Trans. Syst., Man, Cybern. A, Syst., Humans, vol. 34, no. 1, pp. 38–51, Jan. 2004.

[2] Z. W. Li, M. Uzam, and M. C. Zhou, "Comments on 'Deadlock prevention policy based on Petri nets and siphons'," *Int. J. Prod. Res.*, vol. 42, no. 24, pp. 5253–5254, Dec. 2004.

[3] Y. S. Huang, M. D. Jeng, X. L. Xie, and S. L. Chung, "Deadlock prevention policy based on Petri nets and siphons," *Int. J. Prod. Res.*, vol. 39, no. 2, pp. 283–305, Jan. 2001.

[4] J. Ezpeleta, J. M. Colom, and J. Martinez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Trans. Robot. Autom.*, vol. 11, no. 2, pp. 173–184, Apr. 1995

[5] M. P. Fanti and M. C. Zhou, "Deadlock control methods in automated manufacturing systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 34, no. 1, pp. 5–22, Jan. 2004.

[6] Z. W. Li and M. C. Zhou, "Comparison of two deadlock prevention methods for different-size flexible manufacturing systems," *Int. J. Intell. Control Syst.*, vol. 10, no. 3, pp. 235–243, Sep. 2005.

[7] Z. W. Li, W. Ding, and R. M. Zhu, "On deadlock prevention in case of failures in FMS," *Int. J. Manuf. Technol. Manag.* vol. 8, no. 1–3, pp. 58–74, 2006.

Application of an Ordinal Optimization Algorithm to the Wafer Testing Process

Shin-Yeu Lin and Shih-Cheng Horng

***Abstract*—In this correspondence, we have formulated a stochastic optimization problem to find the optimal threshold values to reduce the overkills of dies under a tolerable retest level in wafer testing process. The problem is a hard optimization problem with a huge solution space. We propose an ordinal optimization theory-based two-level algorithm to solve for a vector of good enough threshold values and compare with those obtained by others using a set of 521 real test wafers. The test results confirm the feature of controlling the retest level in our formulation, and the pairs of overkills and retests resulted from our approach are almost Pareto optimal. In addition, our approach spends only 6.05 min in total in a Pentium IV personal computer to obtain the good enough threshold values.**

***Index Terms*—Genetic algorithm (GA), neural network, ordinal optimization (OO), overkill, retest, stochastic optimization, wafer probing.**

I. Introduction

The wafer fabrication process is a sequence of hundreds of different process steps, which results in an unavoidable variability accumulated from the small variations of each process step. Thus, to avoid incurring the significant expense of assembling and packaging chips that do not meet specifications, the wafer probing in the manufacturing process becomes an essential step to identify flaws early.

Wafer probing establishes a temporary electrical contact between test equipment and each individual die (or chip) on a wafer to determine the goodness of a die. In general, an 8-in wafer may consist of 600 to 15 000 dies, and each die is a chip of integrated circuits. Although there exist techniques such as the statistical process control (SPC) [1], [2] for monitoring the operations of the wafer probes, the probing errors may still occur in many aspects and cause some good dies being over killed; consequently, the profit is diminished. Thus, reducing the number of *overkills* is always one of the main objectives in wafer testing process. The key tool to identify or save overkills is *retest*, which is an additional probing on the problematic die. However, retest is a major factor for decreasing the *throughput*. Thus, the overkill and the retest possess inherent conflicting factors, because reducing the former can gain more profit, however, at the expense of increasing the latter, which will degrade the throughput. Consequently, to save more overkills using less retests is a goal of the wafer testing process.

Deciding whether to go for a retest is a decision problem. In current wafer testing processes, this decision is made based on whether the number of good dies and the number of *bins*[1] in a wafer exceed the corresponding threshold values. Manually adaptive adjustments of the *threshold values* based on engineering judgment, three-sigma limit [3], or a looser six-sigma limit are currently used in some semiconductor manufacturing companies. The purpose of this correspondence is using a systematic approach to determine these threshold values. We first formulate a stochastic optimization problem on the threshold values. Since the formulated stochastic optimization problem consists of a huge decision-variable space as will be seen in Section III, this makes the problem become a hard optimization problem. Thus, to cope with the enormous computational complexity, we propose an ordinal optimization (OO) theory-based two-level algorithm to solve the formulated problem for a good enough solution.

II. Problem Statements and Mathematical Formulation

A. Testing Procedures

In this section, we employ typical testing procedures used in a local world-renowned wafer foundry. Fig. 1 shows the flow chart of the real and simulated testing procedures. All the solid blocks represent the real testing procedures, while the dashed blocks are added for the purpose of computer simulation. The operation of the real testing procedures is briefly described in the following.

For every wafer, the wafer probing is performed twice, as shown in the solid square marked by I in Fig. 1. The second probing applies only to those dies failed in the first one. A die is considered to be good if it is good in either probing. If a die is detected to have bins in both tests, the bin detected in the second probing is taken as the bin of that die. We let $g_j(\overline{g}_j)$ denote the number of good (bad) dies in wafer j, and let b_{jk} denote the number of dies of bin k in wafer j. Assume there are K types of bins in a wafer, then $\overline{g}_j = \sum_{k=1}^{K} b_{jk}$ and $g_j = \mathrm{TD}_j - \overline{g}_j$, as shown in the square marked by II in Fig. 1, where TD_j denotes the total number of dies in wafer j. Following the two times of wafer probing and the calculation of g_j and $\overline{g}_j$, a two-stage checking on the number of good dies is performed to determine the necessity of carrying out a retest, i.e., an additional wafer probing. The mechanism of the two-stage checking described in the part of the testing procedures enclosed in the dotted contour can be summarized below. We let $g_{W\min}$ denote the threshold value for the *lower bound* of the number of good dies in a wafer to determine whether to pass or hold the wafer; we let $n_{k\max}$, $k = 1, \ldots, K$, denote the threshold value for the *upper bound* of the number of dies of bin k in the hold wafer to determine whether to perform a retest. If $g_j \geq g_{W\min}$, we pass wafer j, as shown in the diamond-shape block marked by III.a and the square marked by III.c; otherwise, we will hold this wafer and check its bins. For the hold wafer j, if $b_{jk} \leq n_{k\max}$ for all k, then wafer j will be passed, as shown in the diamond-shape

Manuscript received September 30, 2004; revised April 25, 2005 and May 4, 2005. This work was supported in part by the National Science Council in Taiwan, R.O.C., under Grant NSC93-2218-E-009-036. This paper was recommended by Associate Editor P. K. Willet.

The authors are with the Department of Electrical and Control Engineering, National Chiao Tung University, Hsinchu 300, Taiwan, R.O.C. (e-mail: sylin@cc.nctu.edu.tw; schong.ece90g@nctu.edu.tw).

Digital Object Identifier 10.1109/TSMCA.2006.878965

[1]A *bin* denotes a type of circuitry defect in a die. There are various types of bins, and a die of any type of bin is considered to be a bad die.

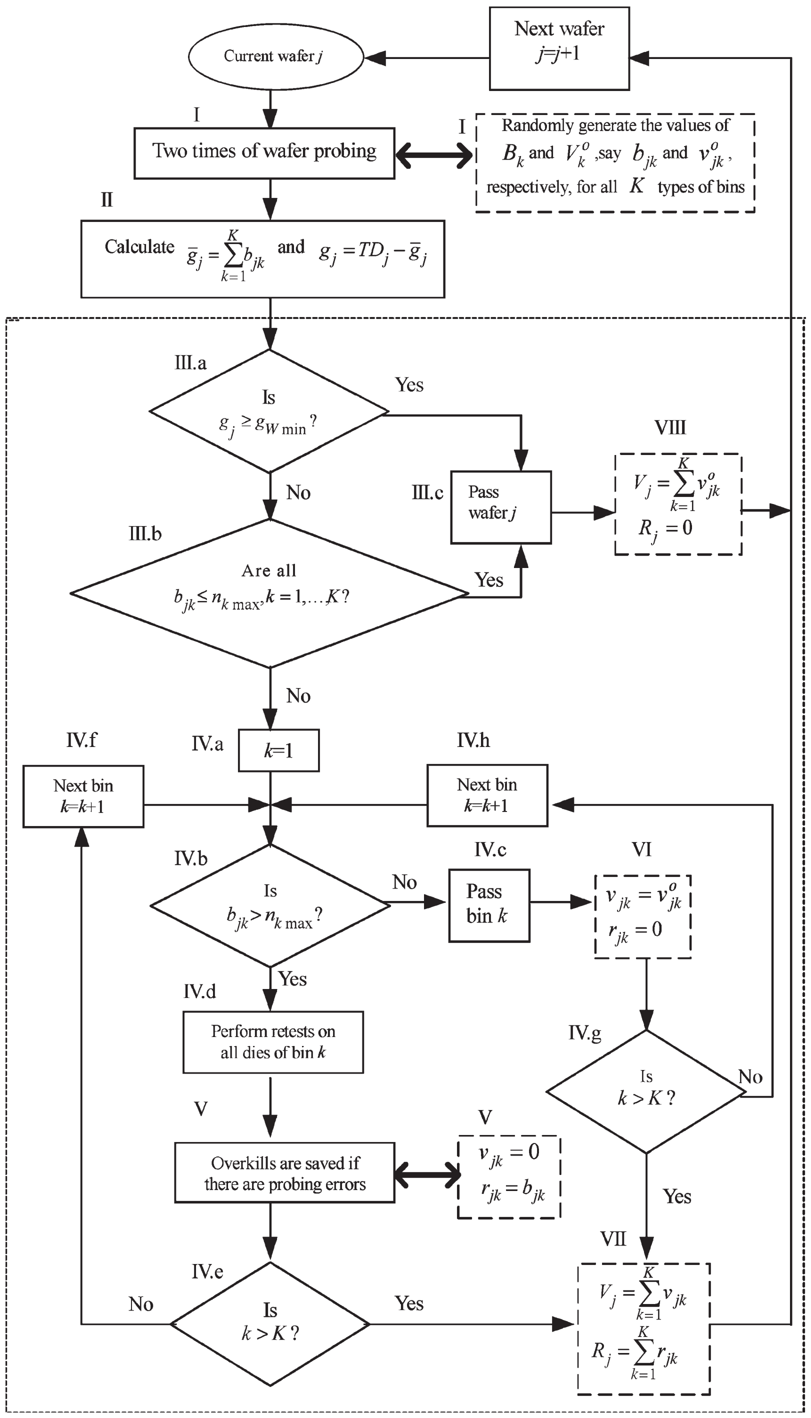
Current wafer j
Next wafer j=j+1
I
Two times of wafer probing
I
Randomly generate the values of B_k and V_k^o, say b_{jk} and v_{jk}^o, respectively, for all K types of bins
II
Calculate $\bar{g}_j = \sum_{k=1}^{K} b_{jk}$ and $g_j = TD_j - \bar{g}_j$
III.a
Is $g_j \geq g_{W\min}$?
Yes
No
III.c
Pass wafer j
VIII
$V_j = \sum_{k=1}^{K} v_{jk}^o$
$R_j = 0$
III.b
Are all $b_{jk} \leq n_{k\max}, k = 1, \ldots K$?
Yes
No
IV.f
Next bin k=k+1
IV.a
k=1
IV.h
Next bin k=k+1
IV.b
Is $b_{jk} > n_{k\max}$?
No
IV.c
Pass bin k
VI
$v_{jk} = v_{jk}^o$
$r_{jk} = 0$
Yes
IV.d
Perform retests on all dies of bin k
IV.g
Is $k > K$?
No
V
Overkills are saved if there are probing errors
V
$v_{jk} = 0$
$r_{jk} = b_{jk}$
Yes
VII
IV.e
No
Is $k > K$?
Yes
$V_j = \sum_{k=1}^{K} v_{jk}$
$R_j = \sum_{k=1}^{K} r_{jk}$

Fig. 1. Flow chart of the real and simulated wafer testing procedures.

block marked by III.b and the square marked by III.c. However, if the hold wafer j consists of any bin k with $b_{jk} < n_{k\max}$, retests will be performed for all dies of bin k in wafer j to check for possible probing errors, as shown in the diamond-shape block and square marked by IV.b and IV.d. Then, the overkills will be saved when there are probing errors, as shown in the square marked by V. For bin k in the hold wafer j with $b_{jk} \leq n_{k\max}$, we pass it, as shown in the diamond-shape block and square marked by IV.b and IV.c. This threshold value checking process will continue until all bins are checked as indicated in the diamond-shape blocks and squares marked by IV.e, IV.f, IV.g, and IV.h.

B. Computer Simulation of the Testing Procedures

1) Simulation Model for the Two-Times Wafer Probing: Since we cannot perform the real wafer probing in the computer, for the purpose of simulation, we need to build up a simulation model for the two times wafer probing. We let B_k denote the discrete random variable for the number of dies of bin k in a wafer. Since $P(B_k = n)$ can be provided by the real data, we can randomly generate the value of B_k for a wafer based on the discrete probability mass function $P(B_k = n)$.

Each die of bin k can be either an actual bin caused by manufacturing errors or an overkill caused by testing errors. Thus, we can treat the overkills in B_k as a binomial random variable with probability

p_k, which represents the probability of overkills in dies of bin k and can be provided by real data. We let V_k^o denote the random variable for the number of overkills in B_k. Then, once the value of B_k is randomly generated, we can randomly generate the value of V_k^o based on a binomial probability distribution with probability p_k.

2) Simulation of the Testing Procedures: We let b_{jk} and v_{jk}^o denote the values generated from the random variables B_k and V_k^o for wafer j, respectively. The two times wafer probing in Fig. 1 will be replaced by the random generator of B_k and V_k^o shown in the dashed square marked also by I in Fig. 1. The dashed squares in Fig. 1 except for the one mentioned above are for calculating the number of overkills and retests resulted from the simulated testing procedures. In contrast to v_{jk}^o, we let v_{jk} denote the number of overkills for bin k of wafer j after completing the testing procedures and let r_{jk} denote the corresponding number of retests. In the testing procedures, although we may pass the wafer when the threshold value test is a success, there may be overkills. We let V_j and R_j denote the total number of overkills and retests in wafer j, respectively. Thus, for the passed wafer j, $V_j = \sum_{k=1}^{K} v_{jk}^o$, and $R_j = 0$, as shown in the dashed square marked by VIII in Fig. 1. The same logic applies to the passed bin k of the hold wafer j that $v_{jk} = v_{jk}^o$ and $r_{jk} = 0$, as shown in the dashed square marked by VI in Fig. 1. However, for any retested bin, the probability of any unidentified overkill is extremely small, because the dies had been probed three times, which include two times wafer probing before any retest. Thus, for any retested bin k, $r_{jk} = b_{jk}$ and we assume $v_{jk} = 0$, because the overkills are saved, as shown in the dashed square marked also by V in Fig. 1; the solid square marked by V will be replaced by this dashed square in the simulated testing procedures. Once all the threshold value tests for all bins of the hold wafer j are completed, we can compute V_j and R_j, as shown in the dashed square marked by VII in Fig. 1. The resulting values of V_j and R_j of wafer j will be used to calculate $E[V] = (1/L)\sum_{j=1}^{L} V_j$ and $E[R] = (1/L)\sum_{j=1}^{L} R_j$, which represent the average overkills and retests per wafer, respectively, and L denotes the total number of tested wafers.

C. Problem Formulation

From Fig. 1, we see that if we increase $g_{W\min}$ while decreasing $n_{k\max}$, which is setting more stringent threshold values, there will be more retests and less overkills. This shows a conflicting nature between the overkills and retests. Thus, to reduce overkills under a tolerable level of retests, we will set minimizing the average number of overkills per wafer, $E[V]$, as our objective function while keeping the average number of retests per wafer, $E[R]$, under a satisfactory level. Thus, our problem for determining the threshold values can be formulated as the following constrained stochastic optimization problem:

$$\min_{x\in X} E[V]$$
$$\text{subject to } \{\text{simulated wafer testing procedures in Fig. 1}\}$$
$$E[R] \le r_{\mathrm{T}} \tag{1}$$

where $x \equiv [g_{W\min}, n_{k\max}, k = 1, \ldots, K]$ denotes the vector of threshold values, which is the vector of decision variables; X denotes the decision variable space; r_{T} denotes the tolerable average number of retests per wafer.

Remark 1: a) The value of r_{T} can be determined by the decision maker based on the economic situation. When the chip demand is weak, the throughput, in general, is not critical in the manufacturing process; therefore, we can allow a larger r_{T} so as to save more overkills to gain more profit. On the other hand, if the chip demand is strong, then the throughput is more important, and we should set the value of r_{T} smaller. Taking the chip demand into account is a *distinguished feature* of the proposed formulation.

b) It is possible to pursue the relationships between the number of retests and the throughput. Then, if we can derive the profit in terms of the throughput and the overkill, we can formulate an unconstrained optimization problem to maximize the profit. However, the relationships between the profit and throughput are very complicated due to the status of chip demand. For instances, when the chip demand is strong, larger throughput implies higher profit; on the other hand, if the chip demand is weak, larger throughput will cause inventory problem, which will hurt the profit. Therefore, the current formulation is simple and direct for a decision maker.

Since the constraint on $E[R]$ shown in (1) is a soft constraint in a sense, we can use a penalty function to relax that constraint and transform (1) into the following unconstrained stochastic optimization problem:

$$\min_{x\in X} E[V] + P \times (E[R] - r_{\mathrm{T}})$$
$$\text{subject to } \{\text{simulated wafer testing procedures in Fig. 1}\} \tag{2}$$

where P denotes a continuous penalty function for the constraint $E[R] \le r_{\mathrm{T}}$.

III. Two-Level OO Algorithm

The size of the decision variable space X in (2) is huge; for example, for an 8-in wafer, which consists of, say, 2500 dies, the possible ranges of the integer values $g_{W\min}$ and $n_{k\max}$ are [1, 2500] and [1, 2500], respectively. Consequently, for the number of bin types $K = 12$, the size of X will be more than 10^{30}. The evaluation of the performance of each vector of decision variables requires a lengthy stochastic simulation of the testing procedures. Therefore, any global searching techniques for solving the simulation optimization-type problem (2) will be very computationally expensive. To cope with the computational complexity of this problem, we propose an OO theory-based two-level algorithm to solve for a good enough solution with high probability instead of searching the best for sure.

The existing searching procedures of OO can be summarized in the following [4].

1) Uniformly or randomly select N, say 1000, vectors of decision variables from X.
2) Evaluate and order the N vectors using an approximate model, then pick the top s, say 35, vectors to form the estimated good enough subset.
3) Evaluate and order all the s vectors obtained from 2) using the exact model, then pick the top k (≥ 1) vectors.

The basic idea of the OO theory is based on the following observation. The performance order of the decision variables is likely preserved even evaluated using a crude model. Thus, the OO approach can reduce the searching space using cheaper evaluation to save computational time as indicated in 2), and the best vector of decision variables obtained in 3) is proved in [4] to be a good enough, top 5%, solution among N $(= 1000)$ with probability 0.95.

From the above description, we see that the quality of the good enough solution heavily depends on the quality of the randomly selected N vectors of decision variables. Thus, to improve the existing OO searching procedures, we can apply the OO theory to select N roughly good vectors of decision variables from X, to ensure the top 5% solutions among N to be the good enough solutions of X. This is what we called the first-level OO approach for replacing the existing searching procedure 1). Combining first level approach with the existing searching procedures 2) and 3) forms a two-level OO algorithm.

A. Constructing a Metamodel for (2)

The very first step for choosing N roughly good vectors from X should be constructing a *metamodel* or *surrogate model* for the considered stochastic simulation optimization-type problem. There are various techniques to approximate the relationships between the inputs and outputs of a system such as the linear regression, response transformation regression, projection-pursuit regression, and artificial neural network (ANN) [5], etc. Among them, ANN is considered to be a universal function approximator [6] due to its genetic, accurate, and convenient property to model complicated nonlinear input–output relationships. The ANN not only approximates the continuous functions well [7], [8], but is also used to construct metamodels for discrete event simulated systems in [9] and [10]. Since what we care here is the performance *order* of the solution rather than the performance *value* as considered in [9] and [10], we can trade off the accuracy of the ANN-based metamodel with the training time by using a simple ANN with a reasonable size of training data set. Two simple feed forward two-layer ANNs are employed here. One is to approximate the relationships between $x \in X$ and the corresponding $E[V]$, and the other is for $x \in X$ and $E[R]$. In these two ANNs, there are 16 neurons with a hyperbolic tangent sigmoid function in the first layer, and one neuron with linear function in the second layer. We obtain the set of training data for the two ANNs by the following two steps.

1) Narrow down the decision-variable space X by excluding the irrational threshold values and denote the reduced decision variable space by $\hat{X}$.[2]
2) Uniformly select M vectors from $\hat{X}$ and compute the corresponding outputs $E[V]$ and $E[R]$ using a stochastic simulation of the testing procedures shown in Fig. 1.

As indicated above, M need not be a very large value. The objective value of (2) can be computed based on the values of $E[V]$ and $E[R]$. Thus, we can obtain M pairs of decision variables and the corresponding objective values for (2). To speed up the convergence of the back propagation training, we employed the Levenberg–Marquardt algorithm [11] and the scaled conjugate gradient algorithm [12] to train the ANNs for $E[V]$ and $E[R]$, respectively. Stopping criteria of the above two training algorithms are when any of the following two conditions occurs.

1) Sum of the mean-squared errors is smaller than 10^{-5}.
2) Number of epochs exceeds 500.

Once these two ANNs are trained, we can input any vector x to the two ANNs to estimate the corresponding $E[V]$ and $E[R]$, which will be used to compute the objective value of (2). This forms our metamodel to estimate the objective value of (2) for a given vector of decision variables x.

B. Using GA to Select N Roughly Good Vectors of Decision Variables From $\hat{X}$

By the aid of the above ANN model, we can search N roughly good vectors of decision variables from $\hat{X}$ using heuristic global searching techniques.

Since the searching techniques of genetic algorithms (GAs), evolution strategies (ES), and evolutionary programming (EP) [13] improve a pool of populations from iteration to iteration, they should best fit our needs. For the sake of explanation and easier implementation, we employ the GA [14, Ch. 14] as our searching tool.

[2]The threshold values, $g_{W\,\min}$ and $n_{k\,\max}$, should lie in a reasonable range determined by the corresponding average values of g_j and b_{jk} collected from a wafer foundry, respectively.

The coding scheme of the GA we employed to represent all the vectors in $\hat{X}$ is rather straightforward, because each component of the vector x is an integer. We start from I, say 5000, randomly selected vectors from $\hat{X}$ as our initial populations. The fitness of each vector is set to be the reciprocal of the corresponding objective value of (2) computed based on the outputs of the two ANNs. The members in the mating pool are selected from the pool of populations using roulette wheel selection scheme. Seventy percent of the members in the mating pool are randomly selected to serve as parents for crossover. We use a single-point crossover scheme and assume the mutation probability to be 0.02. We stop the GA when the iteration number exceeds 30. After the applied GA converges, we rank the final I populations based on their fitness and pick the top N populations, which are the N roughly good vectors of decision variables.

Remark 2: Although there exists in-depth analysis of the approximation errors for an ANN to approximate continuous functions [7], [8], the accuracy of approximating the input and output relationships of a discrete event simulated system is usually addressed using empirical results [9], [10]. Thus, it is not surprising that we do not get any analytical result for the quality of the N vectors selected above. However, similar to the study in [4], we assume various magnitudes of modeling noise of uniform distribution to represent the approximation errors caused by the proposed ANN-based metamodel and make the following simple experiments to compare the quality of the N vectors selected by GA based on the ANN model with those selected in random from the solution space. We let $U[-0.1, 0.1]$ denote the uniform distribution of a random noise ranging from -0.1 to 0.1 to be added to the normalized performance, i.e., the normalized objective value, of the exact model. The normalized performance for all solutions in a solution space is equally spaced ranging from 0 to 1, with 0 as the top performance. In [4], a normalized ordinal performance curve (OPC) is used to describe the performance structure of all the solutions in a solution space. Assume $|X| = 10^{30}$, $N = 1000$, we carried out a Monte Carlo study for vast number of OPCs similar to that in [4] for an assumed noise distribution and pick the top N vectors using GA. We found the following results. For the modeling noise distribution, $U[-0.01, 0.01]$, $U[-0.1, 0.1]$, and $U[-0.5, 0.5]$, the top 5% solutions in N, which are selected by GA, is at least a top 10^{-6}%, top 10^{-3}%, and top 10^{-2}% solution in X with probability 0.95, respectively. However, the top 5% solutions in N, which are selected in random, is at best, i.e., assuming no modeling noise, a top 5% solution in X only. Therefore, we have greatly improved the quality of the N vectors by replacing the existing searching procedure 1).

C. Using an Approximate Model for Selecting the Estimated Good Enough Subset

Starting from the N vectors of decision variables obtained in Section III-B, we will proceed with 2) of the OO searching procedure to compute the objective value of (2) for each vector using an approximate model. As indicated in [15], this approximate model can be a stochastic simulation with moderate number of test wafers, which is to carry out the testing procedures shown in Fig. 1 for L_s, say 300, wafers. We will then order the N vectors of decision variables based on the obtained estimated objective values of (2) and choose the top s vectors that form the estimated good enough subset.

D. Using the Exact Model to Determine the Good Enough Solution

We will compute the objective value of (2) for each of the s vectors in the estimated good enough subset using the exact model that is a stochastic simulation with sufficiently large number of test wafers that makes the estimated objective value sufficiently stable. This exact model is similar to the approximate model mentioned above however

TABLE I
GOOD ENOUGH VECTOR OF THRESHOLD VALUES AND THE AVERAGE OVERKILL PERCENTAGE OF THE CONSIDERED PRODUCT FOR THREE DIFFERENT r_T'S

r_T / Good enough vector of threshold values	10	30	50
$g_{W\min}$	146	163	176
$n_{1\max}$	7	3	8
$n_{2\max}$	3	8	5
$n_{3\max}$	6	6	6
$n_{4\max}$	5	6	5
$n_{5\max}$	51	43	34
$n_{6\max}$	32	23	16
$n_{7\max}$	7	7	3
$n_{8\max}$	7	3	6
$n_{9\max}$	4	3	4
$n_{10\max}$	4	3	2
$n_{11\max}$	3	3	2
$n_{12\max}$	2	9	5
* $\frac{E[V]}{TD}\cdot 100\%$	1.86%	1.07%	0.27%

* *TD* : the total number of dies in a wafer of the considered product.

replacing L_s by $L_e(\gg L_s)$ wafers. Then, the vector associated with the smallest objective value of (2) among s is the good enough solution that we seek.

IV. TEST RESULTS AND COMPARISONS

Our simulations are based on the following data collected from a practical product of a local world-renowned wafer foundry. The product is made in 6-in wafers. Each wafer consists of 203 dies. There are 12 bins in the wafers of this product. The probability mass function $P(B_k = n)$, $k = 1,\ldots,12$, and the probability of the number of overkills in bin k, p_k, $k = 1,\ldots,12$, are given. The yield rate of this product is 68%. The decision-variable space $X = \{x(= [g_{W\min}, n_{k\max}, k = 1,\ldots,K])|g_{W\min} \in [1, 203]$, $n_{k\max} \in [1, 203], k = 1,\ldots,12\}$. We used the sigmoid-type function as our penalty function P in (2), i.e., $P = \eta(1/1 + e^{-(E[R]-r_T)})$, for $E[R] > r_T$ and 0 otherwise, where η ($\cong 0.1594$) is a normalized coefficient such that $\eta = (\max_{i\in\{1,\ldots,M\}} E[V_i]/\max_{i\in\{1,\ldots,M\}} E[R_i])$.

We set $\hat{X} = \{x(= [g_{W\min}, n_{k\max}, k = 1,\ldots,K])|g_{W\min} \in [50, 203], n_{k\max} \in [1, 6\ \mu_k], k = 1,\ldots,12\}$, where μ_k is the mean of the number of dies of bin k. The parameters in the proposed two-level algorithm are set as follows: $L_s = 300$, $L_e = 10\,000$, $M = 1000$, $I = 5000$, $N = 1000$, and $s = 35$. We have simulated three cases of different r_Ts, which are 10, 30, and 50. It should be noted that all the test results shown in this section are simulated in a Pentium IV PC using Borland C++.

The good enough vector of threshold values and the average overkill percentage for the three cases of r_T we obtained from the two-level algorithm are shown in Table I. The CPU time consumed in each case plus the training time is approximately 6.05 min. From Table I, we can observe that when r_T increases, the values of $g_{W\min}$ increase, as shown in row 2, and the values of leading $n_{k\max}$, $k = 5$ and 6, which account for most of the retests decrease, as shown in rows 7 and 8, respectively. This indicates that if we allow more retests (that is increasing r_T), we can set more stringent threshold values (that are increasing $g_{W\min}$ and decreasing the leading $n_{k\max}$s), so as to save more overkills (that is the decreased average overkill percentage), as indicated in the last row of Table I.

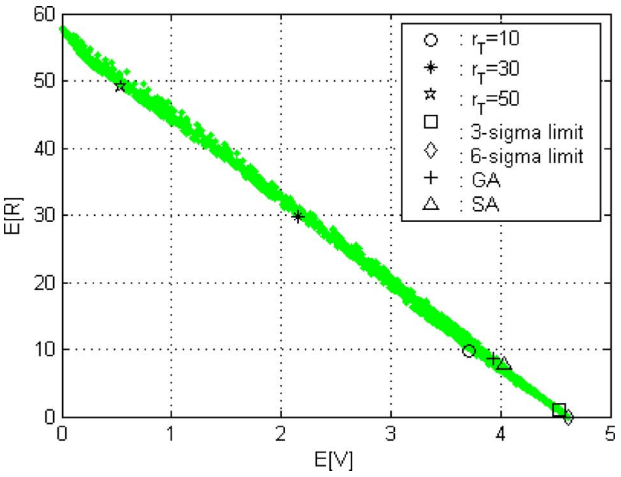

Fig. 2. Resulted ($E[V]$, $E[R]$) pairs of the 521 test wafers based on the vector of threshold values determined by the two-level algorithm, the random generator, the three-sigma limit, and the six-sigma limit. (Color version available online at http://ieeexplore.ieee.org.)

To demonstrate the real-world performance of the vector of threshold values obtained by the two-level algorithm for the three cases shown in Table I, we use 521 real test wafers, whose number of dies of all bins, b_{jk}, $j = 1,\ldots,521$, $k = 1,\ldots,12$, and overkills before retest, v^o_{jk}, $j = 1,\ldots,521, k = 1,\ldots,12$, are known. The corresponding results of the pair of the average overkills per wafer, $E[V](= (1/521)\sum_{j=1}^{521} V_j)$, and the average retests per wafer, $E[R](= (1/521)\sum_{j=1}^{521} R_j)$, for these 521 test wafers are shown in Fig. 2 as the points marked by "$\star$," "$*$," "$\circ$" with the corresponding r_T shown on the top right corner of the figure. We also use 2000 randomly selected vectors of threshold values to test the same 521 wafers; the resulted pairs of $E[V]$ and $E[R]$ are shown as the points marked by "$\bullet$" in Fig. 2. We see that for $E[R] \le 10$, the $E[V]$ resulted by the good enough vector of threshold values obtained by the two-level algorithm is almost the minimum compared with those resulted by the randomly selected vectors of threshold values. Similar conclusions can be drawn for the cases of $r_T = 30$ and 50. Since reducing overkills and retests have conflicting nature, the considered unconstrained stochastic optimization problem (2) possesses Pareto optimal solutions. From Fig. 2, we can see that the results we obtained for the cases of $r_T = 10$, 30, and 50 are almost on the boundary of the region resulted from the randomly generated vectors of threshold values; this implicit boundary represents the ($E[V]$, $E[R]$) pairs resulted by the Pareto optimal vectors of threshold values. We also use the three-sigma limit and the six-sigma limit to determine the threshold values such that $g^{3\sigma}_{W\min} = \mu_g - 3\sigma_g$, $n^{3\sigma}_{k\max} = \mu_k + 3\sigma_k$, $k = 1,\ldots,12$, and $g^{6\sigma}_{W\min} = \mu_g - 6\sigma_g$, $n^{6\sigma}_{k\max} = \mu_k + 6\sigma_k$, $k = 1,\ldots,12$, where μ_g and σ_g, the mean and standard derivation of the number of good dies in a wafer, and μ_k and σ_k, the mean and standard derivation of the number of dies of bin k, are obtained from the data set of 521 test wafers. Using these threshold values to test the same set of 521 test wafers, the resulted ($E[V]$, $E[R]$) pairs from the three-sigma limit and the six-sigma limit are also shown in Fig. 2 marked by "$\square$" and "$\Diamond$," respectively. For $E[R] \le 10$, we can see that our method will save 22% and 24% more overkills than the three-sigma limit and the six-sigma limit, respectively. Considering the vast number of dies manufactured per month, the increased profit due to saving overkills will be too large to neglect. Furthermore, both the three-sigma limit and the six-sigma limit do not generate the Pareto optimal solution for (2), and they cannot control the level of retests like ours. We have also used typical GA and simulated annealing (SA) [13] algorithm to solve (2) for the case of $r_{\mathrm{T}} = 10$. As indicated at the beginning of Section III, the global searching techniques are computationally expensive in solving (2). We stop the GA and SA when they consumed 50 times of the CPU time consumed by the two-level algorithm, and the objective values of (2) they obtained are still 5.4% and 8.1% more than the final objective value obtained by the two-level algorithm, respectively. Using the threshold values they obtained to test the 521 wafers, the resulted ($E[V]$, $E[R]$) pairs are marked by "$+$" and "Δ" in Fig. 2. We found that using two-level algorithm, we can save 6.2% and 8.6% more overkills than using the GA and SA for $E[R] \leq 10$, respectively. In addition, both GA and SA do not generate the Pareto optimal solution, because the best so far solution they obtained for 5 h of CPU time are still far away from the optimal solution of (2).

V. Conclusion

The proposed formulation for reducing overkills and retests is not limited to the testing process of a foundry; it can easily adapt to any general testing procedures. The proposed OO theory-based two-level algorithm is not limited to the problem considered in this correspondence. In fact, it can be used to solve any hard optimization problem that requires lengthy computational time to evaluate the performance of a decision variable.

References

[1] K. R. Skinner, D. C. Montgomery, G. C. Runger, J. W. Fowler, D. R. McCarville, T. R. Rhoads, and J. D. Stanley, "Multivariate statistical methods for modeling and analysis of wafer probe test data," *IEEE Trans. Semicond. Manuf.*, vol. 15, no. 4, pp. 523–530, Nov. 2002.

[2] S. Muriel, P. Garcia, O. Marie-Richard, M. Monleon, and M. Recio, "Statistical bin analysis on wafer probe," in *Proc. IEEE/SEMI Advanced Semicond. Manuf. Conf. and Workshop*, Munich, Germany, Apr. 2001, pp. 187–192.

[3] D. C. Montgomery, *Introduction to Statistical Quality Control*, 5th ed. New York: Wiley, Jul. 2004.

[4] T. W. E. Lau and Y. C. Ho, "Universal alignment probability and subset selection for ordinal optimization," *J. Optim. Theory Appl.*, vol. 93, no. 3, pp. 455–489, Jun. 1997.

[5] G. A. F. Seber and C. J. Wild, *Nonlinear Regression*. New York: Wiley, Sep. 2003.

[6] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[7] T. Chen, H. Chen, and R. W. Liu, "Approximation capability in $\mathrm{C}(\mathrm{R}^{-n})$ by multilayer feedforward networks and related problems," *IEEE Trans. Neural Netw.*, vol. 6, no. 1, pp. 25–30, Jan. 1995.

[8] J. G. Attali and G. Pagès, "Approximation of functions by a multilayer perceptron: A new approach," *Neural Netw.*, vol. 10, no. 6, pp. 1069–1081, Aug. 1997.

[9] C. G. Panayiotou, C. G. Cassandras, and W. B. Gong, "Model abstraction for discrete event systems using neural networks and sensitivity information," in *Proc. Winter Simul. Conf.*, Orlando, FL, Dec. 2000, vol. 1, pp. 335–341.

[10] R. A. Kilmer, A. E. Smith, and L. J. Shuman, "Computing confidence intervals for stochastic simulation using neural network metamodels," *Comput. Ind. Eng.*, vol. 36, no. 2, pp. 391–407, Apr. 1999.

[11] G. Lera and M. Pinzolas, "Neighborhood based Levenberg–Marquardt algorithm for neural network training," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1200–1203, Sep. 2002.

[12] M. F. Moller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Netw.*, vol. 6, no. 4, pp. 525–533, 1993.

[13] S. M. Sait and H. Youssef, *Iterative Computer Algorithms With Applications in Engineering: Solving Combinatorial Optimization Problems*. Los Alamitos, CA: IEEE Comput. Soc., Aug. 1999.

[14] E. K. P. Chong and S. H. Żak, *An Introduction to Optimization*, 2nd ed. New York: Wiley, 2001.

[15] C.-H. Chen, S. D. Wu, and L. Dai, "Ordinal comparison of heuristic algorithms using stochastic optimization," *IEEE Trans. Robot. Autom.*, vol. 15, no. 1, pp. 44–56, Nov. 1999.

A Polynomial Deadlock Avoidance Method for a Class of Nonsequential Resource Allocation Systems

Joaquín Ezpeleta and Rüdiger Valk

***Abstract*—This correspondence introduces a deadlock-avoidance algorithm for a class of manufacturing systems with the following characteristics: 1) Production orders are allowed to have assembly operations (which give the nonsequential nature to the resource allocation system model) but not disassembly operations, 2) the use of system resources must be conservative (resources are neither created nor destroyed), and 3) actions related to the granting of resources are controllable. The proposed solution represents a sufficient condition for a given system state to be safe and is based on an adaptation of the well-known Banker's approach for deadlock avoidance. The time complexity of the proposed solution is proved to be polynomial with the size of the Place/Transition net model.**

***Index Terms*—Assembly systems, Banker's algorithm, deadlock avoidance, Place/Transition nets (Petri nets), resource allocation system (RAS).**

I. Introduction

A resource allocation system (RAS) is a system composed of a set of processes which, in their execution, must compete for the set of system resources. The complexity of dealing with deadlocks strongly depends on the system structure. Depending on whether the involved processes have a concurrent or a sequential nature, whether runtime decisions for a process to choose a given path in its execution are allowed or not, and whether the model point of view considers that a process is allowed to use one or more system resources at a given moment, different classes of systems, whose specific characteristics allow the development of specific solutions, appear.

In the case of RAS, where the set of involved processes have a sequential nature, many different solutions that adopt different points of view (deadlock avoidance, prevention, and detection recovery) can be found. A good (noncomprehensive) list of solutions include [1]–[14].

Less attention has been paid to the case of assembly/disassembly systems. Adopting a Place/Transition net (Petri net) perspective, Roszkowska and Wojcik [15] propose a deadlock avoidance solution for a class of assembly systems where production orders are modeled by means of marked graphs and where one resource is allowed at each production step. From a supervisory control point of view, Fanti *et al.* [16] propose a solution for the same problem in the subclass of assembly systems, where processes have a tree structure and one resource is used at each processing step, based on an adaptation of some previous techniques applied to sequential RASs. Hsieh [17] concentrated on the same class of processes as that in [16] but allowed a more general way of resource usage (i.e., maintaining the constraint of resources to be granted/freed one unit at a time).

Xie and Jeng [10] adopt a Petri-net-based structural point of view and give a liveness characterization based on siphons for a class of systems where production orders are modeled by means of marking graphs and where a multiset of resources is allowed at each production

Manuscript received January 10, 2005; revised May 10, 2005. This work was supported by the German–Spanish Integrated Action under Project HA2000-0047/Deutscher Akademischer Austauschdienst (DAAD), the Projektbezogener Personenaustausch mit Spanien, and the CICYT-FEDER under Project TIC2001-1819. This paper was recommended by Associate Editor M. Jeng.

J. Ezpeleta is with the Department of Computer Science and Aragón Institute for Engineering Research (I3A), University of Zaragoza, 50018 Zaragoza, Spain (e-mail: ezpeleta@unizar.es).

R. Valk is with the Fachbereich Informatik, Universität Hamburg, 20146 Hamburg, Germany (e-mail: valk@informatik.uni-hamburg.de).

Digital Object Identifier 10.1109/TSMCA.2006.878963