Revisions Responding to Reviewers’ Comments on Paper IJITDM-D-11-00183

First of all, we would like to thank the six anonymous reviewers for spending their valuable time in reviewing our paper and given us constructive comments and helpful suggestions to improve the quality and readability of our paper. We would also like to thank the managing editor, Prof. Gang Kou, for his help in processing our paper and giving us a chance to do the revisions.

I wish to show my respect to the management of this journal, because I never have any paper being reviewed by six reviewers. Above all, all the comments from all six reviewers are so valuable that help us improve the quality of our paper a lot. Although it is really a big job to complete the revisions and make us exhausted, we are happy with what we have improved. Thanks to all of you.

Now, we believe that we have addressed all the comments successfully in the revised manuscript. In the following, we will state each comment raised by the six reviewers first then followed by the revisions responding to that comment, and we will also indicate where the revisions are in the revised manuscript. Each comment is labeled such that R1.2 represents the second comment of reviewer #1.

Comment R1.1: The algorithm used (ES) is known as 'Evolution Strategy' NOT 'Evolutionary Strategy'.

Revisions: We wish to thank reviewer #1 for this valuable comment. As suggested, we have changed “Evolutionary Strategy” to “Evolution Strategy” throughout the paper including the title.

Comment R1.2: The authors did not provide any convincing arguments that would justify their choice of ANN topology and architecture. In fact for ANN based meta-models overfitting and underfitting are often huge problems and coming up with a network of right complexity is a challenging task. To circumvent this problem, in recent times, the concept of Evolutionary Neural Net has been proposed, which constructs a Pareto tradeoff between the training error and network complexity through multi-objective genetic algorithms. The authors need to consult the following papers and enhance their discussion this very important issue:

(1) Title: Cu-Zn separation by supported liquid membrane analyzed through Multi-objective Genetic Algorithms

Author(s): Mondal Debanga Nandan; Sarangi Kadambini; Pettersson Frank; et al.

Source: HYDROMETALLURGY Volume: 107 Issue: 3-4 Pages: 112-123

DOI:10.1016/j.hydromet.2011.02.008 Published: MAY 2011

(2) Title: Analyzing Leaching Data for Low-Grade Manganese Ore Using Neural Nets and Multiobjective Genetic Algorithms

Author(s): Pettersson Frank; Biswas Arijit; Sen Prodip Kumar; et al.

Source: MATERIALS AND MANUFACTURING PROCESSES Volume: 24 Issue: 3 Pages: 320-330 Article Number: PII 908801496 DOI: 10.1080/10426910802679386 Published: 2009

(3) Title: A genetic algorithms based multi-objective neural net applied to noisy blast furnace data

Author(s): Pettersson F.; Chakraborti N.; Saxen H.

Source: APPLIED SOFT COMPUTING Volume: 7 Issue: 1 Pages: 387-397

DOI:10.1016/j.asoc.2005.09.001 Published: JAN 2007

Revisions: We wish to thank reviewer #1 for this very important comment. We agree with him/her that improperly trained neural networks may suffer from either underfitting or overfitting. Therefore, in the revised manuscript, we have added a remark, Remark 2, in Section 2.2.1 starting from page 7, line 9 to page 8, line 3 to address this issue and cited the suggested literatures [29]-[31]. These revisions are restated in the following for easier reference.

> **Remark 2**: Improperly trained ANN may cause underfitting or overfitting problems, which will lead to predictions that are far beyond the range of the training data set and produce wild predictions even with noise-free data. To cope with such problems, several techniques were developed such as model selection, jittering, early stopping, weight decay, Bayesian learning, combining networks, and evolutionary neural networks [29]-[31]. Among these techniques, evolutionary neural network, which constructs a Pareto tradeoff between the training error and network complexity through multi-objective genetic algorithms, is considered to be very effective. For example, Mondal *et al.* proposed a data driven model, which involved using an evolutionary neural network that used multi-objective genetic algorithms to configure its weights and topology, to construct for the Cu-Zn separation process [29]. Pettersson *et al.* used a model based upon evolving neural nets and multiobjective optimization conducted through genetic algorithms for the acid leaching of low-grade manganese ores [30]. Pettersson *et al.* utilized a genetic algorithms based multi-objective optimization technique in the training process of a feed forward neural network and applied to noisy data from an industrial iron blast furnace [31]. Fortunately, the OO theory claims that the performance "order" of solutions is likely preserved even if they are evaluated using a rough model. Therefore, a sophisticated evolutionary neural network will definitely help in selecting good solutions. However, the *easily implemented*

feed-forward back propagation ANN can be used as the rough surrogate model required in the OO approach also. The details of the employed ANN will be described in the application example, which will be presented in Section 3.

[29] D.N. Mondal, K. Sarangi, F. Pettersson, P.K. Sen, H. Saxen and N. Chakraborti, Cu-Zn separation by supported liquid membrane analyzed through multi-objective genetic algorithms, *Hydrometallurgy* **107**(3-4) (2011) 112-123.
[30] F. Pettersson, A. Biswas, P.K. Sen, H. Saxen and N. Chakraborti, Analyzing leaching data for low-grade manganese ore using neural nets and multiobjective genetic algorithms, *Mater. Manuf. Process*. **24**(3) (2009) 320-330.
[31] F. Pettersson, N. Chakraborti and H. Saxen, A genetic algorithms based multi-objective neural net applied to noisy blast furnace data, *Appl. Soft. Comput.* **7**(1) (2007) 387-397.

Comment R1.3: Although this paper is intended for the electronics industry, the background information provided for such applications is very poor. The authors significantly need to improve it. Two very important recent papers are indicated below, which the authors need to discuss along with some more similar articles published recently:

(1) Title: Hybrid Differential Evolution and Particle Swarm Optimization Approach to Surface-Potential-Based Model Parameter Extraction for Nanoscale MOSFETs

Author(s): Li Yiming; Tseng Yu-Hsiang

Source: MATERIALS AND MANUFACTURING PROCESSES Volume: 26 Issue: 3 Pages: 388-397 Article Number: PII 936137099 DOI: 10.1080/10426914.2010.526977 Published: 2011

(2) Title: Optimization of Wavelet-Filtered In-Situ Plasma Etch Data Using Neural Network and Genetic Algorithm

Author(s): Kim Byungwhan; Kim Daehyun; Han Dongil; et al.

Source: MATERIALS AND MANUFACTURING PROCESSES Volume: 26 Issue: 3 Pages: 398-402 Article Number: PII 936136671 DOI: 10.1080/10426914.2010.520791 Published: 2011

Revisions: We wish to thank reviewer #1 for this constructive comment. To address this comment, we have added a paragraph at the beginning of Section 1 starting from page 2, line 2 to page 3, line 12 and revise the paragraph that follows. We also cited the suggested literatures [2]-[3] on pages 37. The added and revised paragraphs and the cited references are restated in the following for easier reference.

Numerous engineering optimization problems in electronics industry are highly nonlinear, and global optimum of these problems are usually sought. Therefore, typical solution methods for such a type of problems are nature-inspired meta-heuristic methods [1]. For example, Li and Tseng developed an extraction technique that combined the differential evolution method and particle swarm optimization algorithm to obtain a set of optimal model parameters of the surface-potential-based PSP model for the sub-45-nm MOSFETs [2]. Kim *et al.* applied the back propagation neural network to model the wavelet-filtered optical emission spectroscopy data and used the genetic algorithm to optimize the prediction performance of neural network model [3]. Sohn and Lee used a correlation analysis to investigate the relationship between multiple process control monitoring variables and various probe bin variables such that the yield can be maximized [4]. Ozbakir *et al.* proposed a two-level algorithm which combined a multilayer perceptron neural network and touring ant colony optimization technique to determine the most effective parameters on the quality defects in fabric production [5].

The decision problem in wafer testing process considered in this paper is a real-time combinatorial stochastic simulation optimization problem with huge discrete solution space, which is harder than the above mentioned problems due to its nature of expensive optimization. Expensive optimization problems that posses a time-consuming-evaluation objective function such as stochastic simulation optimization problems [6]-[7], multiple objective optimization problems [8]-[9], some combinatorial optimization problems [10]-[11] and many others have been a challenge to researchers in this area for decades. Evolutionary algorithms (EAs) assisted by surrogate models and memetic algorithms provide a possible solution for such problems [12]-[14]. However, most memetic algorithms are designed for continuous-variable problems, which are different from the combinatorial nature of the considered problem. Additionally, real-time application problems usually allow very limited computing budget for solution process. Due to the tight computing budget, optimality in the real-time application problem is usually traded off by a "good enough" solution that can be obtained in real-time. In fact, using *limited computing time* to solve for a good enough solution of expensive optimization problem is the core concept of *ordinal optimization* (OO) proposed by Y. C. Ho [15]-[16]. The basic idea of OO can be stated as follows. Let S denote the set of *estimated* good enough designs selected from N given designs using *surrogate model*, and let G denote the *actual* good-enough subset of the N designs, then there is a high probability that S contains at least $k(\geq 1)$ elements of G. Therefore, the first step of OO is using a computationally efficient surrogate model to evaluate the N given designs and select the estimated top |S|, the cardinality of S, designs. The second step is using the time-consuming exact model to evaluate each of the selected designs in S, and the best one will be the final good enough solution with high probability.

[2] Y.M. Li and Y.H. Tseng, Hybrid differential evolution and particle swarm optimization approach to surface-potential-based model parameter extraction for nanoscale MOSFETs, *Mater. Manuf. Process*. **26**(3) (2011) 388-397.
[3] B. Kim , D. Kim, D. Han and N.I. Lee, Optimization of wavelet-filtered in-situ plasma etch data using neural network and genetic algorithm, *Mater. Manuf. Process*. **26**(3) (2011) 398-402.

Comment R2.1: The system is properly described.
Response: We appreciate review #2 for his/her support to our paper.

Comment R2.2: Every time you make a clear affirmation you have to reference it. In page 4, authors state: "ANN, because it is not only perfect for approximating the continuous-variable function but also competent for approximating ..." I have used ANN for years and I agree with you; however state that "ANN are perfect for approximating the function" is too much! I recommend changing that statement to: "ANN, because it is competent for approximating both the continuous-variable function and ..." (or something like that).

Revisions: We wish to thank reviewer #2 for this professional comment. As suggested, we have rewritten the sentence on page 6, lines 15-17, which are restated in the following for easier reference.

> Therefore, we will use ANN as the surrogate model in this paper, because it is competent for approximating both highly nonlinear continuous-variable function and the input-output relationship of stochastic discrete event simulated systems [27].

Comment R2.3: Figure 3 doesn't show and print correctly.

Revisions: As suggested, Figure 8 on page 31, which is Figure 3 in the original manuscript, has been enlarged for the sake of visibility, which is also presented in the following for easier reference.

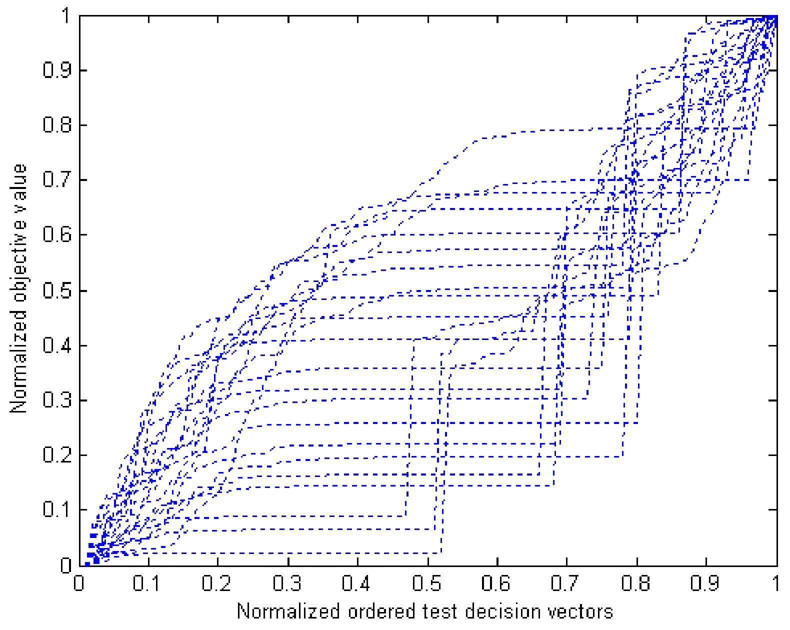

Fig. 8. The range of OPCs of (3).

Comment R2.4: Figure 3, 4 and 5 should be bigger for the sake of visibility.

Revisions: As suggested, Figures 8, 9, and 10 on page 31, 32, and 33, which are Figures 3, 4 and 5 in the original manuscript, respectively, have been enlarged for the sake of visibility. These figures are also presented in the following for easier reference.

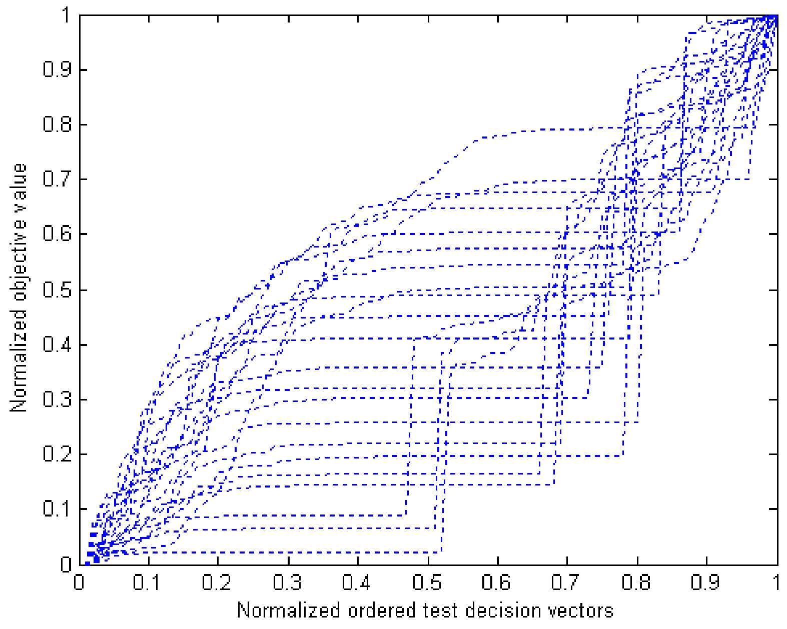

Fig. 8. The range of OPCs of (3).

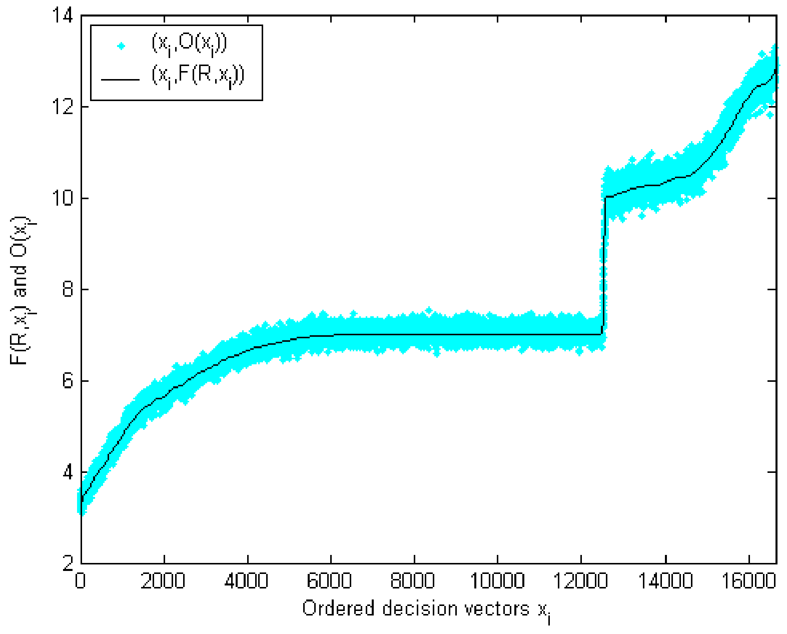

Fig. 9. $F(\bar{R}, x_i)$ and $O(x_i)$ for the M' ordered decision vectors x_i.

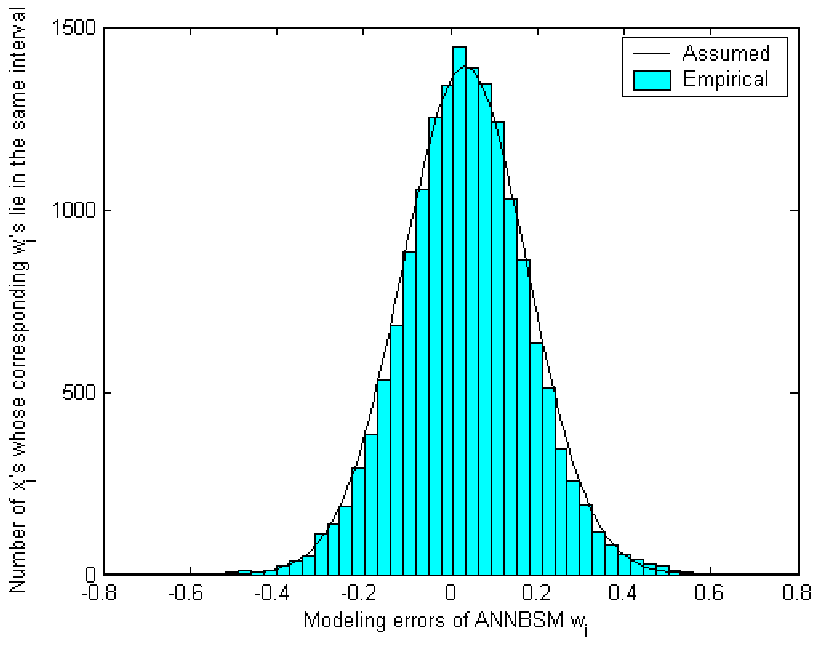

Fig. 10. Histogram of the modeling errors of ANNBSM.

Comment R2.5: Results discussion should be extended and detailed.

Revisions: As suggested, we have done some more tests and comparisons to demonstrate the superiority of our method. In addition, we have also rewritten the part of tests and comparisons in Section 3.4 in a more structural manner. The revisions for indicating the structure of the presentation, the additional test results and additional comparisons are presented in details from last line, page 21 to line 6, page 22; from line 13, page 25 to line 10, page 27 and on page 28, lines 1-14, respectively, which are restated in the following for easier reference.

From last line, page 21 to line 6, page 22:

In this section, we will proceed with the tests and comparisons in three phases. In the first phase, we will test the proposed ($\mu+\lambda$)-ES+OO as well as the GA+OO and compare their performances. In the second phase, we will investigate the merit of using OO approach by comparing the performances of the proposed ($\mu+\lambda$)-ES+OO with the ($\mu+\lambda$)-ES, GA and PSO associated with the exact model. In the third phase, we will use the solution obtained by the proposed ($\mu+\lambda$)-ES+OO in real application and investigate the quality of the obtained solution.

From line 13, page 25 to line 10, page 27:

In phase two, we will demonstrate the merit of using OO approach by comparing the proposed ($\mu+\lambda$)-ES+OO with the ($\mu+\lambda$)-ES, GA and PSO associated with the exact model to solve (3) for the three cases of $\overline{R}$=30, 60 and 90. However, for the sake of simplicity, we will present the comparison results only for case $\overline{R}$ =60.

The parameters employed in the ($\mu+\lambda$)-ES and GA associated with the exact model are the same as those used in ($\mu+\lambda$)-ES+OO and GA+OO presented previously. In the PSO associated with the exact model, the size of population $|Q|$ was also set to be 1000. Both of the *cognitive parameter* and *social parameter* were set to be 2.05 and the *inertia factor* was set to be 1. The *maximum allowable velocity* was set to be 0.5 and the *fitness* of each *particle* was evaluated by the exact model. We also execute each of the three methods for 30 simulation runs. Due to the time-consuming solution process, we terminate their execution when the consumed CPU times exceed 200 minutes, which is around 107 times of the CPU time consumed by the proposed ($\mu+\lambda$)-ES+OO. The progression of the best-so-far objective value with respect to the consumed CPU time marked by "+", "□" and "●" in Fig. 6 are obtained by ($\mu+\lambda$)-ES, GA and PSO associated with the exact model in the first simulation run, respectively. The point marked by "∗" in Fig. 6 represents the pair of the objective value of (3) and the consumed CPU time obtained by the proposed ($\mu+\lambda$)-ES+OO. From Fig. 6, we can observe that when ($\mu+\lambda$)-ES associated with the exact model consumes 107 times of the CPU times consumed by the proposed ($\mu+\lambda$)-ES+OO, the best-so-far objective value obtained by the former is still much larger than that obtained by the latter. Similar conclusions apply to the GA and PSO associated with the exact model. This demonstrates the merit of using OO approach. Notably, the performance of the ($\mu+\lambda$)-ES associated with the exact model is better than GA and PSO associated with the exact model in this case. In fact, the comparisons of the rest 29 simulation runs reach similar conclusions as stated above.

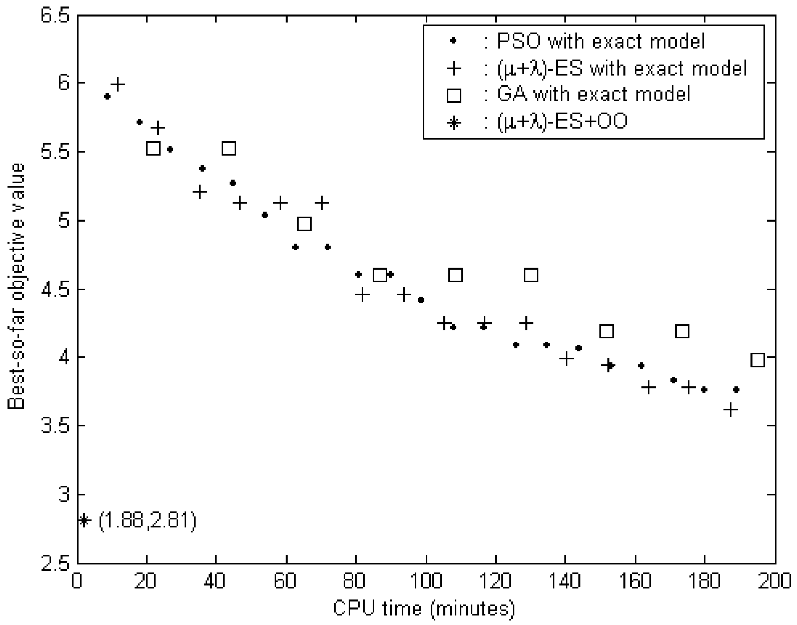

Fig. 6. Progression of the best-so-far objective value obtained by three comparing methods in the first simulation run.

Table 5 shows the statistics of the resulted objective values of the 30 simulation runs of the three methods. For the sake of comparison, we also show the corresponding data obtained by the proposed ($\mu+\lambda$)-ES +OO. These test results reveal that when the three comparing methods consume more than 107 times of the CPU times consumed by the proposed ($\mu+\lambda$)-ES +OO, the mean objective value obtained by the proposed method is still 24.74%, 36.93% and 29.27% smaller than that obtained by the ($\mu+\lambda$)-ES, GA, and PSO with the exact model, respectively. These results further confirm the computational efficiency of and the solution quality obtained by the proposed ($\mu+\lambda$)-ES +OO.

Table 5. Statistics of the objective values resulting from 30 simulation runs for the case $\overline{R}$ =60 using four methods.

Methods	Maximum	Minimum	Mean	Standard deviation
($\mu+\lambda$)-ES+OO	3.02	2.73	2.87	0.08
($\mu+\lambda$)-ES with exact model	4.03	3.06	3.58	0.19
GA with exact model	4.37	3.19	3.93	0.32
PSO with exact model	4.22	3.31	3.71	0.23

Similar conclusions apply to the cases of $\overline{R}$ =30 and $\overline{R}$ =90.

On page 28, lines 1-14:

Finally, in phase three, we use 590 real test wafers, whose bad dies b_{jk} and overkills v_{jk} before retest are known, to test the performance of the vector of threshold values $\hat{x}$ obtained by ($\mu+\lambda$)-ES+OO in the first simulation run of the three cases shown in Table 2. The test results of the pair ($\mathrm{E}[V(\hat{x})]$, $\mathrm{E}[R(\hat{x})]$) for these 590 test wafers with $\overline{R}=90$, $\overline{R}=60$ and $\overline{R}=30$ are shown in Fig. 7 by the points "○", "*", "△", respectively. We have also used 3000 randomly generated vectors of threshold values to test the same 590 test wafers. The resulting pairs of ($\mathrm{E}[V(\hat{x})], \mathrm{E}[R(\hat{x})]$) for these randomly generated threshold values are shown by the points "•" in Fig. 7. Since the two objectives, minimizing overkills and retests, have conflicting nature, the considered problem possesses Pareto optimal solutions, which are shown as the lower boundary of the region resulted from the randomly generated vectors of threshold values. From Fig. 7, we can observe that the pair ($\mathrm{E}[V(\hat{x})]$, $\mathrm{E}[R(\hat{x})]$) resulting from the solution obtained by the proposed ($\mu+\lambda$)-ES+OO for the three cases $\overline{R}=90$, $\overline{R}=60$ and $\overline{R}=30$ are all on the lower boundary. This implies that our algorithm not only controls the level of retests but also obtain a near Pareto optimal solution.

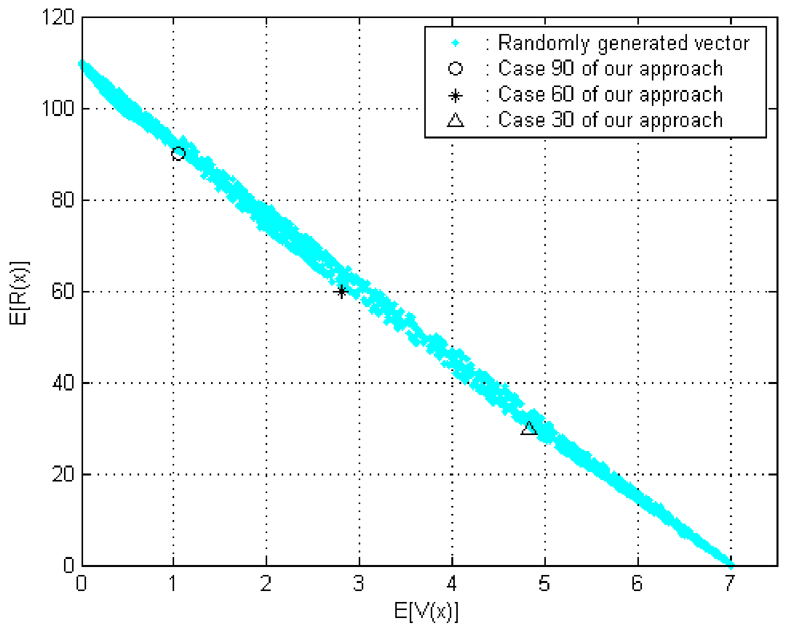

Fig. 7. The pairs of ($\mathrm{E}[V(\hat{x})]$, $\mathrm{E}[R(\hat{x})]$) resulting from the solution obtained by ($\mu+\lambda$)-ES+OO and the 3000 randomly generated vectors of threshold values.

Comment R2.6: No future works are given: does this work will continue? do you expect to improve it and achieve new results? how?

Revisions: To address this comment, we have added several statements about future works in the section of conclusion on page 37 lines 4-12, which are restated in the following for easier reference.

> We have successfully applied ($\mu+\lambda$)-ES+OO to obtain the good enough threshold values of wafer testing problem. However, probing errors still arise from several sources, including contaminative probes, filthy tester, setup mistake of probe station, and improper probe card position. There are many factors that may cause overkills, such as the testing errors of test program on probe station, probes, probe card, tester, material, and operational mistakes of operator. Thus, our future work is to perform bin analysis from wafer bin map, and propose a machine learning approach for fault extraction and isolation of overkill to improve yield in semiconductor manufacturing.

Comment R2.7: The "state of the art" is adequate (most of the references are actual, although some of them are a bit old).

Revisions: To address this comment, we have included five recent references [20], [23]-[24], [27]-[28] on pages 39-40, which are restated in the following for easier reference.

[20] S.B. Andersen and I.F. Santos, A Evolution strategies and multi-objective optimization of permanent magnet motor, *Appl. Soft. Comput.***12**(2) (2012) 778-792.

[23] Y.F. Li, S.H. Ng, M. Xie and T.N. Goh, A systematic comparison of metamodeling techniques for simulation optimization in decision support systems, *Appl. Soft. Comput.*

10(4) (2010) 1257-1273.

[24] J. Zurada, A.S. Levitan and J. Guan, A comparison of regression and artificial intelligence methods in a mass appraisal context, *J. R. Estate Res.* **33**(3) (2011) 349-387.

[27] B. Can and C. Heavey, A comparison of genetic programming and artificial neural networks in metamodeling of discrete-event simulation models, *Comput. Oper. Res.* **39**(2) (2012) 424-436.

[28] J.L. Devore, *Probability and Statistics for Engineering and Science*. 8th ed. (Thomson Brooks/Cole, CA, 2011).

Comment R2.8: Authors should leave his source code accessible to other researchers in order to facilitate the reproduction of scientific results (as the need for reproducibility is a bedrock requirement of modern science).

Revisions: As suggested, we have added a remark, Remark 8, on page 29, lines 1-2, which is restated in the following for easier reference. The technical report [37] listed on page 41, line 3-5, containing all source codes will be uploaded on-line once the paper is accepted.

Remark 8: All the above tests are carried out in a Pentium IV PC using Matlab 6.5 (R13). Interested readers may refer [37] for source code of the proposed method.

[37] S.C. Horng and S.Y. Lin, Combining evolution strategy with ordinal optimization for solving wafer testing problem (2012), *Technical Report NSC100-2221-E-324-006*, http://www.cyut.edu.tw/~schong/eng/paper.htm.

Comment R3.1: The performance analysis of the implementation is quite extensive (I would say too extensive and a bit tiring). Since the approach seems quite general and not problem specific, I would like to see it also applied on other problems. The performance analysis doesn't need to be that thorough in all problems. The writing of the paper is at a very good level.

Revisions: We wish to thank reviewer #3 for his/her support to our paper and the kind suggestions regarding the performance analysis. To address this comment, we have added a remark, Remark 9, at the end of Section 4 starting from line 20, page 36 to line 1, page 37 to illustrate the generality of the proposed performance evaluation procedures. This remark is restated in the following for easier reference.

Remark 9: From the above performance analysis procedures, we see that different application problems may have different OPCs and different modeling errors of the ANNBSM. However, once both OPC and the probability distribution of the modeling errors are obtained, the performance analysis is fairly general.

Comment R4.1: The authors have already employed GA+OO to tackle the wafer testing problem and in this paper they combine ES with OO to achieve the same work. As the GA and ES belong to an identical class of the meta-heuristics and their search strategies are the same in essence, why the authors do not utilize the PSO or ACO or even HS which are more efficient than the ES. The authors should be enough specific to address this critical comment.

Revisions: We wish to thank reviewer #4 for his/her in-depth comments. To address this comment, we have added a paragraph in Section 1 in page 3, line 20 to page 4, line 15 to briefly describe the mentioned methods as well as the GA and ES. However, as suggested, we have also done more tests, which include the utilization of PSO associated with the exact model to solve the considered problem and make comparison. The additional tests and comparisons are presented in Section 3.4 starting from line 2 page 18 to line 14 page 28. Please also refer to the revisions responding to comment R2.5. However, the revised paragraph is restated in the following for easier reference.

> OO is not, itself, a method but it is, rather, a supplement to existing meta-heuristic optimization methods such as the particle swarm optimization (PSO), ant colony optimization (ACO), harmony search (HS), genetic algorithm (GA), and evolution strategy (ES). Among them, PSO and ACO are two popular nature-inspired methods and HS is a music-inspired method [18]. PSO finds the optimal solution by moving the particles in the search space based on the balance of personal experience and best group experience. ACO solves the optimization problems by moving the tracks of ants with the assistance of the pheromone. HS searches for global optima using harmony improvisation operators to iteratively update the harmony memory that contains promising candidate solutions. Although the above three inspired methods seem to be more efficient than the GA, research of them is experimental rather than theoretical, and the theoretical analysis is extremely difficult due to sequences of probabilistic choices [1]. GA and ES belong to the class of evolutionary algorithms that apply mutation, recombination, and selection to a population of individuals containing candidate solutions for evolving iteratively better solutions. GA has the merit of a simple binary string encoding for discrete solution; however, its convergence velocity and convergence reliability may not be as good as ES [19]-[21] when the solution space is huge. It has been demonstrated that in some systems, ES appears to outperform GA, especially in the field of parameter optimization [21]. Although most ESs are designed for continuous variable problem, it can be modified to handle the discrete solution as will be manifested in this work.
>
> Therefore, in this work, we will propose a combination of ES with OO, which is abbreviated as ES+OO, to solve the considered problem.

Comment R4.2: In the ANN model, the authors have employed 15 neurons in the hidden layer. As the number of the input and output neurons are determined according to the data structure, the number of hidden layer neurons can highly affect the computational performance of the ANN. In the sequel the authors should report their sensitivity analyses results and demonstrate that the number of selected hidden layer neurons is efficient.

Revisions: As suggested, we have added a remark, Remark 6, in Section 3.4 and the associated experimental results shown in Figure 5 to explain why we use 15 neurons in the hidden layer.

The revisions are on page 21, lines 6-14, which are restated in the following for easier reference.

Remark 6: The selection of the best number of neurons in the hidden layer depends on many factors. The size of the training set, amount of noise in the targets, complexity of the sought function to be modeled, type of activation functions used and the training algorithm all have interacting effects on the sizes of the hidden layers. There is no analytical method for determining the best number of neurons in the hidden layer. Therefore, we select the number of neurons in the hidden layer based on empirical tests. Fig. 5 shows the mean square errors (MSE), which is computed by (6), of various number of neurons in the hidden layer ranging from 1 to 25. From this figure, we can observe that 15 is the least number of neurons in the hidden layer that will achieve the smallest MSE.

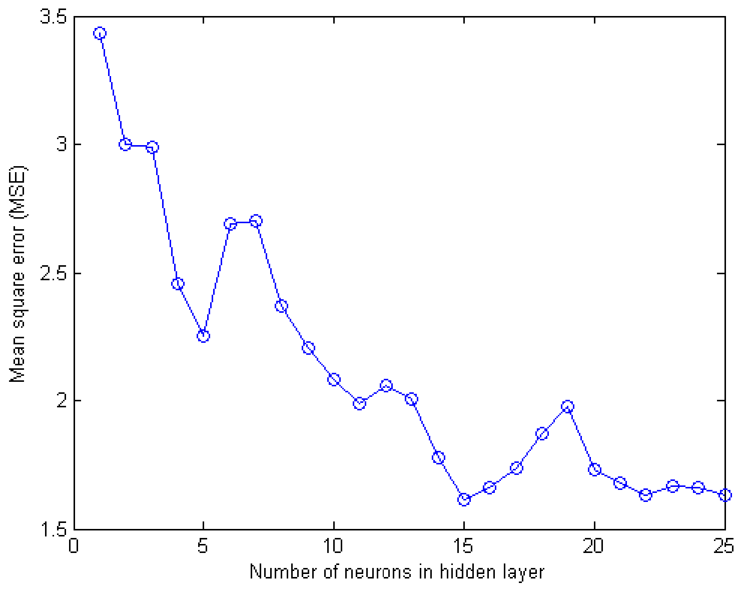

Fig. 5. The MSE of various numbers of neurons in hidden layer.

Comment R4.3: One of the problems that occur during ANN training is overfitting. Reading the paper I realize that the authors have not considered any computational strategy to avoid this problem.

Response: We wish to thank reviewer #4 for this important comment. Indeed, it is our responsibility for not touching the overfitting or underfitting issue in the original manuscript; therefore, two reviewers, reviewers #1 and 4, express their concern on this subject. In fact, this comment is part of the comment R1.2, which has been addressed. Please refer to the revisions responding to comment R1.2.

Comment R5.1: First of all, the authors should compare GA+OO and ES+OO in a slightly different way. When comparing meta-heuristics one should fix the number of evaluations of the fitness functions, and compare the results obtained for the same number of evaluations (or even for several values). This gives a fairer idea of the difference of performance of the two different meta-heuristics, so nobody can argue that maybe the stopping criteria is favoring one of the method. Additionally, you can compare also the CPU time, because the operators applied for one or the other method could involve different computational expenses.

Response: We wish to thank reviewer #5 for this important comment. Due to the random nature of the considered problem and the random nature of ES and GA, we have repeated the simulation process for each method and each case for 30 times. As suggested, we compare the performance of GA+OO and ($\mu+\lambda$)-ES+OO in a fairer manner by directly pointing the objective values obtained and the CPU time consumed by them. We also report the mean and standard deviation of the objective value obtained and the average CPU times consumed by the GA+OO and ($\mu+\lambda$)-ES+OO for the 30 simulation runs. These additional simulation results and revisions have been described in the revisions responding to comment R2.5.

Comment R5.2: (i) Why some comparisons (Tables 2 and 4) are done only once? You should repeat the process a certain number of times (say 30), obtain a mean and standard deviation, and compare these values. (ii) You should also apply statistical tests to check if the differences obtained are statistically significant, otherwise your conclusions could be based only on the stochastic nature of the methods and processes involved in the experimentation.

Revisions: (i) We wish to thank reviewer #5 for this important comment. As suggested, we have repeated the simulation process for 30 times for each algorithm and each case. Therefore, we have rewritten the paragraphs that introduce Table 2 and add a new table, Table 3, to present the mean and standard deviation of the objective values obtained and the average CPU times consumed by the two comparing algorithms in all considered cases. These revisions appear from page 23, line 15, to page 25, line 10, which is restated in the following for easier reference.

Due to the random nature of the considered problem and the random nature of ES and GA, we have repeated the simulation process for each algorithm and each test case for 30 times. However, for the sake of explanation, we present the test results of the first of the 30 simulation runs in Table 2. The test results consist of the good enough solution $\hat{x}$, the corresponding average overkills $E[V(\hat{x})]$ and retests $E[R(\hat{x})]$, the corresponding objective values $F(\bar{R}, \hat{x})$ obtained and the corresponding CPU times consumed by the two algorithms ($\mu + \lambda$)-ES+OO and GA+OO for the three cases $\bar{R}$ =30, 60 and 90. The consistent features of the test results revealed by Table 2 are described below. When $\bar{R}$ increases, the values of good enough $g_{W\min}$ for the corresponding algorithm increases as presented in row 2 of Table 2, and the corresponding values of leading good enough $n_{k\max}$, $k=8$, 9 and 10, which accounts for most of the retests, decrease as presented in rows 10, 11 and 12, respectively. This indicates that if we allow more retests by increasing $\bar{R}$, we can set more stringent threshold values by increasing $g_{W\min}$ and decreasing the leading $n_{k\max}$ to save more overkills, that is smaller $E[V(\hat{x})]$, as indicated in row 13 for the corresponding algorithm in Table 2.

As indicated in Remark 3 that the penalty function is arbitrarily designed to have a sharp jump when $E[R(x)] = \bar{R}$. Hence, the resulting $E[R(\hat{x})]$ in each case is less than or equal to $\bar{R}$ such that $P(E[R(x)] - \bar{R})$=0. Consequently, the obtained objective value $F(\bar{R}, \hat{x})$ is equal to the average overkills per wafer, $E[V(\hat{x})]$, in all cases.

Table 2. Good enough vectors of threshold values, average overkills, objective values and consumed CPU times of ($\mu + \lambda$)-ES+OO and GA+OO for three cases of $\bar{R}$ in the first simulation run.

Good enough solution $\hat{x}$ \ Methods	$\bar{R}$ =30		$\bar{R}$ =60		$\bar{R}$ =90	
	(μ+λ)-ES+OO	GA+OO	(μ+λ)-ES+OO	GA+OO	(μ+λ)-ES+OO	GA+OO
$g_{W\min}$	79	88	118	125	135	145
$n_{1\max}$	2	3	2	2	1	2
$n_{2\max}$	2	2	2	2	1	2
$n_{3\max}$	3	3	3	3	3	3
$n_{4\max}$	3	4	3	3	2	2
$n_{5\max}$	2	3	3	3	1	1
$n_{6\max}$	8	10	7	9	7	5
$n_{7\max}$	7	9	8	8	6	6
$n_{8\max}$	69	78	66	66	51	51
$n_{9\max}$	93	92	82	71	57	49
$n_{10\max}$	34	31	23	22	18	19
E[$V(\hat{x})$]	4.83	5.01	2.81	3.09	1.05	1.22
E[$R(\hat{x})$]	29.89	29.46	59.91	59.52	89.78	89.27
$F(\bar{R},\hat{x}) = E[V(\hat{x})] + P(E[R(\hat{x})] - \bar{R})$	4.83	5.01	2.81	3.09	1.05	1.22
CPU time (sec)	109.15	141.26	112.75	145.08	115.23	143.29

Table 2 also reveals that ($\mu + \lambda$)-ES+OO use less computing time and obtain better objective value than GA+OO. This fact not only exists in the first simulation run but also exists in the rest 29 runs as demonstrated in Table 3. Table 3 presents the mean, standard deviation of the objective value $F(\bar{R}, \hat{x})$ obtained by and the corresponding average CPU time consumed by the two algorithms for the three cases of different values of $\bar{R}$ of the 30 simulation runs. Both mean and standard deviation of $F(\bar{R}, \hat{x})$ obtained by the proposed (μ+λ)-ES+OO is smaller than that obtained by the GA+OO in all cases. Additionally, the average CPU times consumed by the proposed (μ+λ)-ES+OO is smaller than that consumed by GA+OO in each case. This demonstrates that ($\mu+\lambda$)-ES+OO can use less computing time to obtain better solution than GA+OO. Above all, the CPU times consumed by ($\mu+\lambda$)-ES+OO in all cases are within two minutes, which demonstrate the proposed algorithm is really suitable for *real-time application*.

Table 3. Comparisons of ($\mu+\lambda$)-ES+OO and GA+OO in three cases of $\bar{R}$ for 30 simulation runs.

Methods / $F(\bar{R},\hat{x})$	$\bar{R}$ =30		$\bar{R}$ =60		$\bar{R}$ =90	
	(μ+λ)-ES+OO	GA+OO	(μ+λ)-ES+OO	GA+OO	(μ+λ)-ES+OO	GA+OO
Mean	4.86	5.05	2.87	3.04	1.06	1.24
Standard deviation	0.08	0.10	0.08	0.09	0.04	0.05
Average CPU time (sec.)	110.76	142.85	111.32	144.66	113.41	143.57

(ii) To statistically test the performance of the proposed algorithm in real applications, we have compared the solution we obtained with 3000 randomly generated vectors of threshold values. The test results show the significance of the proposed algorithm. These additional simulations and the revisions appear on page 28, lines 1-14, which is restated in the following for easier reference.

Finally, in phase three, we use 590 real test wafers, whose bad dies b_{jk} and overkills v_{jk} before retest are known, to test the performance of the vector of threshold values $\hat{x}$ obtained by ($\mu+\lambda$)-ES+OO in the first simulation run of the three cases shown in Table 2. The test results of the pair ($\mathrm{E}[V(\hat{x})]$, $\mathrm{E}[R(\hat{x})]$) for these 590 test wafers with $\bar{R}=90$, $\bar{R}=60$ and $\bar{R}=30$ are shown in Fig. 7 by the points "○", "*", "△", respectively. We have also used 3000 randomly generated vectors of threshold values to test the same 590 test wafers. The resulting pairs of ($\mathrm{E}[V(\hat{x})], \mathrm{E}[R(\hat{x})]$) for these randomly generated threshold values are shown by the points "•" in Fig. 7. Since the two objectives, minimizing overkills and retests, have conflicting nature, the considered problem possesses Pareto optimal solutions, which are shown as the lower boundary of the region resulted from the randomly generated vectors of threshold values. From Fig. 7, we can observe that the pair ($\mathrm{E}[V(\hat{x})]$, $\mathrm{E}[R(\hat{x})]$) resulting from the solution obtained by the proposed ($\mu+\lambda$)-ES+OO for the three cases $\bar{R}=90$, $\bar{R}=60$ and $\bar{R}=30$ are all on the lower boundary. This implies that our algorithm not only controls the level of retests but also obtain a near Pareto optimal solution.

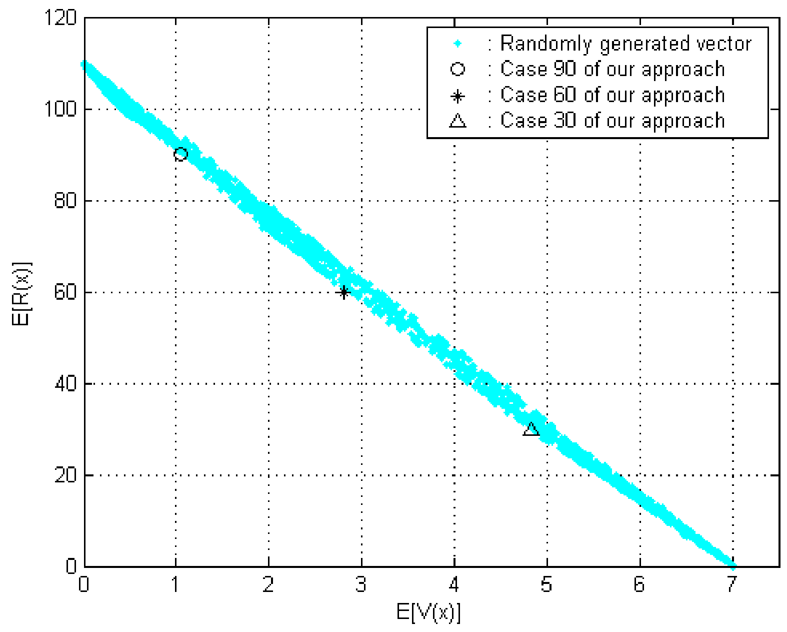

Fig. 7. The pairs of ($\mathrm{E}[V(\hat{x})]$, $\mathrm{E}[R(\hat{x})]$) resulting from the solution obtained by ($\mu+\lambda$)-ES+OO and the 3000 randomly generated vectors of threshold values.

Comment R5.3: Another problematic point is the selection of the parameter values. They could be too specific for the given problem, and one could argue that your methodology is difficult to apply to other problems (several parameters are defined). Moreover, the differences observed between the methods could be a result of this selection. Could you please provide some comments about how to select the parameter values? Usually, evolutionary algorithms (EAs) are quite robust to these values, and maybe the values you provide are reasonable for almost all problems, but, since you only show the results of one given problem, this is difficult to check.

Revisions: We wish to thank reviewer #5 again for raising this important question and helpful suggestion. To address this comment, we have completely revised the corresponding paragraph appearing in page 22, line 11 to page 23, line 4, which is restated in the following for easier reference.

The parameters employed in ($\mu+\lambda$)-ES+OO and GA+OO are described below. For ($\mu+\lambda$)-ES+OO, the parameters required in Fig. 3 are: μ =1000, λ =2000, $\tau_0=\frac{1}{\sqrt{11}}$ and $\sigma_{\min}=0.05$. For GA+OO, the parameters, which were indicated in Section 3.3, are: $|Q|$=1000, p_c =0.7, p_m =0.02 and $i_{\max}=100$. In general, the larger the values of μ,

λ and $|Q|$, the better the quality of the obtained solution; however, at the expense of computing time. Therefore, the first concern regarding the selection of the values of the above parameters is the computing budget. In OO theory, the typical size of the representative set, N, is usually taken as 1000. It has been shown in [35] that N=1000 is sufficient to represent a huge search space. Thus, we set the size of population μ and $|Q|$ to be 1000 for ES and GA, respectively. Typical truncation ratios μ/λ in ($\mu+\lambda$)-ES for huge solution space are in the range from 1/7 to 1/2 [19]. In this work, λ is set to be two times of μ, i.e. λ=2μ. Typical values of p_c and p_m suggested by Chong and Zak in [36] are 0.7 and 0.02, respectively. For the sake of saving computing time, $|S|$ should not be too large; however, too small $|S|$ may miss some good solutions. We set $|S|=100$ for both ($\mu+\lambda$)-ES+OO and GA+OO, and the reason for this selection will be stated in a remark, Remark 7, to be presented at the end of phase one. Similar reasons apply to the selection of $\sigma_{\min}=0.05$ for ($\mu+\lambda$)-ES+OO and $i_{\max}=100$ for GA+OO.

Remark 7: To investigate the effect of the size $|S|$ of the good enough subset S on the quality of the obtained good enough solution and the computing time, tests were executed 30 simulation runs with various $|S|$ in ($\mu+\lambda$)-ES+OO for $\bar{R}=60$. Table 4 shows the statistic of objective value and average CPU time by ($\mu+\lambda$)-ES+OO for various $|S|$ of 30 simulation runs with $\bar{R}$=60. We found that when $|S|$ increases, the corresponding objective value decreases and the CPU time increases. Thus, the choice of $|S|$ should depend on the *available computing budget* of the application problem. Based on the results presented in Table 4, $|S|$=100 is a good choice for the current application problem.

Table 4. Relationship between various $|S|$ and the performance of ($\mu+\lambda$)-ES+OO for $\bar{R}$=60 in 30 simulation runs.

$\lvert S\rvert$		1000	500	100	50	25
(μ+λ)-ES+OO with $\bar{R}$=60	Mean	2.84	2.85	2.87	2.92	2.95
	Standard deviation	0.07	0.07	0.08	0.13	0.15
	Average CPU time (sec.)	410.38	247.49	111.32	97.65	89.51

Comment R5.4: Why do you select these R values for the comparisons (R=30, R=60 and R=90)?

Revisions: To address this comment, we have added two statements on page 19, lines 17-18, and page 22, lines 8-10, which are restated in the following for easier reference.

Since the yield rate of the tested wafer product is 46.5%, the average upper limit of retests is 206*(1-0.465)=110.

As indicated previously, $\bar{R}$ should not exceed 110. Consequently, to uniformly select three different values of $\bar{R}$ as test cases, we set $\bar{R}$ = 30, 60 and 90.

Comment R5.5: One surprising fact is that the objective value and the average overkills are similar (at least in Table 2). Does this happen with the rest of experiments? Your penalty function is maybe too sharp.

Revisions: To address this comment, we have added a remark, Remark 3, on page 12, lines 18-21, and a paragraph on page 23, lines 20-23. These revisions are restated in the following for easier reference.

> **Remark 3**: $E[R(x)]$ is a critical value in wafer manufacturing. As indicated previously, larger $E[R(x)]$ implies that the quality of the fabrication process is skeptical. Therefore, $\bar{R}$ is a critical input parameter provided by the manager, and a sharp jump of the penalty function is designed to ensure that $E[R(x)]$ will not exceed $\bar{R}$.
>
> As indicated in Remark 3 that the penalty function is arbitrarily designed to have a sharp jump when $E[R(x)] = \bar{R}$. Hence, the resulting $E[R(\hat{x})]$ in each case is less than or equal to $\bar{R}$ such that $P(E[R(x)] - \bar{R})$=0. Consequently, the obtained objective value $F(\bar{R}, \hat{x})$ is equal to the average overkills per wafer, $E[V(\hat{x})]$, in all cases.

Comment R5.6: In my humble opinion (I am not a native English speaker), the paper should be carefully revised with respect to the English quality. Some sentences sound strange to me, and some of them are too long.

Revisions: We wish to thank reviewer #5 for reminding us to improve the written presentation. We have asked our colleague who is a native English speaker to edit our revised manuscript. Hopefully, the written presentation is improved in the revised manuscript.

Comment R6.1: Using an artificial neural network (ANN) to construct a surrogate model should be more justified and the model of ANN used should be better described, justifying (i) why, in this case, you consider 15 hidden nodes and (ii) why you apply one model with two outputs and not two models with one output each one.

Revisions: We wish to thank reviewer #6 for pointing out what need to be cleared in our paper. To address the issue raised in (i), we have added a remark, Remark 6, on page 21, lines 6-14, which are restated in the following for easier reference.

> **Remark 6**: The selection of the best number of neurons in the hidden layer depends on many factors. The size of the training set, amount of noise in the targets, complexity of the sought function to be modeled, type of activation functions used and the training algorithm all have interacting effects on the sizes of the hidden layers. There is no analytical method for determining the best number of neurons in the hidden layer. Therefore, we select the number of neurons in the hidden layer based on empirical tests. Fig. 5 shows the mean square errors (MSE), which is computed by (6), of various number of neurons in the hidden layer ranging from 1 to 25. From this figure, we can observe that 15 is the least number of neurons in the hidden layer that will achieve the smallest MSE.

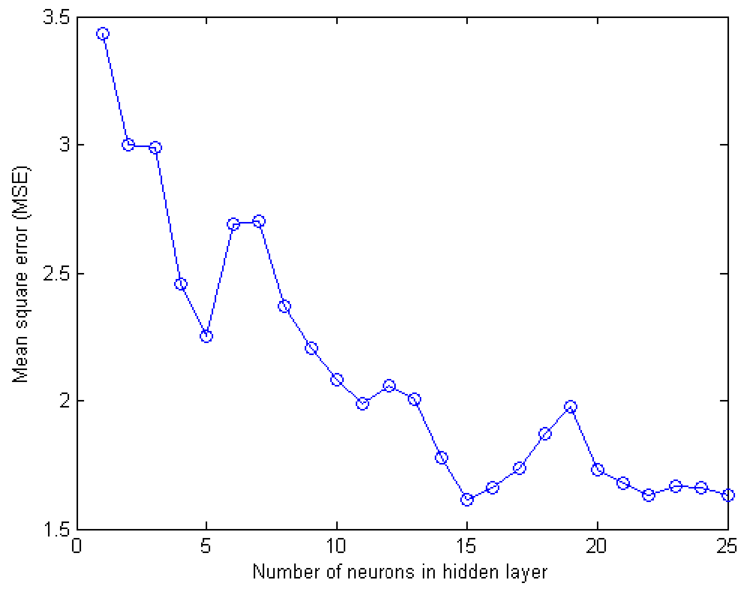

Fig. 5. The MSE of various numbers of neurons in hidden layer.

To address the issue raised in (ii), we have also added a remark, Remark 5, in page 19, lines 12-15. This remark is restated in the following for easier reference.

Remark 5: One can also use two ANNs of the same structure as that shown in Fig. 4. However, one ANN has the output as $E[R(x)]$, and the other has the output as $E[V(x)]$. Then a simple calculation is needed to obtain $F(\bar{R}, x)$. Such a two-ANN structure is more complicated than that shown in Fig. 4.

Comment R6.2: (i) Regarding the training algorithms used then assisted by the ANN, the GA or the ES, it is not justified the procedure to obtain the values of the different parameters: for the GA, mu=1000, pc =0.7, pm =0.02 and i_max >=100; and for (mu+lambda)-ES+OO, mu=1000, lambda=2000, and sigma_min=0.05. (ii) Do you apply a cross-validation process over the training set? If so, how do you divide the set of decision variable vector x's from the discrete solution space.

Revisions: We wish to thank reviewer #6 again for this important comment. (i) As suggested, we have completely revised the corresponding paragraph appearing in page 22, line 11 to page 23, line 4. Please refer to the revisions responding to comment R5.3.

(ii) No, we did not apply the typical cross-validation process over the training set. Instead, we provide the modeling error analysis of the trained ANN. First, we uniformly select M(=16641) points from the reduced solution space to train the ANN. (Note: the value of M

is justified in Remark 1 in page 7, lines 6-8.) The training process is described in Section 3.4 starting from line 16 page 19 to line 5 page 21. Second, we also use M' (=16641) points, which are uniformly selected to construct the OPC of the considered problem (Section 4.1) and different from the M points used to train the ANN, to proceed with the modeling error analysis of ANNBSM. The modeling error analysis is presented in Section 4.2 staring from line 2 page 31 to last line page 32.

Comment R6.3: With respect to the results in Table 3 associated to the 30 executions of the algorithms, apart from showing statistics of the objective value for 30 simulation runs of case R=60, you should apply some test of average comparison to show that significant differences are found. In other case, the sentence "significantly outperforms the two comparing methods " does not make any sense.

Revisions: We wish to thank reviewer #6 for pointing out our inadequate conclusion based on Table 3. In the revised manuscript, we have eliminated this inadequate statement. Additionally, owing to this comment as well as comment R2.5, we have restructured the part of test results and comparisons in Section 3.4. We believe the revised test results and comparisons staring from line 1 page 22 to line 14 page 28 will make more sense. Please also refer to the revisions responding to comments R2.5 and R5.2.

Comment R6.4: (i) On the other hand, when the method is compared against other methodologies, it is a bit risky to use only your own algorithms previously published [10], it has two main disadvantages: if the new version is slightly better than the previous one (as it seems), one could think that we send a little change of a paper every now and then to improve a bit our results, and this can be showing a lack of novelty, and also, if we don't compare against other proposals we could never now the real potential of our approximation. This is reason why it is absolutely necessary that the authors compare their results against other methods that solve the same problem, and even (ii) they should apply their methodology with other different meta-heuristics, apart from evolutionary algorithms, and (iii) also other regression models different from ANNs.

Revisions: We wish to thank reviewer #6 for this valuable comment. However, reviewer #6 probably overlooks that one of the contributions of our paper is proposing a systematic performance analysis procedure for ES +OO. This point has been emphasized in the last statement of the abstract and in the section of introduction appearing in page 4, lines 3-11 counted from the bottom. Above all, Section 4, performance analysis, should not be ignored. (i) Inspired by this comment, we have restructured the part of test results and comparisons, which is also indicated in the revisions responding to comments R2.5, R5.2 and R6.3. In the revised manuscript, we have divided the tests and comparisons into three phases. The additional simulations in phase three demonstrate that the resulting pair ($\mathrm{E}[V(\hat{x})]$, $\mathrm{E}[R(\hat{x})]$) obtained by the proposed ($\mu+\lambda$)-ES+OO is approximately a Pareto optimal solution. (ii) Additionally, in phase two, we have also use PSO associated with the exact model to solve the same problem, and we found that ($\mu+\lambda$)-ES outperforms PSO in the considered case. The facts stated in (i) and (ii) are presented in the revised test results and comparisons starting from line 7 page 22 to line 10 page 27. (iii) To justify the utilization of ANN as the surrogate model, we have added a paragraph in Section 2.2.1 appearing in page 6 lines 6-17, which is restated in the following for easier reference.

> There are various techniques to approximate the relationships between the inputs and outputs of a system. Polynomial regression, support vector regression (SVR), artificial neural network (ANN), radial basis function (RBF) and Kriging model are commonly used [23]-[24]. Bhattacharya pointed out in [25] that ANN is particularly suitable in modeling high-dimensional and highly nonlinear problems, because of its ability to learn and generalize from data, its nonlinear processing nature, and its massively parallel structure. In [26], Fonseca *et al.* utilized ANN as the metamodel for stochastic simulation. The combinatorial stochastic simulation optimization problem to be tackled in this paper is high-dimensional and highly nonlinear. The studies in both [25] and [26] reveal that the ANN is a suitable surrogate model for the considered problem. Therefore, we will use ANN as the surrogate model in this paper, because it is competent for approximating both highly nonlinear continuous-variable function and the input-output relationship of stochastic discrete event simulated systems [27].

Finally, we wish to thank the Managing Editor, Prof. Gang Kou, and the six reviewers again for their valuable comments and suggestions.