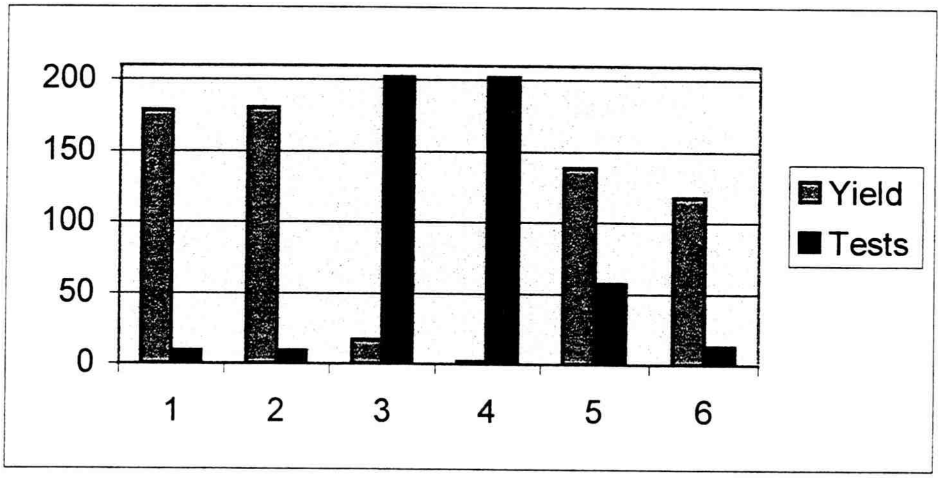


Figure 51: Detecting Process Problems: Yields and Tests

Figure 51 depicts the yield and test relationship on the six wafers. Note that yield in this figure refers to the count of true good dice on the wafers. For wafers with high yield values, the system performs few tests. For wafers with low yield values, the system performs extensive testing. For wafers with yields between the high and low values, the system performs a number of tests between that which was performed for the high-yield and low-yield wafers.

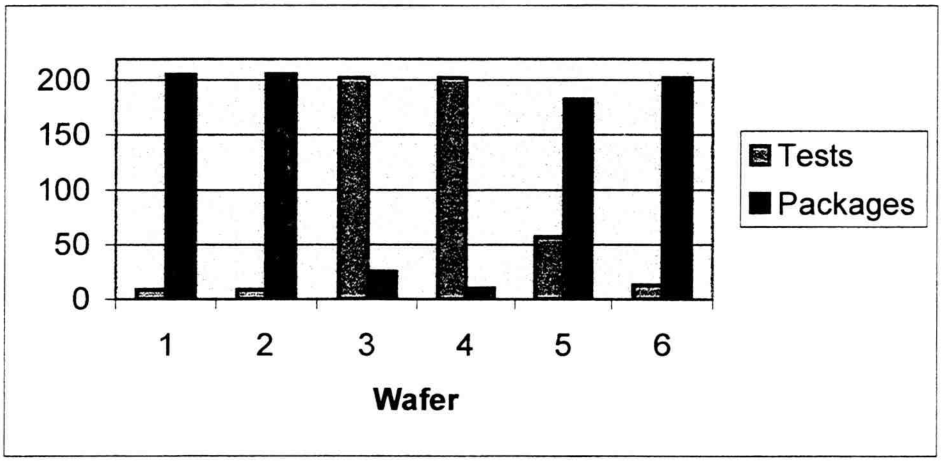


Figure 52: Detecting Process Problems: Number Tests and Packages

Figure 52 depicts the relationship between test and package decisions. In general the number of tests performed and the number of dice packaged are inversely related: the system tests more on wafers with lower yield and subsequently packages fewer dice. This relationship between wafer yield and test and package decisions holds for the larger wafer test set. Figure 53 is a scatterplot of yield (number of good dice) vs. the number of tests performed by system on the test set of 600 wafers.

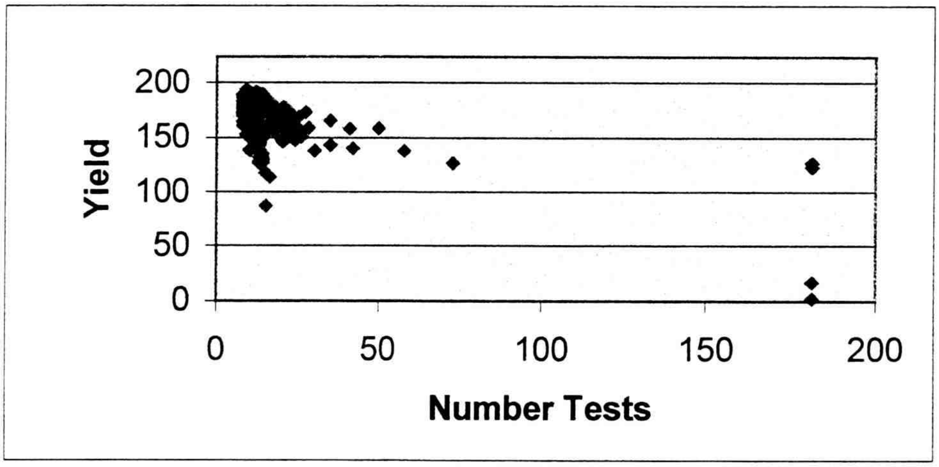


Figure 53: Detecting Process Problems: Yield vs. Number Tests

This behavior in which only bad wafers are tested extensively means that for mature and stable products only a relatively few wafers are tested extensively. Figure 54 presents a histogram of the number of wafers and their level of testing. From this graph it can be seen that the majority of wafers (353) received between 11 and 20 tests, and only 33 wafers received more than 21 tests.

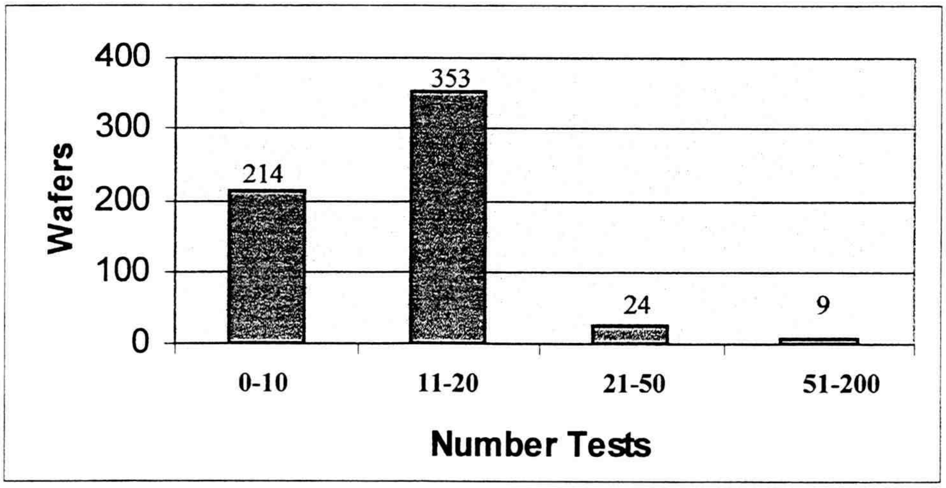


Figure 54: Detecting Process Problems: Number Tests Histogram

To summarize the tests to determine the response to process problems, the selective test approach is responsive to abnormal wafers. In general, there is a direct relationship between the wafer yield and the number of functional test that are performed. The higher the yield, the fewer tests. This means that for a stable and mature product the system will test only a small fraction of the total dice. However, the system is sensitive to abnormal yields, which are indicative of process problems. On such wafers, the system tends to test more thoroughly. For wafers with extremely low yield, the system responds by testing almost all dice. This means that a minimal amount of resources are expended on good wafers, yet bad wafers are detected. This also suggests that the output from the selective test approach could be employed in the SPC methods that routinely monitor for process problems. A straightforward extension to the current SPC system would be to replace actual test measures with predicted measures. So, for example, rather than setting control limits around the actual functional test results, the control limits could be set around the predicted, i.e., predicted by the model, functional test results. Thus, the selective test approach provides dual benefits. First, it greatly reduces the requirement for testing resources. Second, it satisfies the requirement for prompt detection of process problems.

5.5 Robustness to Changes in Utility Parameters

One benefit of constructing a test and package policy based on expected utility is that changes in utility parameters should result in rational changes in performance without explicit re-engineering the learned models or control structures. The learned models capture the expected distribution of good and bad dice, and the control structures combine this information with utility parameter values to make test and package decisions. Changes in utility parameters result in policy and performance changes through their effects on expected utility. In theory, as utility parameters change, these values are simply passed to the system which automatically adjusts its test and package policy to maximize overall expected utility. To explore this behavior in the wafer test problem, changes to two of the utility parameters were considered:

1. Changes to the cost of performing a single functional test
2. Changes to the cost of packaging a single die

For each of these parameters, a series of tests were performed in which the parameter of interest was swept through a range of values and performance on a testing scenario was measured. For these tests, a single stochastic model with four wafer classes was trained via EM on 600 wafers. The learned model was then combined with the utility model and simulated testing was performed on 48 wafers. Performance was measured in terms of the number of functional tests performed, the number of dice packaged, the number of false positives, and the number of true negatives.

5.5.1 Changes to Package Cost

In the first set of tests, the cost to package a single die was manipulated. Let c_p represent the normal package cost. Then consider the effects of cutting the package cost in half (.5 c_p) and of doubling the package cost (2 c_p). The results are summarized in table 12.

Package Cost	Number Tests	Number Packages	Number False Positives	Number True Negatives
$.5c_p$	240	9854	1568	178
c_p	796	9672	1386	360
$2c_p$	2484	9312	1026	720

Table 12: Robustness Tests: Changes to Package Cost: Results

The first result to note is that as the package cost is increased the system increases the number of functional tests that are performed. This relationship can be observed in figure 55.

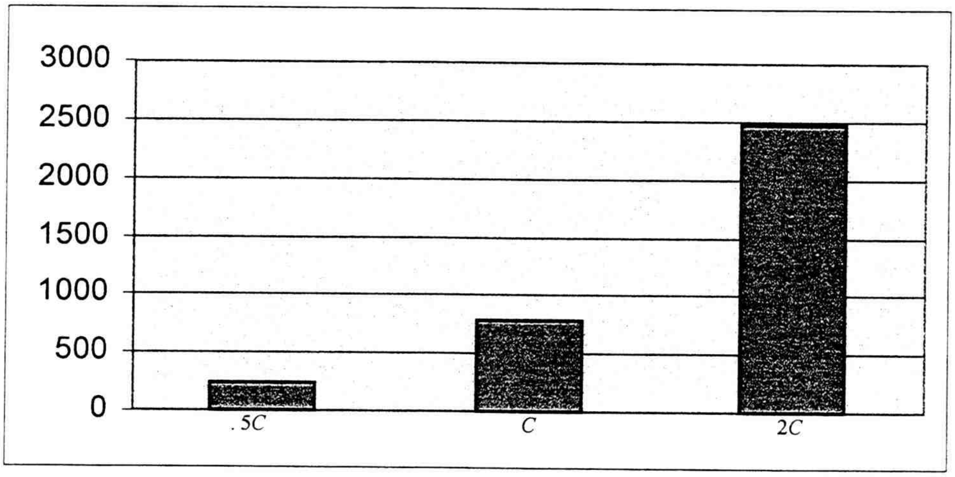


Figure 55: Robustness Tests: Changes to Package Cost: Number Tests

As the number of tests increases, the number of packages decreases. The explanation for this behavior is that the increased testing discovered additional bad dice that were then rejected rather than packaged. This can be seen in figure 56, which depicts the relationship between package cost and the number of packages, and in figure 57 which depicts the relationship between package cost and the number of true negatives.

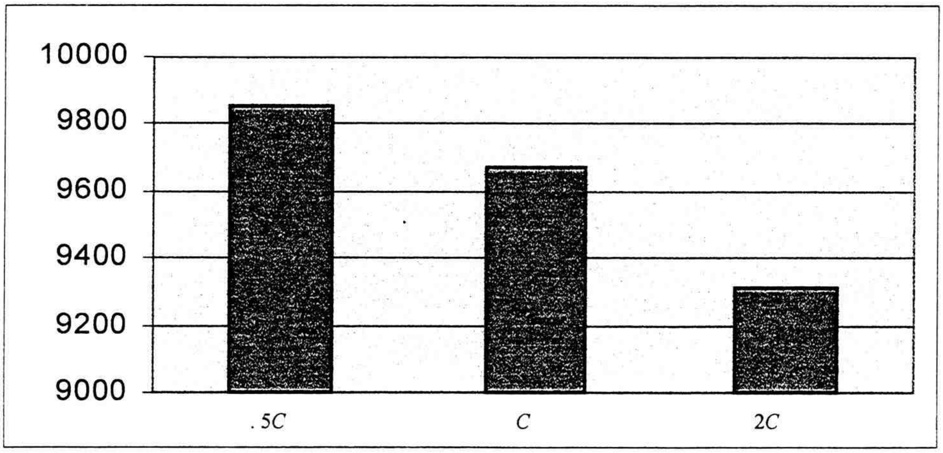


Figure 56: Robustness Tests: Changes to Package Cost: Number Packages

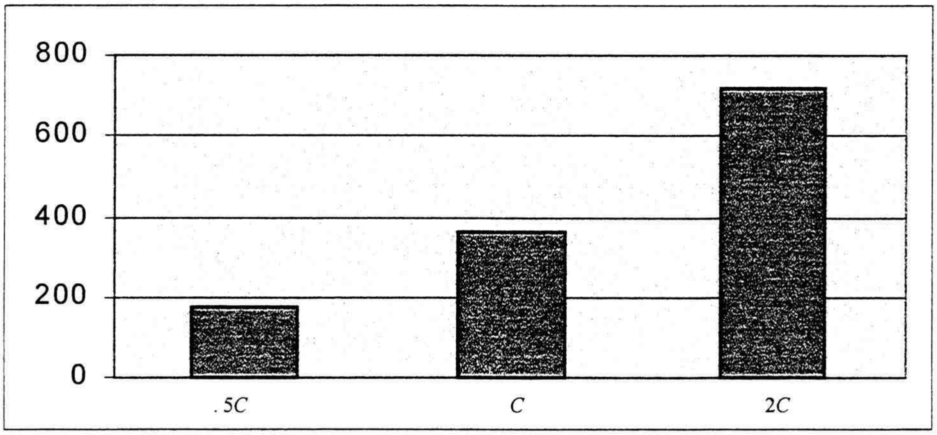


Figure 57: Robustness Tests: Changes to Package Cost: Number True Negatives

The explanation that increasing package cost results in an increase in the number of tests and a decrease in the number of packages due to the discovery of more bad dice is further confirmed by the measure of false positives (i.e., the number of bad dice packaged). The relationship between package cost and the number of false positives is presented in figure 58.

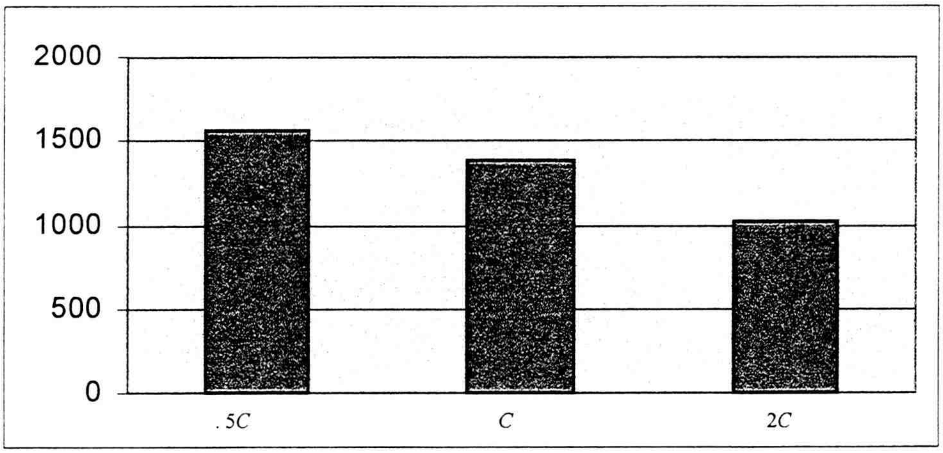


Figure 58: Robustness Tests: Changes to Package Cost: Number False Positives

To summarize the effects of varying the package cost, when it is relatively inexpensive to package dice then the system packages more and tests less. As the package cost increases, false positives become more expensive, so more tests are performed to reduce this risk. Thus, with respect to changes in package cost, the system performs rationally by adjusting its testing and package decisions to maximize expected profits.

5.5.2 Changes to Functional Test Cost

A second set of tests was performed in which the functional test cost was manipulated. Let c_f represent the current cost of a single functional test. Then consider the effects of setting the functional test cost at $.1\,c_f$, $.67\,c_f$, c_f, $1.33\,c_f$, $.167\,c_f$, $2\,c_f$, and $10\,c_f$. The results are summarized in table 13.

Test Cost	Number Tests	Number Packages	Number False Positives	Number True Negatives
.1*C*	10032	8286	0	1746
.67*C*	1509	9485	1199	547
C	796	9672	1386	360
1.33*C*	321	9835	1549	197
1.67*C*	240	9854	1568	178
2.0*C*	169	9912	1626	120
10*C*	0	10032	1746	0

Table 13: Robustness Tests: Changes to Functional Test Cost: Results

The first result to consider is that the number of functional tests is directly related to the functional test cost: the cheaper the single test cost, the more tests are performed. At $.1\,c_f$ tests are so inexpensive that the system performs exhaustive testing. At $10c_f$ tests are so expensive that no functional tests are performed. Figure 59 summarizes this relationship between functional test cost, the number of tests performed, and the number of dice packaged.

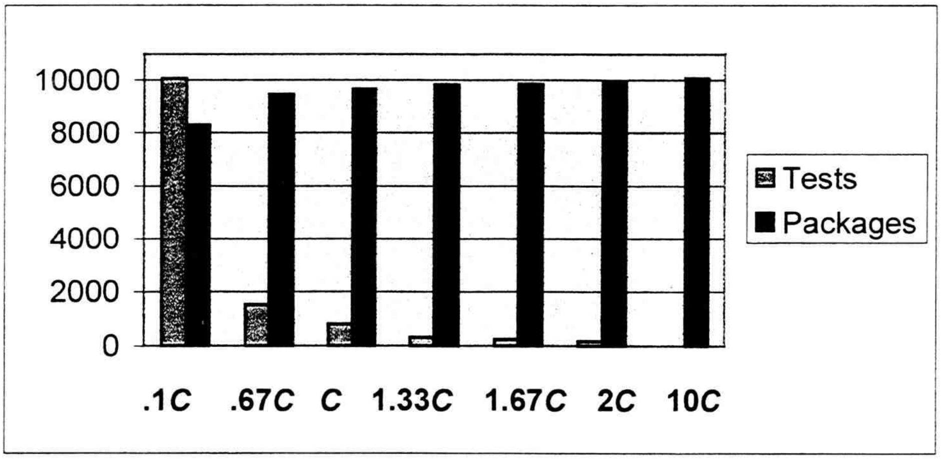


Figure 59: Robustness Tests: Changes to Functional Test Cost: Tests and Packages

The next result to consider is that the number of true negatives (bad dice that are rejected) decreases as the functional test cost is increased. This reflects the fact that with a higher test cost, fewer dice are tested and thus fewer bad dice are discovered. This demonstrates the overall optimistic bias that the system has toward dice; in the absence of evidence the system is more likely to package a die than to reject it. The relationship between functional test cost and the number of true negatives is summarized in figure 60.

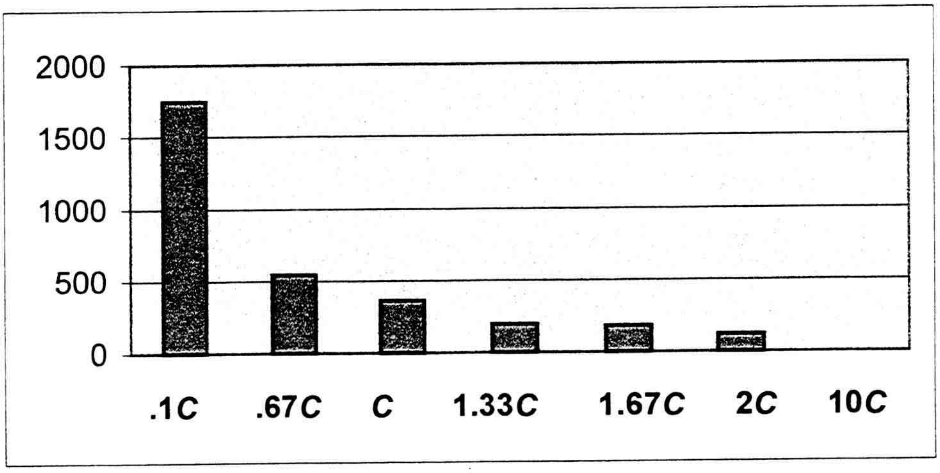


Figure 60: Changes to Functional Test Cost: Number True Negatives

A final result to consider from these tests is the relationship between functional test cost and the number of false positives (package bad die). An increase in the functional test cost results in an increase in the number of false positives. This is just the complement to the true negative results. As single test cost is increased, fewer tests are performed, and thus the system packages more dice, including some that turn out to be bad. At .1 c_f all dice are tested, and thus there are no false positives. The relationship between the functional test cost and the number of false positives is shown in figure 61.

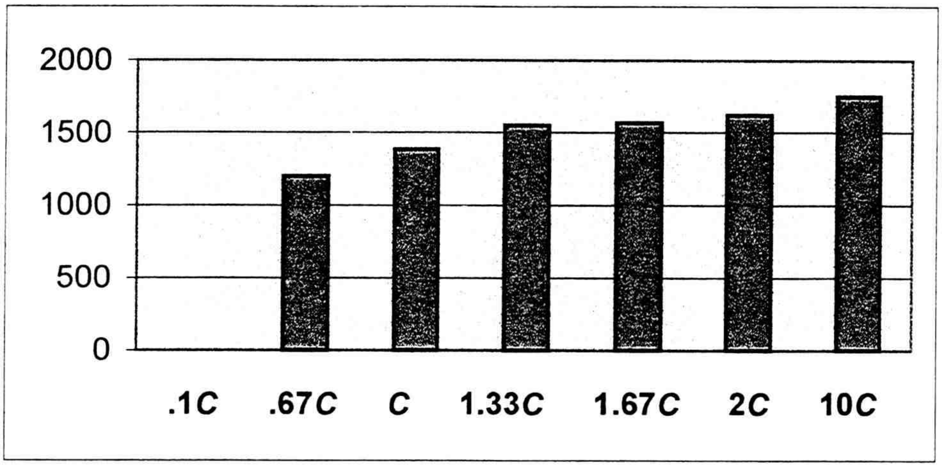


Figure 61: Changes to Functional Test Cost: Number False Positives

To summarize the robustness tests, the system behaves rationally by adjusting its testing and packaging decisions to reflect changes in cost parameters. When the package cost is increased, the system tests more to avoid wasting resources by packaging bad dice. On the other hand, when the functional test cost is increased, the system tests less and packages more. If the test cost is set sufficiently low, then the system tests all dice. If the test cost is set sufficiently high, then the system tests none of the dice. In none of the test scenarios was it profitable to miss a good die, so the number of true positives was always equal to the total number of good dice, and the number of false negatives was always zero. This behavior is the result of two factors. First, given the quality of the wafers in the training set, all dice had a reasonable prior probability of being good. Second, the value of a good package was sufficient to justify packaging all dice based on these prior probabilities. For the tests reported in this section, the shifts in testing and packaging behavior involved a tradeoff between false positives and true negatives.

5.6 Tests with Holdout Data

In order to test the generality of the results, tests were performed on a separate data set consisting of 50 lots (1200 wafers) that had not been examined during any of the experiments presented above. For these tests a single stochastic model with four wafer classes was trained with the EM algorithm on 25 lots (600 wafers) of test data. A second stochastic model with four wafer classes was trained via Gibbs Sampling on the same training set. These models were embedded in the selective test structure and performance was measured on simulated testing of a separate data set of 25 lots (600 wafers). The exhaustive test policy was also evaluated on this test data set. The model performance is compared in terms of net profit, the number of tests, the number of packages, the number of true negatives, and the number of false positives.

The descriptive statistics for the test data set are presented in table 14.

Number Dice	125400
Number Good Dice	101516
Yield	80.95

Table 14: Test Data Statistics

The results from the wafer tests are presented in table 15.

					Number	Number
	-log	Profit	Number	Number	False	True
	likelihood		Tests	Packages	Positives	Negatives
Exhaustive	N/A	1174900	125400	101516	0	23884
Gibbs	138.7088	1218226	4899	122427	20911	2973
EM	138.3240	1218309	5194	122292	20776	3108
Test None	N/A	1215211	0	125400	23884	0

Table 15: Holdout Data Tests: Results

The results of the tests on the holdout data set are consistent with the results from the earlier tests. Both selective test approaches produce a net profit over the exhaustive test approach. The EM-trained model performs slightly more testing and packages slightly less than the Gibbs-trained model. This results in a small profit over the Gibbs-trained model. The number of false positives and true negatives are consistent with this interpretation: the EM-trained model is more accurate in its package decisions due to its increased testing. The consistency of these results with the earlier tests suggests that this approach generalizes beyond the data considered in this study. However, since both data sets were generated from the same production line these results only suggest broader applicability to other parts.

6. CONCLUSIONS

6.1 Summary

This dissertation explored the application of concepts from machine learning and decision theory to the problem of optimizing IC test. The optimal test problem was defined and a solution was developed and evaluated. The solution is a real-time selective test policy that determines which dice to test, which dice to package, and when to stop testing. The development of this policy involved several tasks. First, the selective test policy was formulated as a series of wafer test decisions. These decisions correspond to the control points in the testing process. Second, principles from decision theory (i.e., expected utility and value of information) and probabilistic reasoning (i.e., belief nets, influence diagrams, and inference algorithms) were employed to generate tractable decision models for each of the wafer test decisions. A key construct in the decision models is a stochastic representation of wafer test results. This takes the form of a naïve Bayes belief net that encodes the assumption that wafer test results are conditionally independent given an unobservable wafer class variable. Third, unsupervised machine learning techniques, specifically, the EM and Gibbs sampling algorithms, were adapted to generate the real-valued parameters of the belief net models. Since the wafer class variable was unobservable the parameter estimation task was formulated as an incomplete data problem and solved with a data-augmentation approach.

Evaluation consisted of empirical tests on historical test data from Hewlett-Packard Company. There were two sets of evaluation tests. The first set provided performance measures for the machine learning algorithms on the parameter estimation task. Various models were constructed and trained on the wafer test data. Performance was measured in terms of negative log likelihood and cross-validation was employed to investigate overfitting.

The principle results from the learning tests are the following:

1. The learning algorithms work. They were able to extract a signal from the wafer test data as measured by negative log likelihood scores. Furthermore, they were efficient. Convergence for both algorithms was typically reached in less than 50 iterations on a training set of 600 examples (approximately 3 minutes of CPU time with non-optimized LISP code on a sparc workstation).

2. Conditional independence is more accurate than independence, i.e., models that contained multiple wafer classes performed distinctly better than single-class models.

3. The EM algorithm results in overfitting to the training data set when the number of wafer classes exceeds eight. The Gibbs algorithm also exhibited some overfitting for large class dimensions, however the effect was less dramatic.

4. For the Gibbs sampling algorithm, performance can be improved by combining multiple models into ensembles. Performance, as measured by negative log likelihood, continued to improve as the ensemble size increased, however after about ten models the scale of the improvements was small.

The second set of evaluation tests consisted of wafer test simulations. The decision models with the learned parameters were embedded within the selective test policy and simulated testing was performed on historical wafer test data. A number of experiments were performed to investigate the behavior of the selective test policy. The primary results from these experiments are summarized below.

1. The selective test policy produced more net profit than either the exhaustive test policy or the no-test policy. The best EM-trained model contained eight wafer classes. This model performed slightly better than the model with four classes. Since the 4-class model exhibited a lower negative log likelihood score than the 8-

class model, this result demonstrates that likelihood rankings are not perfect indicators of net profit. The results from the Gibbs ensemble tests reinforce this observation. A single Gibbs model consistently outperformed the Gibbs ensembles, even though the ensembles exhibited better likelihood scores. Taken together these results suggest that the likelihood-profit relationship is not simple. A possible explanation is that, since the differences in likelihood scores were relatively small, the differences in profit are due to the stochastic nature of the task. Evidence of this can be observed in the results from the 1-class model. The 1-class model had a distinctly poorer negative log likelihood score and performed distinctively poorer on the wafer tests. The other models were similar is negative log likelihood scores and in net profits. The poorer performance of the Gibbs ensembles appears to be a true phenomenon and was observed on numerous tests. One possible explanation for this behavior is that the conditional ensemble likelihoods that result from belief updating are less accurate than the conditional likelihoods from a single model. Thus, although the initial likelihood scores suggest an accurate fit, once some observations are processed the resulting conditional likelihoods are less accurate than those from a single model. This phenomenon deserves more attention and is a recommended topic for future research.

2. For a single EM-trained model the myopic VOI stopping criterion produced near optimal performance. This reflects the benign nature of the wafer test environment; there are no catastrophic costs associated with a single test. It is always possible to test any die; it is always possible to stop at any point; it is never necessary to backtrack. Furthermore, all tests have the same cost and all tests provide some information. Thus myopic VOI is a reasonable choice of stopping criterion and these tests demonstrate how well it performs. The results also demonstrate that myopic-VOI stopping was not uniformly early or late. On some wafers myopic VOI resulted in too few tests being performed, on other wafers myopic VOI resulted in too few tests being performed. Thus no constant adjustment to myopic VOI will produce optimal stopping. It would be interesting to compare a 2-step myopic VOI policy to the current 1-step myopic VOI policy. However, given the nature of the

task and the near-optimal performance of the 1-step myopic VOI, it is doubtful that the benefits of a 2-step myopic VOI policy would justify the additional computational cost.

3. The training experiments demonstrated that good performance could be achieved with a small training set, and larger training sets produce better performance. Given the efficiency of the learning algorithms it seems reasonable to employ relatively large training sets, on the order of hundreds of wafers. However, the ability to produce good testing behavior from small training sets may be of benefit in other applications of the selective test policy. For example, rather than training the models off-line on historic data sets, in some cases it may be possible to improve performance by training in near real-time. In this type of application models can be generated on a lot-by-lot basis, or on a batch-by-batch basis. Thus a small sample of wafers from a lot (or batch) would be tested exhaustively and the results from these tests would then be used to train a model for testing the rest of the lot (or batch). This type of on-line training deserves further investigation.

4. One of the most significant results from the wafer experiments is that the selective test approach is responsive to individual wafers. For good wafers the system tests only a few of the dice before making package decisions. For bad wafers the system tests almost all of the dice. Thus the system is efficient with its testing. Few resources are expended on good wafers, yet abnormal wafers are detected and tested more thoroughly. The selective test policy produces more profits than either the exhaustive test or the no-test policy, and still maintains adequate process monitoring. If wafer starts are increased to take advantage of the reduced testing then the selective test policy will yield even larger profits. The relationship between the rate of wafer starts and the selective testing policy is another topic for future research.

5. Experiments with changes to utility model parameters demonstrated the robustness of the selective test policy. One of the benefits of the expected utility formulation is

that changes to processing costs do not necessitate explicit reengineering of the selective test policy. The new parameters are simply fed to the system and it adjusts its test and package decisions accordingly. The result is always rational behavior. Thus when test costs go down, the system tests more. When test costs go up, the system tests less. Similar behavior was observed when changes to the package cost were introduced. An increase in the package cost resulted in an increase in functional testing to avoid packaging bad dice. A decrease in the package cost resulted in a decrease in functional testing, since there was a smaller loss associated with packaging a bad die.

6. The tests on the holdout data set demonstrated the generality of the approach. The consistency of these results with the earlier tests suggests that this approach generalizes beyond the data considered in this study. However, since both data sets were generated from the same production line these results only suggest broader applicability to other parts. The application of the selective test approach to other types of wafers is a recommended topic for future research.

Perhaps the most significant result is a demonstrated proof of concept. The selective test approach is feasible to implement and produces rational testing behavior. In addition, these results show that the selective test approach can produce an expected net profit in manufacturing costs as compared to the current testing policy. Furthermore, the selective test approach can greatly reduce the amount of testing while maintaining an appropriate level of performance monitoring. Although more study is necessary before the selective test approach is ready for industrial implementations, this research suggests that such applications are possible and it provides a blueprint for their development.

6.2 Recommendations for Future Research

There are a number of directions in which this research can be developed. The following list summarizes those that appear to be most interesting and significant.

1. Perform further explorations into the relationship between ensemble methods and myopic VOI. Search for methods that exploit the good negative log likelihood scores of the Gibbs ensemble to improve testing behavior.

2. Investigate alternative model representations. Consider alternatives to the naïve Bayes belief net model of wafer test results. Explore more succinct model representations, models of re-locatable patterns, and models at higher levels of data granularity, e.g., lot-level models.

3. Consider variations on the inference procedures, in particular a multi-step lookahead.

4. Incorporate parametric test results into the testing policy. These results often carry useful information regarding wafer quality and their incorporation into the test decisions could yield performance benefits.

5. Extend the selective test approach to perform fault detection and diagnosis. Incorporate additional actions such as re-testing of dice. Consider additional inference tasks such as post-fabrication (off-line) yield analysis.

6. Enhance the utility model to include the explicit modeling of wafer starts and throughput.

7. Generalize the system to other parts and other manufacturing lines. Perform tests with data sets and utility models that represent distinctly different wafer types.

BIBLIOGRAPHY

Albin, S., and Friedman, D. 1989. The impact of clustered defect distributions in IC fabrication. *Management Science* 35(9): 1066-1078.

Bellman, R. E. 1957. *Dynamic programming.* Princeton, NJ: Princeton University Press.

Berger, J. 1985. Statistical decision theory and bayesian analysis. New York:Springer-Verlag.

Boyd, J. 1997. Moore's law also drives semiconductors. John Boyd's Computer Corner. The Japan Times. URL: http://www.japantimes.co.jp/home.html.

Buntine, W. 1994. Operations for learning with graphical models. *Journal of Artificial Intelligence Research* 22:159-225.

Buntine, W. 1996. A guide to the literature onlearning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering* 8:196-210.

Casela, G. and George, E. 1992. Explaining the Gibbs sampler. In *The American Statistician* 46(3): 167-174.

Cassandra, A.R., Kaelbling, L.P., and Littman, M.L. 1994. Acting optimally in partially observable domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, San Mateo, CA:Morgan Kaufmann 1023-1028.

Charniak, E. 1991. Bayesian networks without tears. *AI Magazine,* Menlo Park,CA:AAAI Press 12(4):50-63.

Cheeseman, P.; Self, M.; Kelly, J.; Taylor, W.; Freeman, D.; and Stutz, J. 1988. Bayesian classification. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, San Mateo, CA:Morgan Kaufmann 607-611.

Cobb, D., Castrucci, P., Everton, J., Schewe, M., Scoville, J., and Smith, G. l. 1994. The role of CIM for the fab of the future and fustom IC manufacturing. In Proceedings of *IEEE Advanced Semiconductor Manufacturing Conference*, 150-153.

Cooper, G. 1988. A method for using belief networks as influence diagrams. In *Proceedings of the 1988 Workshop on Uncertainty in AI.* San Mateo, CA:Morgan Kaufmann 55-63.

Cunningham, J. 1990. The use and evaluation of yield models in integrated circuit manufacturing. *IEEE Transactions on Semicionductor Manufacturing* 3(2):60-71.

Cunningham, S. 1995a. Applications of spatial statistics to semiconductor wafer defects. MA Thesis. Industrial-Engineering and Operations Research. University of California at Berkeley.

Cunningham, S. 1995b. The development and use of in-line yield estimates in semiconductor manufacturing. Ph.D. Dissertation. Industrial-Engineering and Operations Research. University of California at Berkeley.

D'Ambrosio, B. 1992. Incremental probabilistic inference. In *Ninth Annual Conference on Uncertainty on AI,* San Mateo, CA:Morgan Kaufmann 301-309.

Dean, T., and Wellman, M. 1991. *Planning and control.* San Mateo, CA:Morgan Kaufmann.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1976. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society* B 39:1-38.

Devroye, L. 1986. Non-Uniform random variate generation. New York:Springer-Verlag.

Dietterich, T. G. 1997. Machine-Learning research four current directions. *AI Magazine*, Menlo Park, CA: AAAI Press 18(4): 97-136.

Dislis, C., Ambler, A., Dear, I., and Dick, J. 1993. Economics in design and test. In *Proceedings of IEEE International Test Conference 1993*, 384-387.

Einhorn, H. and Hogarth, R., 1981. Behavioral decision theory: processes of judgement and choice. *Annual Reviews Psychology* 32:53-88.

Gefland, A.E., and Smith, A.F.M. 1990. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85:398-409.

Geman, S., and Geman, D. 1984. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6:721-741.

Hansen, M., Friedman, D., and Nair, V. 1997. Monitoring wafer map data from integrated circuit fabrication processes for spatially clustered defects. *Technometrics* 39(3).

Hanson, R., Stutz, J., and Cheeseman, P. 1990. Bayesian classification theory. NASA Technical Report FIA-90-12-7-01. NASA Ames Research Center, Moffet Field, CA.

Heckerman, D. 1996. A tutorial on learning with Bayesian networks, MSR-TR-95-06, Advanced Technology Division, Microsoft Research, Redmond, WA.

Heckerman, D. 1997. Bayesian networks for data mining. *Data Mining and Knowledge Discovery* The Netherlands:Kluwer Academic 1:79-119.

Heckerman, D.; Geiger, D.; and Chickering, D. M. 1995. Learning bayesian networks: the combination of knowledge and statistical data. *Machine Learning* 20: 197-243.

Henrion, M. 1988. Propagation of uncertainty by probabilistic logic sampling in Bayes' networks. In *Uncertainty in Artificial Intelligence, Vol. 2* (Lemmer and Kanal, editors). Amsterdam:North Holland.

Henrion, M. 1990. Towards efficient probabilistic diagnosis with a very large knowledge-base, In Proceeding of *AAAI Workshop on the Principles of Diagnosis*.

Henrion, M. 1991. Search-based methods to bound diagnostic probabilities in very large belief nets. In *Preoceedings of the Seventh Annual Conference on Uncertainty in Artificial Intelligence* (UAI-91). Menlo Park, CA:Morgan Kaufmann 142-150.

Holtzman, S. 1988. *Intelligent decision systems.* Reading, MA:Addison-Wesley.

Horvitz, E., Suermondt, H., and Cooper, G. 1989. Bounded conditioning: Flexible inference for decisions under scarce resources. In *Proceedings of the Fifth Conference on Uncertainty in Artificial Intelligence* (UAI-89). Menlo Park,CA:Morgan Kaufmann 182-193.

Horvitz, E. and Breese, J. and Henrion, M., 1991. Decision analysis and expert systems. *AI Magazine* San Mateo:AAAI Press 12(4):64-91.

Howard, R.A., 1960. *Dynamic programming and Markov processes.* Cambridge, Massachusetts:The MIT Press.

Howard, R. and Matheson, J., 1989. Influence diagrams. In *The principles and applications of decision analysis.* Eds. Howard and Matheson. Menlo Park, CA:Addison-Wesley.

Kass, R. and Raftery, A. 1994. Bayes factors. Technical Report no. 254. Department of Statistics, University of Washington.

Kohyama, S. 1994. Semiconductor technology crisis and challenges towards the year 2000. *IEEE Symposium on VLSI Technology Digest of Technical Papers*, 5-8.

Koppenhoefer, B., Wuerthner, S., Ludwig, L., Rosenstiel, W., Kuge, H., Hummel, M., and Federl, P. 1997. Analysis of electrical test data using a neural network approach. In Proceedings of *IEEE/SEMI Advanced Semiconductor Manufacturing Conference.*

Kumar, H. and Erjavic, S. 1993. Knowledge based testing. *Transactions of the IEEE International Test Conference* 900-917.

Lauritzen, S. and Spiegelhalter, D., 1988. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society* B 50.

Li, Z. and D'Ambrosio, B., 1992. Efficient inference in belief networks as a combinatorial optimization problem. Technical Report. Dept. of Computer Science, Oregon State University.

Lin, C. 1996. Shop floor scheduling of semiconductor wafer fabrication using real-time feedback control and predictions. Dissertation in Industrial Engineering and Operations Research, University of California, Berkeley.

Littman, M., Cassandra, A., and Kaebling, L. 1995. Learning policies for partially observable environments: scaling up. Proceedings of *The Twelfth International Conference on Machine Learning*, San Francisco,CA: Morgan Kaufmann 12:362-370.

Longtin, M., Wein, L., and Welsch, R. 1996. Sequential screening in semiconductor manufacturing, 1: exploiting spatial dependence. *Operations Research* 44(1): 173-195.

Luria, M, Adin, E., Moran, M., Yaffe, D., Haemek, M., and Kawski, J. 1993. Automatic defect classification using fuzzy logic. In Proceedings of *IEEE/SEMI Advanced Semiconductor Manufacturing Conference.*

Maniwa, T. 1996, Moore's Law Revisited. *Integrated system design Magazine*. URL: http://www.isdmag.com/Events/IEDM.html.

McLachlan, G. and Krishnan, T., 1997. *The EM algorithm and extensions*. New York:Wiley.

Madigan, D., Raferty, A.E., York, J.C., Bradshaw, J.M., and Almond, R.G. 1994. Strategies for graphical model selection. In *Selecting models from data: AI and statistics IV*, (P. Cheeseman and R.W. Oldford, editors) New York:Springer-Verlag 91-100.

Meindl, J. 1993. Opportunities for gigascale integration (GSI) beyond 2003. *IEEE/SEMI Advanced Semiconductor Manufacturing Conference.*

Monahan, G. 1982. *A survery of partially observable Markov decision processes: Theory, Models, and Algorithms.* Management Science, 28(1): 1-16.

Neal, R. 1993. Probabilistic inference using Markov chain Monte Carlo methods, Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto.

North, D. 1968. A tutorial introduction to decision theory. In *IEEE Transactions on Systems and Systems Science and Cybernetics,* Ssc-4, No. 3:117-127.

Oliver, R. and Smith, J., 1990. *Influence diagrams, belief nets, and decision analysis,* New York:John Wiley and Sons.

Ou, J. and Wein, L. 1996. Sequential screening in semiconductor manufacturing, II: exploiting lot-to-lot variability. *Operations Research* 44(1):196-205.

Parr, R. and Russell, S. 1995. Approximating optimal policies for partially observable stochastic domains. In *Proceedings of the international joint conference on artificial intelligence,* 1088-1094.

Pearl, J., 1988. Probabilistic reasoning in intelligent systems. Palo Alto, CA:Morgan Kaufmann.

Pearl, J., 1987. Evidential reasoning using stochastic simulation of causal models. *Artificial Intelligence* 32(2):245-258.

Raiffa, H., 1968. *Decision analysis.* Reading, MA:Addison-Wesley.

Russell, S. and Norvig, P. 1995. *Artificial intelligence: a modern approach.* New Jersey:Prentice Hall.

Savage, L., 1954. *The foundations of statistics.* New York:Wiley.

Shachter, R., 1986. Evaluating influence diagrams. *Operations Research,* 34(6):871-882.

Shachter, R., 1990. An ordered examination of influence diagrams. New York:Wiley *Networks* 20:535-563.

Shachter, R. and Peot, M., 1992. Decision making using probabilistic inference methods. In *Proceedings of the Eighth Workshop on Uncertainty on AI* Palo Alto,CA:Morgan Kaufmann 276-283.

Shachter, R., D'Ambrosio, B. and DelFavero, B., 1990. Symbolic probabilistic inference in belief networks. In *Proceedings Eighth National Conference on AI* Palo Alto, CA:Morgan Kaufmann 126-131.

Shepard, R., 1964. On subjectively optimum selection among multiattribute alternatives. In *Human Judgement and Optimality* New York:Willey 257-281.

Shwe, M. and Cooper, G., 1990. An empirical analysis of likelihood-weighting simulation on a large, multiply connected belief network. *In Proceedings of the 1990 Workshop on Uncertainty in AI* Palo Alto, CA:Morgan Kaufmann 498-508.

Simon, H. 1983. Alternate visions of rationality. In *Reason in Human Affairs.* Stanford University Press, Stanford, CA.

Smallwood, R. and Sondik, E. 1973. *The optimal control of partially observable Markov processes over a finite horizon.* Operations Research, 21(5):1071-1088.

Smyth, P. and Wolpert, D., 1997. Stacked density estimation. Technical Report No. 97-36, Information and Computer Science Department, University of Irvine, CA.

Smyth, P., Roden, J., Ghil, M., Ide, K., and Fraser, A., 1998. Detecting atmospheric regimes using cross-validated clustering. In *Proceeding of KDD-97*, 61-66.

Spiegelhalter, D.; Dawid, A; Lauritzen, S.; and Cowell, R. 1993. Bayesian analysis in expert systems. *Statistical Science* 8:219-282.

Spiegelhalter, D. and Lauritzen, S. 1990. Sequential updating of conditional probabilities on directed graphical structures. *Networks* New York:Wiley 20:579-605.

Tanner, M.A. 1991. Tools for statistical inference: observed data and data augmentation methods. In *Lecture Notes in Statistics.* Eds. Berger, J.; Fienberg, S.; Gani, J.; Krickeberg, K.; Olkin, I.; and Singer, B., Berlin:Springer-Verlag.

Tanner, M.A. and Wong, W.H. 1987. The calculation of posterior distributions by data augmentation, *Journal of American Statistical Association*, 82:528-550.

Tobin, K., Gleason, S., and Karnowski, T. 1998. Adaptation of the fuzzy k-nearest neighbor classifier for manufacturing automation, In Proceedings of *International Society for Optimal Engineering Phontonics West Symposium on Electrical Imaging*, San Jose,CA.

Tomlinson, W., Nurani, R., Burns, R., and Shanthikumar, J. 1997. Develoment of cost effective sampling strategy for in-line monitoring. In Proceedings of *IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, 8-12.

Turney, P. 1995. Data engineering for the analysis of semiconductor manufacturing data. *IJCAI-95 Workshop on Data Engineering for Inductive Learning.*

Tversky, A., and Kahneman, D., 1981. The framing of decisions and the psychology of choice. *Science* 211:945-950.

Van Zant, P. 1997. *Microchip fabrication: a practical guide to semiconductor processing, third edition.* New York: McGraw-Hill.

White, D., 1978. *Finite dynamic programming.* New York: Wiley.

White, D. 1993. A survey of applications of Markov decision processes. *Journal of the Operational Research Society*, 44(11): 1073-1096.

Zhang, W., and Milor, L. 1993. A neural network based approach for surveillance and diagnosis of statistical parameters in IC manufacturing process. In Proceedings *IEEE/SEMI International Semiconductor Manufacturing Science Symposium.*

Zinke, K., Nasr, M., Hicks, A., Crawford, M., and Zawrotny, R. 1997. Yield enhancement techniques using neural network pattern detection. In Proceedings of *IEEE/SEMI Advanced Semiconductor Manufacturing Conference.*

Zorich, R. 1991. *Handbook of Quality Integrated Circuit Manufacturing.* San Diego, CA:Academic Press.