# 行政院國家科學委員會專題研究計畫　期中進度報告

## 模擬最佳化演算法則的研擬與軟體製作(1/2)

計畫類別：個別型計畫
計畫編號：NSC93-2218-E-009-036-
執行期間：93 年 08 月 01 日至 94 年 07 月 31 日
執行單位：國立交通大學電機與控制工程研究所

計畫主持人：林心宇

計畫參與人員：洪士程，黃榮壽，陳亮元，林啟新

報告類型：精簡報告
報告附件：出席國際會議研究心得報告及發表論文
處理方式：本計畫可公開查詢

中　華　民　國 94 年 5 月 26 日

# 國科會研究計劃成果期中報告

## 計劃名稱：模擬最佳化演算法則的研擬與軟體製作(1/2)

## 計劃執行起迄：2004.08.01 至 2005.07.31


## 計劃主持人：林心宇



## 計劃編號：NSC93-2218-E-009-036-



## 執行機關：國立交通大學電機與控制工程研究所



**計畫中文摘要**

模擬最佳化(Simulation optimization)是最佳化方法領域中最新的發展，模擬最佳化問題主要是指對任一個 input variable setting 的 objective function 的 evaluation 都須利用模擬(simulation)的方法來求得，所以我們無法以傳統最佳化方法解決此類型的的問題。此類問題涵蓋的範圍甚廣，如一些具有廣大輸入變數空間(huge input-variable)的隨機模擬最佳化問題(stochastic simulation optimization problem)，大型系統中具有決策變數與 discrete variable 的最佳化問題等等。

本計劃擬針對一些模擬最佳化問題加以歸類，並針對所歸納出的類型問題提出模擬最佳化的演算法則，基本上在模擬最佳化問題中，其結構資訊(structural information)很難用解析的方法萃取(extract)出來，所以我們所提出的模擬最佳化方法即以模擬(simulation)做為萃取系統的結構資訊的手段，然後以所萃取的結構資訊作為縮小搜尋範圍的依據。如此迭代進行，將可得到一個不錯的解。

我們將以序的最佳化理論(ordinal optimization theory)來證明我們所得的解是不錯的解。

除此之外我們將以基因演算法(genetic algorithm)，模擬退火法(simulated annealing)，以及塔布搜尋法(tabu search)等常被用來解模擬最佳化問題的方法來解我們所提出的類型的問題並比較所得的結果，以及所花費的計算時間及實用性。

演算法研擬擬完成且經完善的電腦模擬驗證後，我們將製作成軟體以供應廣大的用途。

*Abstract*

Simulation optimization is one of the most frontier research area in optimization. The main characteristics of simulation optimization problem is the evaluation of the objective function of an input-variable setting requires lengthy simulation. Therefore we cannot use the conventional optimization techniques to solve them. There are various types of simulation optimization problems such as stochastic optimization problems with huge input-variable space and large scale optimization problems with decision and discrete control variables.

In this project, we intend to categorize some classes of simulation optimization problems and propose algorithms to solve

them. Basically, in the simulation optimization problem, it is almost impossible to extract structural information of the system analytically. Therefore, the proposed simulation optimization algorithms will use simulations as a tool to extract the structural information. The extracted structural information will be used in the proposed algorithm to reduce the searching space. Such an iterative simulation optimization technique will use only reasonable computation time to obtain a good enough solution.

We will use ordinal optimization theory to prove the quality of the solution we obtain. In addition, we will compare our results with those obtained by the competing methods such as the genetic algorithms, simulated annealing, and the tabu search methods.

We will implement our algorithms in the form of commercial software for a more general purpose.

## 一、前言

在1992年時，Professor Azadivar 在[1]中給模擬最佳化問題(simulation optimization problem)下了一個簡單的定義：對任一個 input(variable) 的 objective function 的 evaluation 都需利用模擬(simulation)才能得知其 objective value 的最佳化問題即是模擬最佳化問題。此類型問題涵蓋範圍甚廣，如在隨機模擬最佳化問題(stochastic simulation optimization)中需要以隨機模擬(stochastic simulation)來計算一個 input variable 值 的 objective value ，由於每一個隨機模擬都需相當長的計算時間，所以如果輸入變數空間(input-variable)很龐大的話，那麼要得到最佳化解所需要的計算量實在是無比的冗長。雖然，模擬最佳化問題本身不具備自然的 structural information，因此有些計算其 gradient 的複雜方法，如 IPA(Infinitesimal purtaation analysis) LR(Likelyhoud ratio)等，對特殊的製造系統而言有其各別的貢獻，而一般最常用的這是 heuristic methods 如 genetic algorithm，simulated annealing method 及 tabu search method 等，然而我們這個計畫想嘗試一種新的 search technique。

## 二、研究目的

在工業界中，尤其是製程複雜的半導體產業，充斥著不同形式的最佳化問題，由於其製程中充滿不確定因素，這些最佳化問題基本上都是具有 stochastic simulation 特性的最佳化問題，所以它們是屬於模擬最佳化問題的類型。又如大規模電力系統的最佳化問題中如最佳電力潮流問題(optimal power flow)，最佳電容裝置問題(optimal capacitor placement)等，皆具有 discrete control variables (如 switchable capacitors，transformer taps)，所以這些最佳化問題是 mixed integer-discrete continuous nonlinear programming 的問題，所以也屬於模擬最佳化類型的問題。因此，本研究的目的在於解決產業中的重要問題。

## 三、文獻探討

在國內外對個別的問題皆有相當多的人在研究，例如在解含 discrete control variables 的最佳電力潮流問題中就有多組國外研究群分別以 genetic algorithm [11]，simulated annealing [3]，tabu search [12]等方法求解。而在半導體製程中的排程問題亦屬於模擬最佳化問題，在國外有聲舉卓著的伊利偌大學 Kumar 教授及其領導的研究群在從事這方面的研究。關於模擬最佳化方法，由於此領域尚屬萌芽階段，但其重要性已引起學術界及工業界的重視，在美國已有數個大學展開此方面的研究如 Prof. Azadivar[1]，Professor Carson[2]，Professor Fu [4]，Professor Johnson[6]。Professor Meketon[13]，Professor Paul [14]，

Professor Stuckman[15]，Professor Tompkins[16]等。

## 四、研究方法

A. 第一類問題：

我們所考慮的第一類問題是隨機模擬最佳化(stochastic simulation optimization)的問題，其問題的型態可敘述如下：

$$\min \ f(x)$$
$$\text{subject to} \quad x \in X$$

我們所考慮的此類型問題具下列兩項特性：

(i) Input-variable space $X$ 會 exponentially grows with respect to problem size。

(ii) 若給予一個 input variable $x$ 的值，則 $f(x)$ 的計算需相當冗長的 stochastic simulation。
所以若以 global searching techniques 來解此類問題將花費非常龐大的計算時間。

為克服計算 $f(x)$ 需冗長的計算時間的缺點，我們的模擬最佳化解法的構想如下：

(a) 首先以模擬為手段，均勻地由 $X$ 中挑選適當數量的 $x$ 值，然後以大約的 stochastic simulation 來求出相對應的 $f(x)$。

(b) 以所求出之相當數量的 $(x, f(x))$ pairs 來建立約化系統 I/O 的類神經網路模式，於是這個可以快速求取 $f(x)$ 的類神經網路模式便成為我們在第一階段的 optimization 的工具。

(c)在第一階段的 optimization scheme 中，我們將以 global searching technique 以及(b)中所建立的類神經網路 I/O 模式來找出一個數量不多但具有大約最佳目標函數的 input-variable 的子集解。

(d)在第二階段的 optimization scheme 中，我們將針對(c)中求出的不錯的子集解的每一個 input variable 解作較短的 stochastic simulation，並選取具有大約最佳目標函數的更小子集。

(e)於是在最後一階段的 optimization scheme 中，我們將針對(d)中所選出之更小的子集中的每一個 $x$，做一精準的 stochastic simulation，於是具有最佳的目標函數的 input variable $x$ 值即為我們所要求的不錯的解。

我們在這一年的計畫中已成功地找到第一類問題的應用例子即如何在晶圓測試程序中減少晶粒誤宰及重測的問題。

B. 第二類問題：

我們所考慮的第二類型問題是 mixed integer-discrete continuous nonlinear programming problem，此類型問題可 formulate 成如下形式：

$$\min \ f(x,d,k)$$
$$g(x,d,k)=0$$
$$h(x)\le 0$$
$$q(d,k)\le 0$$
$$d\in D,\ k\in K,\ x\in X$$

其中 $g$ 為等式限制式，$h$ 為 continuous variable $x$ 的不等式限制式，$q$ 為 integer variables $k$ 及 discrete variables $d$ 的不等式限制式。$D$，$K$ 及 $X$ 各為 $d$，$k$ 及 $x$ 的定義域。

我們已初步找到第二類問題的一個很好的應用例子即分散式的具有離散控制變數的最佳電力潮流問題。此問題的困難處在於它不僅僅是模擬最佳化所涵蓋的範圍，同時它也必須要用分散式處理

(distributed processing)的計算方式來解才可得到不錯的解。

## 五、結果與討論

雖然此計畫僅進行到一半，我們卻已有相當豐碩的成果。首先，我們已將晶圓測試程序中減少晶粒誤宰及重測的問題formulate 成一個模擬最佳化問題，並依照所提出的構想研擬出一個 two-level ordinal optimization 的 algorithm 並成功地解決了此問題。此成果所撰寫成的論文已被 IEEE Trans. on Systems，Man and Cydernetico Part A 期刊 accept，如附件一所示。同時，我們在第二類問題所擬解的具離散控制變數的分散式最佳電力潮流問題上已初步研擬出一個連續變數的分散式最佳電力潮流演算法。此成果已撰寫成會議論文(如附件二所示)並將於今年(2005)6 月 15 日至 17 日在 EuroPES 2005 於西班牙召開的會議中發表。同時，我們也在這兩篇文章中註明此研究成果係本國科會計畫所贊助，並將計畫編號明列其上

## 六、參考文獻


[1] Azadivar, F.(1992) Tutorial on Simulation Optimization, In *Proceedings of the 1992 Winter Simulation Conference*, Arlington, ed. J. Swain, D. Glodsman, R. C. Crain, and J. R. Wilson, Institute of Electrical and Electronics Engineers, Piscataway, New Jersey, 198-204.

[2] Carson, Y. Maria, A. (1997) Simulation Optimisation Methods And Applications, in *Proceedings of the 1997 Winter Sumulation Conference*, Georgia, ed. S. Andradottir, K. J. Healy, D. H. Withers, B. L. Nelson, Institute of Electrical and Electronics Engineers, Piscatatway, New Jersey, 118-126.

[3] Chen,L., Suzuki, H. and katou, K., "Mean field theory for optimal power flow,"*IEEE Trans*. on *Power System*, vol. 12, no. 4, Nov. 1997.

[4] Fu, M. C. (1994) A Tutorial Review of Techniques for Simulation Optimisation, **In Proceedings of the 1994 Winter Simulation Conference**, Lake Buena Vista, ed. Jeffey D. Tew, S. Manivannan, D. A Sadowski, and A. F. Seila, Institute of Electrical and Electronics Engineers, Piscataway, New Jersey, 149-156.

[5] Ho, Y.C., Cassandras, C.C., Chen, C.H., and Dai, L., "Ordinal optimization and simulation," *Journal of Operation Research Society*, vol. 21, pp. 490-500, 2000.

[6] Johnson, D. S., Aragon, C. R., McGeoch, L. A., and Schevon, C. (1989) Optimization by Simulated Annealing: An Experimental Evaluation; part I, Graph Partitioning, *Operations Research*, 37(6): 865-892.

[7] Lau, T.W.E. and Ho, Y.C., "Universal alignment probability and subset selection for ordinal optimization," *JOTA*, vol. 39, no. 3, June 1997.

[8] Lee, Y. H. and Iawate, K. (1991) Part Ordering through Simulation-Optimization in an FMS. *International Journal of Production Research*, 29(7): 1309-1323

[9] Lin, C. and Lin, S., "A new

dual-type method used in solving optimal power flow problems." *IEEE Trans. on Power Systems*, vol.12, pp.1667-1675, Nov. 1997.

[10] Lin, S. and Ho, Y.C., "Universal alignment probability revisited," *JOTA*, pp.299-407, May 2002.

[11] Ma, J. T. and Lai, L. L., "Evolutionary programming approach to reactive power planning,"*IEEE Proc.*, *Generation, Transmission and Distribution*, vol. 143, no.4, 1996.

[12] Mantawy, a., Abdel-Magid, Y., and Selim, S., "Unitcommitment by tabu search," *IEEE Proc., Generation, Transmission and distribution*, vol 145, no. 1, pp. 56-64, 1998.

[13] Meketon, M. S. (1987) Optimization in Simulation: A Survey of Recent Results, In *Proceedings of the 1987 Winter Simulation Conference*, ed. A. Thesen, H. Grant, and W. D. Kelton, Institute of Electrical and Electronics Engineers, Piscataway, New Jersey, 58-67.

[14] Paul, R. J. and Chanev, T. S. (1998) Simulation Optimization Using a Genetic Algorithm, *Simulation Practice and Theory*, 6(6):601-611

[15] Stuckman, B., Evans, G., and Mollaghasemi, M. (1991) Comparison of Global Search Methods for Design Optimization using Simulation, In *Proceedings of the 1991 Winter Simulation Conference*, ed. B. L. Nelson, W. D. Kelton, and G. M. Clark, Institute of Electrical and Electronics Engineers, Piscataway, New jersey, 937-943.

[16] Tompkins, G. and Azadiver, F. (1995) Genetic Algorithms in Optimizing Simulated Systems, In *Proceedings of the 1995 Winter Sumulation Conference*, ed. C. Alexopolous, K. Kang, W. R. Lilegdon, and D. Goldsman, Institute of Electrical and Electronics Engineers, Piscataway, New Jersey, 757-762.

## 七、計劃成果自評

本計劃已進行將近一年，所獲成果堪稱豐碩，總計已發表了一篇知名國際期刊論文，以及一篇國際會議論文，同時，我們也在這兩篇文章中註明此研究成果係本國科會計畫所贊助，並將計畫編號明列其上。我們期望明年此時，能有更豐富的成果報告。

附件一

# Application of an Ordinal Optimization Algorithm to the Wafer Testing Process[1]

***Shin-Yeu Lin*** *and* ***Shih-Cheng Horng***

sylin@cc.nctu.edu.tw schong.ece90g@nctu.edu.tw

Department of Electrical and Control Engineering

National Chiao Tung University

Hsinchu 300, Taiwan, R.O.C.

Accepted by the IEEE Transactions on Systems, Man and Cybernetics, Part A

Correspondent : Professor Shin-Yeu Lin

Department of Electrical and Control Engineering

National Chiao Tung University

1001 Ta Hsueh Road, Hsinchu 300

Taiwan, R.O.C.

e-mail : sylin@cc.nctu.edu.tw

---

[1] Manuscript received September 30, 2004; revised April 25, 2005. This work was supported in part by the National Science Council in Taiwan, R.O.C., under grant NSC93-2218-E-009-036.

Shin-Yeu Lin is 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).

Shih-Cheng Horng is with the Department of Electrical and Control Engineering, National Chiao Tung University, Hsinchu 300, Taiwan, R.O.C.

*Abstract*

In this paper, 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 (OO) 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 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 minutes in total in a Pentium IV PC to obtain the good enough threshold values.

*Index Terms*— wafer probing, overkill, retest, ordinal optimization, stochastic optimization, neural network, genetic algorithm.

## 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-inch wafer may consist of 600 to 15000 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 process, this decision is made based on whether the number of good dies and the number of *bins*[2] 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 paper 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 3, this makes the problem becomes a hard optimization problem. Thus, to cope with the enormous computational complexity, we propose an ordinal optimization 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

---

[2] 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.

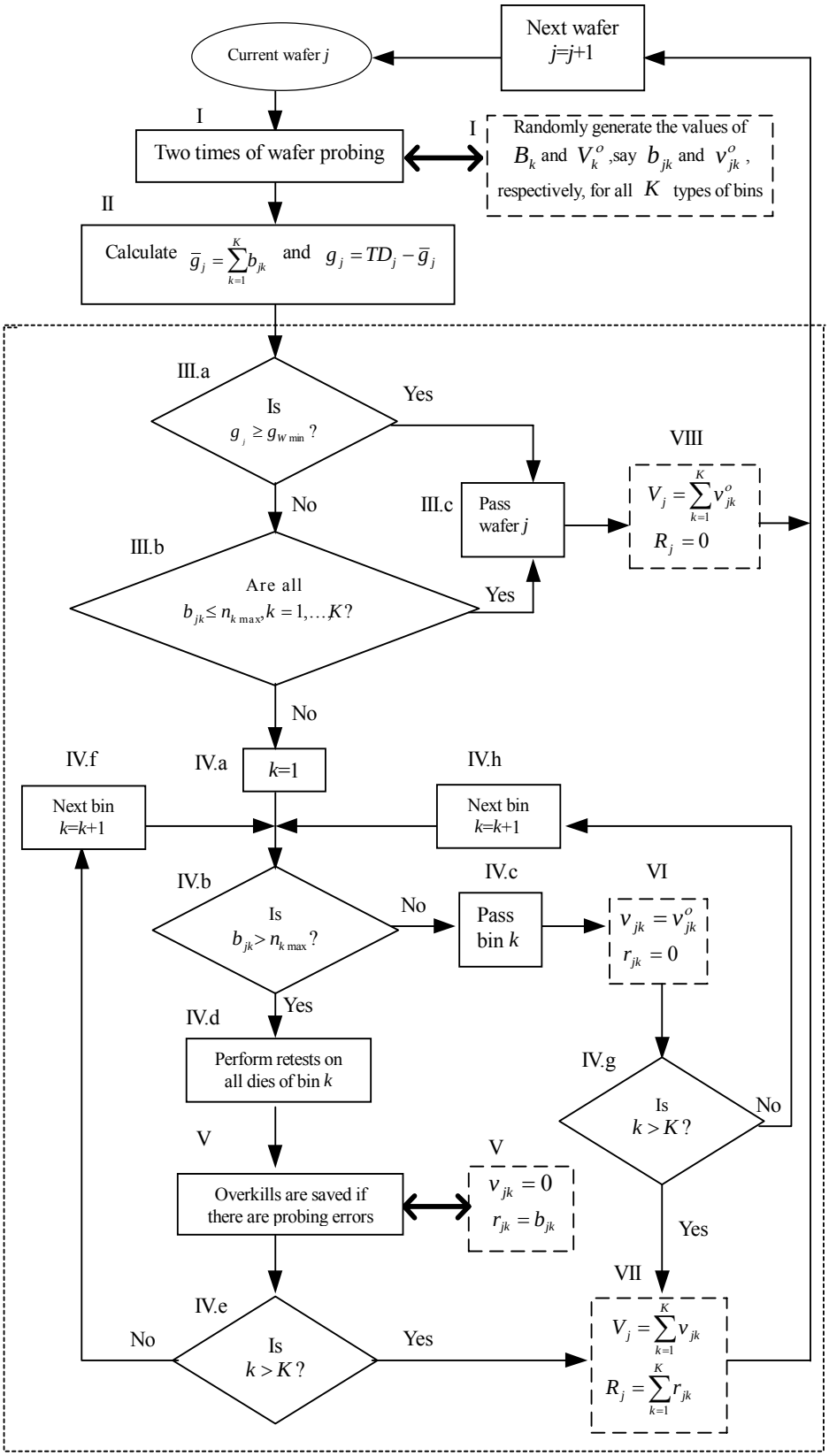


Fig. 1: Flow chart of the real and simulated wafer testing procedures.

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 $g_j = TD_j - \overline{g}_j$ 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 $n_{k\max}$, $k = 1,...,K$, 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 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*

*Simulation model for the two-times wafer probing*

Since we cannot perform the real wafer probing in 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 $E[V] = \frac{1}{L}\sum_{j=1}^{L} V_j$ and $E[R] = \frac{1}{L}\sum_{j=1}^{L} R_j$, 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 $n_{k\max}$, that 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:

$$\min_{x \in X} \; E[V]$$

subject to {simulated wafer testing procedures in Fig. 1},

$$E[R] \le r_T, \tag{2}$$

where $x \equiv [g_{W\min}, n_{k\max}, k = 1,...,K]$ denotes the vector of threshold values, that 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 (2) is a soft-constraint in a sense, we can use a penalty function to relax that constraint and transform (2) into the following unconstrained stochastic optimization problem:

$$\min_{x \in X} \; E[V] + P \times (E[R] - r_T)$$

subject to $\{$simulated wafer testing procedures in Fig. 1$\}$, (3)

where $P$ denotes a continuous penalty function for the constraint $E[R] \le r_T$.

## III The Two-Level Ordinal Optimization Algorithm

The size of the decision variable space $X$ in (3) is huge; for example, for an 8-inch 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 (3) will be very computationally expensive. To cope with the computational complexity of this problem, we propose an Ordinal Optimization (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]: (i) Uniformly or randomly select $N$, say 1000, vectors of decision variables from $X$. (ii) 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. (iii) Evaluate and order all the $s$ vectors obtained from (ii) using the exact model, then pick the top $k\,(\geq 1)$ 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 (ii), and the best vector of decision variables obtained in (iii) 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 (i). Combining first level approach with the existing searching procedures (ii) and (iii) forms a two-level OO algorithm.

*A. Constructing a Metamodel for (3)*

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. ANN not only approximate the continuous functions well [7,8], but also being 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 simple ANN with 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 hyperbolic tangent sigmoid function in the first layer, and 1 neuron with linear function in the second layer. We obtain the set of training data for the two ANNs by the following two steps. (a) Narrow down the decision-variable space $X$ by excluding the irrational threshold values and denote the reduced decision variable space by $\hat{X}$ [3]. (b) 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 (3) 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 (3). 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

[3] 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.

algorithms are when any of the following two conditions occurs: (i) the sum of the mean squared errors is smaller than $10^{-5}$, and (ii) the 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 (3). This forms our metamodel to estimate the objective value of (3) 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 Algorithm (GA), 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, Chapter 14] as our searching tool.

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 (3) 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. 70% 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 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 (i).

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

Starting from the $N$ vectors of decision variables obtained in Section 3.2, we will proceed with (ii) of the OO searching procedure to compute the objective value of (3) 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, that 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 (3) and choose the top $s$ vectors which form the estimated good enough subset.

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

We will compute the objective value of (3) 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 replacing $L_s$ by $L_e$ ($>> L_s$) wafers. Then the vector associated with the smallest objective value of (3) 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-inch 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,…,12, and the probability of the number of overkills in bin $k$, $p_k$, $k$=1,…,12, are given. The yield rate of this product is 68%. The decision-variable space $X = \{x(=[g_{W\min}, n_{k\max}, k = 1,...,K]) \mid g_{W\min} \in [1,203], n_{k\max} \in [1,203], k = 1,...,12\}$. We used the sigmoid-type function as our penalty function $P$ in (3), i.e., $P = \eta \frac{1}{1+e^{-(E[R]-r_T)}}$, where $\eta\,(\cong 0.1594)$ is a normalized coefficient such that $\eta = \frac{\max\limits_{i\in\{1,...,M\}} E[V_i]}{\max\limits_{i\in\{1,...,M\}} E[R_i]}$.

We set $\hat{X} = \{x(=[g_{W\min}, n_{k\max}, k = 1,..., K]) \mid g_{W\min} \in [50,203], n_{k\max} \in [1,6\mu_k], k = 1,...,12\}$, where $\mu_k$ 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 3 cases of different $r_T$'s, 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 1. The CPU time consumed in each case plus the training time is approximately 6.05 minutes. From Table 1, 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 1.

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 1, we use 521 real test wafers, whose number of dies of all bins, $b_{jk}$, $j = 1,...,521$, $k = 1,...,12$, and overkills before retest, $v_{jk}^{o}$, $j = 1,...,521$, $k = 1,...,12$, are known. The corresponding results of the pair of the average overkills per wafer, $E[V](=\frac{1}{521}\sum_{j=1}^{521}V_j)$, and the average retests per wafer, $E[R](=\frac{1}{521}\sum_{j=1}^{521}R_j)$, for these 521 test wafers are shown in Fig. 2 as the points marked by "☆", "*", "○" with the corresponding $r_T$ shown on

Table 1: The 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 |
| * $\frac{E[V]}{TD}\times 100\%$ | 1.86% | 1.07% | 0.27% |

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

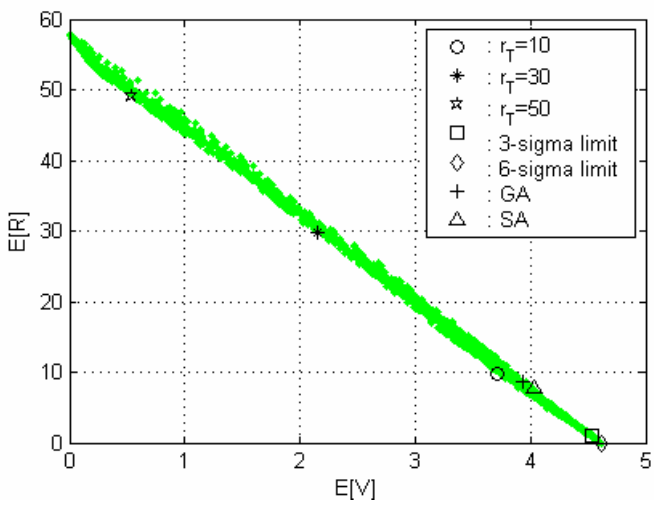


Fig. 2: The resulted $(E[V], E[R])$ pairs of the 521 test wafers based on the vector of threshold values determined by two-level algorithm, random generator, three-sigma limit, six-sigma limit, GA and SA algorithm.

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 "•" 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 (3) 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 six-sigma limit to determine the threshold values such that $g_{W\min}^{3\sigma} = \mu_g - 3\sigma_g$, $n_{k\max}^{3\sigma} = \mu_k + 3\sigma_k$, $k = 1,...,12$, and $g_{W\min}^{6\sigma} = \mu_g - 6\sigma_g$, $n_{k\max}^{6\sigma} = \mu_k + 6\sigma_k$, $k = 1,...,12$, where $\mu_g$ and $\sigma_g$, the mean and standard derivation of the

number of good dies in a wafer, and $\mu_k$ and $\sigma_k$, 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 three-sigma limit and six-sigma limit are also shown in Fig. 2 marked by "□" and "◇", 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 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 three-sigma limit and six-sigma limit do not generate the pareto optimal solution for (3), and they cannot control the level of retests like ours. We have also used typical GA and Simulated Annealing (SA) [13] algorithm to solve (3) for the case of $r_T$=10. As indicated at the beginning of Section 3, the global searching techniques are computationally expensive in solving (3). 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 (3) 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 $E[R] \le 10$, respectively. In addition, both GA and SA do not generate the pareto optimal solution, because the best so far solution they obtained for 5 hours of CPU time are still far away from the optimal solution of (3).

## V. Conclusions

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 ordinal optimization theory based two-level algorithm is not limited to the problem considered in this paper. 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 Transactions on Semiconductor Manufacturing*, 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," *2001 IEEE/SEMI Advanced Semiconductor Manufacturing Conference and Workshop*, Munich, Germany, pp. 187-192, April 2001.

[3] D. C. Montgomery, *Introduction to statistical quality control.* 5th edition, New York: John Wiley and Sons, July 2004.

[4] T.W.E. Lau and Y.C. Ho, "Universal alignment probability and subset selection for ordinal optimization," *Journal of Optimization Theory and Applications*, vol. 93, no. 3, pp. 455-489, June 1997.

[5] George A. F. Seber and C. J. Wild, *Nonlinear Regression.* New York: John Wiley and Sons, Sept. 2003.

[6] K. Hornik, M. Stinchcombe and H. White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, vol. 2, no. 5, pp. 359-366, 1989.

[7] T. Chen, H. Chen and R. W. Liu, "Approximation capability in $C(R^{-n})$ by multilayer feedforward networks and related problems," *IEEE Transactions on Neural Networks*, 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 Networks*, vol.10, no.6, pp. 1069-1081, August 1997.

[9] C. G. Panayiotou, C. G. Cassandras, W. B. Gong, "Model abstraction for discrete event systems using neural networks and sensitivity information," *Proceedings of the 2000 Winter Simulation Conference,* Orlando, FL, USA, vol. 1, pp. 335-341, Dec. 2000.

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

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

[12] M. F. Moller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, 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 :IEEE Computer Society, Aug. 1999.

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

Sons, 2001.

[15] C.-H. Chen, S.D. Wu, and L. Dai, “Ordinal comparison of heuristic algorithms using stochastic optimization,” *IEEE Transactions on Robotics and Automation,* vol. 15, no. 1, pp. 44–56, Nov. 1999.

附件二: To be presented at EuroPES 2005

# A DISTRIBUTED OPTIMAL POWER FLOW ALGORITHM

Shin-Yeu Lin
Dept. of Elec. and Contr. Eng.
National Chiao Tung Univ.
Hsinchu 300
Taiwan, ROC
e-mail:sylin@cc.nctu.edu.tw

Ch'i-Hsin Lin
Dept. of Electronics Eng.
Kao-Yuan Inst. of Tech.
Kaoshiung, 700
Taiwan, ROC
e-mail:chsinlin@cc.kyit.edu.tw

## ABSTRACT

In this paper, we propose a distributed computational algorithm for solving the distributed optimal power flow problem under deregulated environment. Except for the data of the self-subsystem, the proposed distributed algorithm requires only the data of the boundary buses of the connecting subsystems. Therefore, it is easily implemented in a computer network. We have tested the proposed distributed algorithm on the IEEE 118-bus system, which is arbitrarily partitioned into four subsystems. The test results demonstrate the convergence and the computational efficiency of the proposed distributed algorithm.

**KEY WORDS**

Distributed optimal power flow, distributed computation, nonlinear programming, power deregulation.

## 1. Introduction

Power deregulation is a trend in recent years and pushes the power market to be very active. Various research issues are then raised under the deregulated environment; among them, Distributed Optimal Power Flow (DOPF) is one of the most important subjects.

Although the Optimal Power Flow (OPF) problem has a long history in power system research [1-5], the study of DOPF is introduced only recently. Kim and Baldick proposed a course-grained DOPF algorithm in [6], and they also compared three decomposition coordination methods for implementing DOPF algorithm in [7]. Hur et al. had evaluated the convergence rate of the auxiliary problem principle for DOPF algorithm in [8]. In a more recent paper [9], Hur et al. consider the security constraints for DOPF.

In this paper, we consider the DOPF in deregulated environment such that each subsystem will buy (sell) power from (to) the neighboring subsystems if they fall short of (have surplus) power. Since the power flow from the neighboring subsystem is set by the contract, each subsystem would utilize the generated and purchased power in an optimal way. However, to achieve a system-wise optimality, all the subsystems have to cooperate to solve the DOPF in a whole.

From the hardware viewpoint, DOPF is easier to implement nowadays, because the mature computer network technology and the prevailed network infrastructure. Thus, what is most needed is a reliable distributed computation software. Despite the existence of the DOPF algorithms mentioned above, the DOPF problem considered here is different from those in [6-9], because the tie line flows should be equality constraints in our formulation. Therefore, we need to propose a DOPF algorithm that is suitable for the environment of deregulated power systems.

The paper is organized in the following manner. In Section II, we will state the considered DOPF mathematically. In Section III, we will propose a solution method to solve the considered DOPF. In Section IV, we will present distributed algorithmic steps to execute the proposed solution method in the computer network. In Section V, we will test our distributed algorithm on the IEEE 118-bus system, which is arbitrarily partitioned into 4 subsystems. Finally, we make a conclusion in Section VI.

## 2. Problem Statement

Since the amount of power delivered from the selling subsystem is fixed and set by the contract. Thus, the power flow of the tie lines from the selling subsystem to the buying subsystem is considered to be the additional line flow constraints in the DOPF. Thus, our DOPF can be stated in the following:

$$\min \sum_{i=1}^{N} c_i(P_{G_i})$$

subject to

$$g_i(\overset{0}{x}_i, x_{i,b}, P_{G_i}, Q_{G_i}, x_{i,e}) = 0$$

$$h(\overset{0}{x}_i, x_{i,b}, P_{G_i}, Q_{G_i}) \le 0, i = 1,..., N$$

$$P_{ij}(x_{i,b}, x_{i,e_j}) = p_{ij}$$

$$Q_{ij}(x_{i,b}, x_{i,e_j}) = q_{ij} \quad \forall (i, j) \in M \qquad (1)$$

where $c_i(P_{G_i})$ denote the total generation cost function of subsystem $i$, $x_i = (\overset{0}{x}_i, x_{i,b})$ denote the vector of complex voltage of subsystem $i$, where $\overset{0}{x}_i$, and $x_{i,b}$ denote the vector of complex voltage of interior buses and boundary buses, respectively; $g(\overset{0}{x}_i, x_{i,b}, P_{G_i}, Q_{G_i}, x_{i,e}) = 0$ denotes the flow balance equations and $h(\overset{0}{x}_i, x_{i,b}, P_{G_i}, Q_{G_i}) \le 0$ denotes the security constraints such as the thermal limit, voltage magnitude limit, power generate limit of subsystem $i$; $x_{i,e}$ denotes the vector of complex voltage of the boundary buses of other subsystems connecting with subsystem $i$; $N$ denotes the total number of subsystems in the deregulated power network, and $P_{ij}(x_{i,b}, x_{i,e_j}) = p_{ij}$ denotes the real power line flow equation from subsystem $i$ to $j$, where $p_{ij}$ denote the amount of power purchased by subsystem $j$ from $i$; $Q_{ij}(x_{i,b}, x_{i,e_j}) = q_{ij}$, denote the corresponding reactive power flow of the purchased real power $p_{ij}$ with reasonal power factor; $x_{i,e_j}$ denotes the vector of complex voltage of the boundary bus of subsystem $j$ such that $(i, j) \in M$; $M = \{(i, j)'s\}$ denotes the set of selling and purchasing pairs $(i, j)$ of power companies $i$ and $j$.

We assume each subsystem has its own control center, which is equipped with a computer. The computers of all subsystems are connected through a computer network. The proposed distributed algorithm for solving the DOPF (1) will be carried out in this network.

## 3. The Solution Method

In order to achieve the system-wise optimal objective, we have to solve the DOPF problem shown in (1) in a whole using a method that can be computed distributedly. In our previous research work, we have developed the DPPQN method based algorithm to solve the centralized OPF problems [10-12] for quite a while. This algorithm gains its computational efficiency and stability by its decomposition effects and the capability of handing the binding inequality constraints. As a matter of fact, the decomposition effects of the DPPQN method can play the key for distributed computation. Thus, we will describe how the DPPQN method solves the DOPF (1) in the following:

First of all, we define the vector functions $\overline{g}_i = (g_i, P_{ij}, Q_{ij})$ and partition $\overline{g}_i$ into $(\overset{0}{\overline{g}}_i, \overline{g}_{i,b})$ such that the sub-vector function $\overset{0}{\overline{g}}_i$ involves only the variable vector $(\overset{0}{x}_i, x_{i,b}, P_{G_i}, Q_{G_i})$ of subsystem $i$, while $\overline{g}_{i,b}$ involves not only the variable vector of subsystem $i$ but also the variable vector $x_{i,e}$ of the subsystems connecting with subsystem $i$. Using this notation, we can rewrite (1) as

$$\min \sum_{i=1}^{N} c_i(P_{G_i})$$

subject to

$$\overset{0}{\overline{g}}_i(\overset{0}{x}_i, x_{i,b}, P_{G_i}, Q_{G_i}) = 0$$

$$\overline{g}_{i,b}(\overset{0}{x}_i, x_{i,b}, P_{G_i}, Q_{G_i}, x_{i,e}) = 0$$

$$h_i(\overset{0}{x}_i, x_{i,b}, P_{G_i}, Q_{G_i}) \le 0, i = 1,..., N \qquad (2)$$

For the sake of notational simplicity, we define the following variable vector $y = (y_1,..., y_N)$ where $y_i = (\overset{0}{y}_i, x_{i,b})$ and $\overset{0}{y}_i = (\overset{0}{x}_i, P_{G_i}, Q_{G_i})$.

The proposed DPPQN method based algorithm is a combination of the Successive Quadratic Programming (SQP) method with the Dual Projected Pseudo Quasi Newton (DPPQN) method such that the Quadratic Programming Problem (QPP) induced in the SQP method is solved by the DPPQN method. The SQP method uses the following iterations to solve (2)

$$y(k+1) = y(k) + \alpha(k)\Delta y(k) \qquad (3)$$

where $k$ is the iteration index, $\alpha(k)$ is a positive step-size and $\Delta y(k) = (\Delta y_1(k),..., \Delta y_N(k))$, in which $\Delta y_i(k) = (\Delta \overset{0}{y}_i(k), \Delta x_{i,b}(k))$ and

$\Delta \overset{0}{y}(k) = (\Delta \overset{0}{x}_i(k), \Delta P_{G_i}(k), \Delta Q_{G_i}(k))$. $\Delta y(k)$ is the solution of the following.

QPP:

$$\min_{\Delta y} \sum_{i=1}^{N} \Delta P_{G_i}^T \frac{\partial^2 c_i(k)}{\partial P_{G_i}^2} \Delta P_{G_i} + \frac{\partial c_i(k)}{\partial P_{G_i}} \Delta P_{G_i}$$

subject to

$$\overset{0}{\overline{g}}_i(k) + \frac{\partial \overset{0}{\overline{g}}_i(k)}{\partial \overset{0}{y}_i} \Delta \overset{0}{y}_i + \frac{\partial \overset{0}{\overline{g}}_i(k)}{\partial x_{i,b}} \Delta x_{i,b} = 0$$

$$\overline{g}_{i,b}(k) + \frac{\partial \overline{g}_{i,b}(k)}{\partial \overset{0}{y}_i} \Delta \overset{0}{y}_i + \frac{\partial \overline{g}_{i,b}(k)}{\partial x_{i,b}} \Delta x_{i,b} + \frac{\partial \overline{g}_{i,b}(k)}{\partial x_{i,e}} \Delta x_{i,e} = 0$$

$$h_i(k) + \frac{\partial h_i(k)}{\partial \overset{0}{y}_i} \Delta \overset{0}{y}_i \frac{\partial h_i(k)}{\partial x_{i,b}} \Delta x_{i,b} \le 0$$

$$i = 1, \ldots, N \qquad (4)$$

where $\overset{0}{\overline{g}}_i(k)$ represents $\overset{0}{\overline{g}}_i(\overset{0}{y}(k), x_{i,b}(k))$ and the same abbreviation applies to $\overline{g}_{i,b}(k)$ and $h_i(k)$. The QPP in (4) will be solved by the DPPQN method. Thus, most of the computations of the proposed algorithm lie on the DPPQN method to solve (4) for $\Delta y(k)$, because the SQP method simply update $y(k)$ by (3) once $\Delta y(k)$ is obtained in each iteration.

Instead of solving (4) directly, the DPPQN solves the dual problem of (4), which is stated below

$$\max \ \phi(\lambda) \qquad (5)$$

where the dual function

$\phi(\lambda) =$

$$\min_{\Delta y \in \Omega(k)} \sum_{i=1}^{N} \left[ \Delta P_{G_i}^T \frac{\partial^2 c_i(k)}{\partial P_{G_i}^2} \Delta P_{G_i} + \frac{\partial c_i(k)}{\partial \Delta P_{G_i}} \Delta P_{G_i} \right]$$

$$+ \sum_{i=1}^{N} \overset{0}{\lambda}_i^T \left[ \overset{0}{\overline{g}}_i(k) + \frac{\partial \overset{0}{\overline{g}}_i(k)}{\partial \overset{0}{y}_i} \Delta \overset{0}{y}_i + \frac{\partial \overset{0}{\overline{g}}_i(k)}{\partial x_{i,b}} \Delta x_{i,b} \right]$$

$$+ \sum_{i=1}^{N} \lambda_{i,b}^T \left[ \overline{g}_{i,b}(k) + \frac{\partial \overline{g}_{i,b}(k)}{\partial \overset{0}{y}_i} \Delta \overset{0}{y}_i + \frac{\partial \overline{g}_{i,b}(k)}{\partial x_{i,b}} \Delta x_{i,b} + \frac{\partial \overline{g}_{i,b}(k)}{\partial x_{i,e}} \Delta x_{i,e} \right] \qquad (6)$$

in which the constraint set

$$\Omega(k) = \left\{ \Delta y \left| h_i(k) + \frac{\partial h_i(k)}{\partial \overset{0}{y}_i} \Delta y_i + \frac{\partial h_i(k)}{\partial x_{i,b}} \Delta x_{i,b} \le 0 \right. \right.$$

$$\left. i = 1, \ldots, N \right\} \qquad (7)$$

The DPPQN method uses the following iterations to solve the dual problem (5)

$$\lambda(t+1) = \lambda(t) + \beta(t) \Delta \lambda(t) \qquad (8)$$

where $t$ is the iteration index; $\beta(t)$ is a positive step-size; $\lambda(t) = (\lambda_1(t), \ldots, \lambda_N(t))$ in which $\lambda_i(t) = (\overset{0}{\lambda}_i(t), \lambda_{i,b}(t))$, where $\overset{0}{\lambda}_i(t)$ corresponds to the equality constraints related with subsystem $i$ only, while $\lambda_{i,b}(t)$ corresponds the equality constraints involving $x_{i,e}$; $\Delta \lambda(t) = (\Delta \lambda_1(t), \ldots, \Delta \lambda_N(t))$ denotes the increment of $\lambda(t)$, where $\Delta \lambda_i(t) = (\Delta \overset{0}{\lambda}_i(t), \Delta \lambda_{i,b}(t))$.

The $\Delta \lambda(t)$ in (8) can be obtained by solving the following linear equations.

$$\Phi \Delta \lambda(t) + \frac{\partial \phi(\lambda(t))^T}{\partial \lambda} = 0 \qquad (9)$$

where the block diagonal matrix $\Phi = diag(\Phi_1, \ldots, \Phi_N)$, and the $i$ th diagonal block sub-matrix

$$\Phi_i = \begin{bmatrix} \Phi_{i00} & \Phi_{i0b} \\ \Phi_{ib0} & \Phi_{ibb} \end{bmatrix} \qquad (10)$$

in which

$$\Phi_{i00} = \frac{\partial \overset{0}{\overline{g}}_i(k)}{\partial \overset{0}{y}_i} \overset{0}{H}_i^{-1} \frac{\partial \overset{0}{\overline{g}}_i(k)}{\partial \overset{0}{y}_i}^T + \frac{\partial \overset{0}{\overline{g}}_i(k)}{\partial x_{i,b}} H_{i,b}^{-1} \frac{\partial \overset{0}{\overline{g}}_i(k)}{\partial x_{i,b}}^T \qquad (11)$$

$$\Phi_{i0b} = \frac{\partial \overset{0}{\overline{g}}_i(k)}{\partial \overset{0}{y}_i} \overset{0}{H}_i^{-1} \frac{\partial \overline{g}_{i,b}(k)}{\partial \overset{0}{y}_i}^T + \frac{\partial \overset{0}{\overline{g}}_i(k)}{\partial x_{i,b}} H_{i,b}^{-1} \frac{\partial \overline{g}_{i,b}(k)}{\partial x_{i,b}}^T \qquad (12)$$

$$\Phi_{ib0} = \frac{\partial \overline{g}_{i,b}(k)}{\partial \overset{0}{y}_i} H_{i,b}^{-1} \frac{\partial \overset{0}{\overline{g}}_i(k)}{\partial \overset{0}{y}_i}^T + \frac{\partial \overline{g}_{i,b}(k)}{\partial x_{i,b}} H_{i,b}^{-1} \frac{\partial \overset{0}{\overline{g}}_i(k)}{\partial x_{i,b}}^T \quad (13)$$

$$\Phi_{ibb} = \frac{\partial \overline{g}_{i,b}(k)}{\partial \overset{0}{y}_i} \overset{0}{H}_i^{-1} \frac{\partial \overline{g}_{i,b}(k)}{\partial \overset{0}{y}_i}^T + \frac{\partial \overline{g}_{i,b}(k)}{\partial x_{i,b}} H_{i,b}^{-1} \frac{\partial \overline{g}_{i,b}(k)}{\partial x_{i,b}}^T \quad (14)$$

and

$$\overset{0}{H}_i = \begin{bmatrix} \eta I_1 & 0 & 0 \\ 0 & \dfrac{\partial^2 c_i(k)}{\partial P_{G_i}^2} & 0 \\ 0 & 0 & \eta I_2 \end{bmatrix}$$

$$H_{i,b} = [\eta I_3]$$

where the matrices $I_1, I_2$ and $I_3$ are identity matrices with dimension of $|\overset{0}{x}_i| \times |\overset{0}{x}_i|$, $|\overset{0}{Q}_{G_i}| \times |\overset{0}{Q}_{G_i}|$ and $|x_{i,b}| \times |x_{i,b}|$, respectively, and $|(\cdot)|$ denotes the member of components of the vector $(\cdot)$. The $\dfrac{\partial \phi(\lambda(t))}{\partial \lambda}$ in (9) denotes the derivative of the dual function $\phi(t)$ with respect to $\lambda$ at $\lambda(t)$, and $\dfrac{\partial \phi(\lambda(t))}{\partial \lambda} = (\dfrac{\partial \phi(\lambda(t))}{\partial \lambda_1}, ..., \dfrac{\partial \phi(\lambda(t))}{\partial \lambda_N})$, in which $\dfrac{\partial \phi(\lambda(t))}{\partial \lambda_i} = (\dfrac{\partial \phi(\lambda(t))}{\partial \overset{0}{\lambda}_i}, \dfrac{\partial \phi(\lambda(t))}{\partial \lambda_{i,b}})$. $\dfrac{\partial \phi(\lambda(t))}{\partial \lambda_i}$ can be computed by

$$\frac{\partial \phi(\lambda(t))^T}{\partial \overset{0}{\lambda}_i} = \overset{0}{\overline{g}}_i(k) + \frac{\partial \overset{0}{\overline{g}}_i(k)}{\partial \overset{0}{y}_i} \Delta \hat{\overset{0}{y}}_i + \frac{\partial \overset{0}{\overline{g}}_i(k)}{\partial x_{i,b}} \Delta \hat{x}_{i,b} \quad (17)$$

$$\frac{\partial \phi(\lambda(t))^T}{\partial \lambda_{i,b}} = \overline{g}_{i,b}(k) + \frac{\partial \overline{g}_{i,b}(k)}{\partial \overset{0}{y}_i} \Delta \hat{\overset{0}{y}}_i + \frac{\partial \overline{g}_{i,b}(k)}{\partial x_{i,b}} \Delta \hat{x}_{i,b} + \frac{\partial \overline{g}_{i,b}(k)}{\partial x_{i,e}} \Delta \hat{x}_{i,e} \quad (18)$$

The $(\Delta \hat{\overset{0}{y}}_i, \Delta \hat{x}_{i,b}),\ i = 1,...,N$ required in (17) and (18) for $i = 1,...,N$ is the optimal solution, $\Delta \hat{y}_i$, of the minimization problem on the RHS of (6), which can be solved by solving N independent minimization subproblems as follows. For the given $\lambda(t) = (\lambda_1(t),...,\lambda_N(t))$, the minimization problem on the RHS of (6) can be decomposed into the following $N$ independent minimization subproblems:

For $i = 1,...,N$,

$$\begin{aligned} \min_{\Delta y_i \in \Omega_i(k)} \ & \Delta P_{G_i}^T \frac{\partial^2 c_i(k)}{\partial P_{G_i}} \Delta P_{G_i} + \frac{\partial c_i(k)}{\partial P_{G_i}} P_{G_i} \\ & + \overset{0}{\lambda}_i^T \left[ \overset{0}{\overline{g}}_i(k) + \frac{\partial \overset{0}{\overline{g}}_i(k)}{\partial \overset{0}{y}_i} \Delta \overset{0}{y}_i + \frac{\partial \overset{0}{\overline{g}}_i(k)}{\partial x_{i,b}} \Delta x_{i,b} \right] \\ & + \lambda_{i,b}^T \left[ \overline{g}_{i,b}(k) + \frac{\partial \overline{g}_{i,b}(k)}{\partial \overset{0}{y}_i} \Delta \overset{0}{y}_i + \frac{\partial \overline{g}_{i,b}(k)}{\partial x_{i,b}} \Delta x_{i,b} \right] \\ & + \sum_{j \in J_i} \left[ \lambda_{j,b}^T \frac{\partial \overline{g}_{i,e_j}(k)}{\partial x_{i,e_j}} \Delta x_{i,e_j} \right] \end{aligned} \quad (19)$$

where

$$\Omega_i(k) = \left\{ \Delta y_i \middle| h_i(k) + \frac{\partial h_i(k)}{\partial \overset{0}{y}_i} \Delta \overset{0}{y}_i + \frac{\partial h_i(k)}{\partial x_{i,b}} \Delta x_{i,b} \le 0 \right\}$$

and $\Omega(k) = \bigcup_{i=1}^{N} \Omega_i(k)$ ; $J_i$ denotes the index set of subsystems connecting with subsystem $i$, and $\lambda_{j,b}$ denotes the Lagrange multipliers associated with the equality constraints of subsystem $j$ involving the complex voltage of the boundary buses of subsystem $i$. Once the optimal solution of (19) are obtained, we can calculate $\dfrac{\partial \phi(\lambda(t))}{\partial \lambda}$ by (17) and (18). Then we can solve $\Delta\lambda(t)$ from (9) by first decomposing (9) into the following $N$ independent set of linear equations,

$$\Phi_i \Delta \lambda_i(t) + \frac{\partial \phi(\lambda(t))^T}{\partial \lambda_i} = 0,\ i = 1,...,N \quad (20)$$

then solving these $N$ independent set of linear equations to obtain $\Delta\lambda(t) = (\Delta\lambda_1(t),...,\Delta\lambda_N(t))$. We will then update $\lambda(t)$ by (8) using an Armijo-type step-size $\beta(t)$ [10]. The iterations of the DPPQN method will continue until they converge to an optimal $\lambda^*$ for the dual problem (5). The corresponding optimal solution on the RHS of (6) when $\lambda = \lambda^*$ will be $(\Delta \overset{0}{y}_i(k), \Delta x_{i,b}(k)), i = 1,...,N$ which will be used to update $y(k+1)$ in (3) using an Armijo-type step-size $\alpha(k)$ [10]. Then we will start the next iteration of the SQP method. The iterations of the SQP method will

continue until it converges to the optimal solution of the DOPF (1).

## 4. The Distributed Algorithm

From the computational formulae of the SQP method and the DPPQN method described above, we see that the SQP method does nothing but update $y(k)$ in each iteration as shown in (3), which of course can be carried out in individual subsystems. The major computations required in the DPPQN method are solving the optimization problems (19) and the linear equations (20), which are already decomposed such that each subsystem can execute its part of the computations as long as the necessary data from the connecting subsystems are passed through the computer network. This indicates that the proposed solution method is very suitable for implementation in a distributed computer network.

In order to govern the synchronization of the convergence of the DPPQN and SQP methods in the distributed computing network, we assign a root sub-system among the connecting sub-systems to be responsible for this task. Following is the distributed algorithmic steps for each subsystem $i$:

Step 0: Initially guess $y_i$; initially guess $\lambda_i$.

Step 1: Send $\lambda_{i,b}$ to connecting subsystems.

Step 2: Once receiving all $\lambda_{j,b}$, $j \in J_i$ from all connecting subsystems, compute $\Delta\hat{y}_i$ from solving (19).

Step 3: Send $\Delta\hat{x}_{i,b}$ to connecting subsystems.

Step 4: Once all $\Delta x_{j,b}$, $j \in J_i$ are received, calculate $\frac{\partial\phi(\lambda)}{\partial\lambda_i} = (\frac{\partial\phi(\lambda)}{\partial\overset{0}{\lambda}_i}, \frac{\partial\phi(\lambda)}{\partial\overset{0}{\lambda}_{i,b}})$ by (17) and (18).

Step 5: Compute $\Phi_i$ and solve $\Delta\lambda_i$ from (20)

Step 6: If $\|\Delta\lambda_i\| < \varepsilon$, send a signal to the root subsystem to inform the convergence of DPPQN method in this subsystem.

Step 7: Update $\lambda$ by (8) and return to Step 1.

Step 8: Once receiving the signal to update $y_i$ from the root subsystem, compute $\Delta y_i$ from (19) and update $y_i$ by (3). If $\|\Delta y_i\| < \varepsilon$, send a signal to the root sub-system to inform the convergence of the SQP method in this subsystem and return to Step 1.

As indicated above, an extra task for the root subsystem in addition to the above distributed algorithmic execution steps is checking the system-wise convergence of the DPPQN and SQP methods. Thus, in Step 6, if the root subsystem receives the signal indicating the convergence of the DPPQN method from all subsystems, it will send a signal to all subsystems to update $y$. Similarly, if the root subsystem receives the convergence signal of the SQP method from all subsystems, it will send a signal to all subsystems to stop the algorithm and output the solution.

## 5. Test Results

We have applied our distributed algorithm to solve the DOPF on the IEEE 118-bus system, which is arbitrarily partitioned into four subsystems. These four subsystems are indexed by A1, A2, A3 and A4, respectively. The interconnecting relationships of these four subsystems is shown in Fig. 1. The power selling and purchasing pairs are (A1,A2), (A1, A4), (A1, A3), and (A2,A3). The cost function of each generation bus in each subsystem is a quadratic function of the real power generation. We have arbitrarily assumed the amount of power sale in each selling and purchasing pair and distribute the power among the tie lines.

Since we do not have a distributed computing network at hand, we currently simulate our distributed algorithm in a Pentium IV PC. The final objective value we obtain is 46,414 dollars/hour, and the consumed sequential CPU time is 0.62 seconds. However, if we take the parallel computation effect into account but not including the data communication time, the CPU time for the longest subsystem is 0.24 seconds. To verify our result, we also solve the DOPF using the centralized DPPQN method based algorithm [10] and obtain the following results: the final objective value is 46,412 dollars/hour, and the consumed CPU time is 0.49 seconds. This demonstrate that our distributed algorithm does converge to the true solution, and the consumed CPU time is much faster than the centralized method if we take the parallel computation effect into account.

## 6. Conclusion

Due to the trend of deregulation, DOPF becomes a focus in power system research. We have proposed a DOPF algorithm under deregulated environment in this paper. The communication requirement in the algorithm can be easily handled. The test results demonstrate the superiority of the proposed distributed algorithm.

## 7. Acknowledgements

This research work is supported by the National Science Council in Taiwan, R.O.C. under the grant NSC93-2218-E-009-036.

## References

[1] B. Stott, J. L. Marinho, and O. Alsac, "Review of linear programming applied to power system rescheduling," pp. 142154, *PICA* 1979.

[2] T. C. Giras and S. N. Talukdar, "Quasi-Newton method for optimal power flows," *International Journal of Electrical Power Energy Systems*, vol.3, no.2, pp.59-64, Apr. 1981.

[3] R. C. Burchett, H. H. Happ, and D. R. Vierath, "Quadratically convergent optimal power flow," *IEEE Trans. on Power Apparaius and Systems*, vol. PAS-103, no.10, pp.2864-2880, Oct. 1984.

[4] D. I. Sun, I. I. Hu, G. S. Lin, C. J. Lin, and C. M. Chen, "Experiences with implementing optimal power flow for reactive scheduling in the Taiwan power system," *IEEE Trans. on Power Systems*, vol.3, no.3, Aug. 1988.

[5] Y. C. Wu, A. S. Debs and R. E. Marsten, "A direct nonlinear predictor-corrector primal-dual interior point algorithm for optimal power flows," *IEEE Trans. on Power System,* pp.876-883, May. 1994.

[6] Kim, G.H. and Baldick, R., "Coarse-grianed distributed optimal power flow," *IEEE Trans. on Power Systems,* vol.12, no.2, pp.932-939, May 1997.

[7] Kim, G.H. and Baldick, R., "A comparison of distributed optimal power flow algorithms," *IEEE Trans. on Power Systems,* vol.15, no.2, pp.599-604, May 2000.

[8] Hur, D., Park, J.-K and Kim, B.H., "Evaluation of convergence rate in the auxiliary problem principle distributed optimal power flow," *Generation, Transmission and Distribution, IEE Proceedings-,* vol.149, pp.525-532, Sept. 2002.

[9] Hur, D., Jong-Keun Park, Kim, B.H., and Kwang-Myoung son, "Security constrained optimal power flow for the evaluation of transmission capability on Korea electric power system," *Power Engineering Society Summer Meeting,* vol.2, pp. 1133-1138, 2001.

[10] C.-H. Lin and S.-Y. Lin, "A new dual-type method used in solving optimal power flow problems," *IEEE Trans. Power Syst.*, vol.12, pp.1667-1675, Nov. 1997.

[11] ---, "Improvements on the dual-type method used in solving optimal power flow problems," *IEEE Trans. Power Syst.,* vol.17, pp.315-323, May 2002.

[12] S.-Y. Lin, Y.C. Ho, and C.-H Lin, "An ordinal optimization theory based algorithm for solving the optimal power flow problem with discrete control variables," *IEEE Trans. on Power Syst.,* vol. , pp. , Feb. 2004.

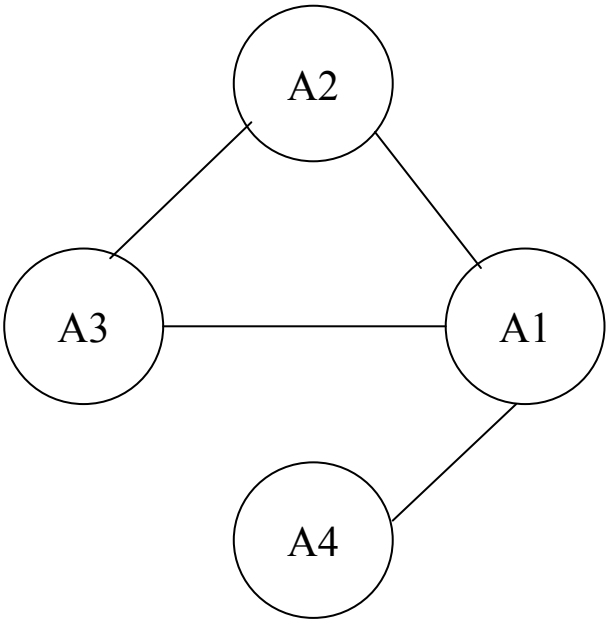


Figure 1. The interconnecting relationships between the four subsystems.