Harmony Search augmented with Optimal Computing Budget Allocation Capabilities for Noisy Optimization

George Georgoulas, Petros Karvelis, Giorgio Iacobellis, Valentina Boschian, Maria Pia Fanti, Walter Ukovich and Chrysostomos D. Stylios

***Abstract*—In this work we introduce the combinatory use of Harmony Search (HS) with Optimal Computing Budget Allocation (OCBA) as a means to tackle noisy optimization situations as those that occur during the execution of Discrete Event Systems (DES) for modeling complex systems. The OCBA procedure is employed for the exclusion of the worst harmony during the memory updating process in order to minimize the computational cost and at the same time retain a pool of promising solutions. The proposed hybrid approach is tested on real valued test functions as a proof of concept and the results are promising in case of small computational budgets.**

***Index Terms*— Budget Allocation, Harmony Search, Noisy Optimization, Optimal Computing, Simulation Optimization**

I. Introduction

Noisy optimization problems arise in many real life application domains. One of the most common problems is the attempt to optimize a performance index for a complex procedure that is captured through a simulation model, giving rise to what is known as simulation-optimization [1]-[4]. In the field of logistic systems and operations research, Discrete Event Systems (DES) are among the most prominent members for simulating complex logistic systems. Their main drawback is that in most cases their execution is quite slow combined with the need to run multiple replications for a specific design due to "simulation noise".

The simulation approach has been adopted by many researchers especially over the past decade due to the availability of modern simulation software and ample computational power [5]. Simulation can assist in either a static or a dynamic analysis of a system. A dynamic analysis is enhanced with software that shows the time-sequenced operation of the system that is being predicted or analyzed. Simulation is by its nature a descriptive tool that can be used for both prediction and exploration of the behavior of a specific system. A complex simulation embedded in a model-driven Decision Support System (DSS) can help a decision maker plan activities, anticipate the effects of specific resource allocations and assess the consequences of actions and events [6]. "What if analysis" performed via a simulation tool can sometimes offer extra confidence to the decision maker compared to just the simple presentation of numbers in a tabular format. However, detailed simulations are time consuming and can be computational heavy when a large number of alternatives are considered. Therefore simulation optimization is usually employed for tactical or strategic planning using some kind of metaheuristic process to guide the search.

In this paper we present for the first time the combination of two well-known procedures, each one suitable for tackling a different aspect of the simulation optimization problem, namely the Harmony Search (HS) and the Optimal Computing Budget Allocation (OCBA) procedures. It is known that HS can tackle the problem of efficiently searching the solution space, while the OCBA tackles the problem of effectively allocating the, usually restricted, available number of replications. Thus their combinatory usage may lead to a new advanced methodology.

The rest of the paper is structured as follows: Section II presents a brief introduction to simulation optimization aspects. Sections III and IV describe the essentials of HS and OCBA. Section V presents the proposed approach and Section VI summarizes some preliminary results. Section VII concludes the paper offering some hints for future work.

II. Simulation Optimization

The primary reason why simulation optimization is difficult to apply is the stochastic nature of evaluating the objective function. There is a basic trade-off between devoting computational effort on searching the space for new candidate solutions (exploration) versus getting more accurate estimates of the objective function as there are some currently promising solutions (exploitation). In other words, we have to investigate how much of the simulation budget should be allocated to additional replications at already visited points and how much to allocate for replications at newly generated points.

In mathematical terms, adopting the notation of [1], we have to find a set of values θ that minimizes (maximizes) an expression (the expression sometimes represents just one

G. Georgoulas, P. Karvelis and C.D. Stylios are with the Knowledge and Intelligent Computing (KIC) Laboratory of the Department of Informatics Engineering, Technological Educational Institute of Epirus, 47100, Arta, Greece; (e-mail: {georgoul; karbelis}@kic.teiep.gr and stylios@teiep.gr)

G. Iacobellis and M.P. Fanti are with the Department of Electrical and Electronic Engineering, Polytechnic of Bari, Bari Italy; (e-mail: {fanti;iacobellis}@deemail.poliba.it).

V. Boschian and W. Ukovich are with the Department of Engineering and Architecture, University of Trieste, Trieste, Italy; (mail: valentina.boschian@di3.units.it; ukovich@units.it)

Key Performance Indicator (KPI) of the problem or it can be a function including more than one KPIs)

$$\min_{\theta\in\Theta} J(\theta) \tag{1}$$

Where $\theta \in \Theta$ represents the (vector of) input variables, $J(\theta)$ is a (scalar) objective function, and Θ is the constraints set, which may be either explicitly given or implicitly defined.

The assumption in the simulation optimization setting is that $J(\theta)$ is not available directly, but must be estimated through simulation, e.g., the simulation output provides $\hat{J}(\theta) = L(\theta,\omega)$, a noisy estimate of $J(\theta)$. The most common form for $J(\theta)$ is an expectation, e.g.

$$J(\theta) = E\left[L(\theta,\omega)\right] \tag{2}$$

where ω represents a sample path (simulation replication), L is the sample performance measure [1] and $E[\cdot]$ is the expectation operator. Although this form is fairly general (includes probabilities by using indicator functions), it does exclude certain types of performance measures such as the median (and other quantiles) and the mode.

Lately, metaheuristic approaches have become more and more popular in the settings of simulation-optimization augmented with some kind of mechanism to allocate the available computational budget among the various alternatives [6]-[10]. In this paper, the HS represents the metaheuristc part and the mechanism for allocating the available replications is OCBA, which are presented in the following sections.

III. Harmony Search

Harmony search is a relatively new metaheuristic method that was introduced in an attempt to simulate the improvising process [11] of a music band in search of an improved harmony/melody. The original formulation considered integer variables but newer developments have been proposed also for real [12], [13] as well as binary variables [14]. The way that each variable is treated also makes it suitable for problems with mixed types of variables.

As almost all metaheuristic approaches, HS consists of a number of stages, such as memory consideration, pitch adjustment and random selection, but also it includes problem-specific features for every application. In this work, we are not exploiting all the possible alternatives of the HS but we just use the basic steps of the method that are described in the rest of this section.

A. Harmony memory initialization

The basic component on HS is a pool of promising solutions that are stored in harmony memory (HM). Therefore, before the application of each one of the aforementioned steps we have to generate randomly (or alternatively some could be provided by the user based on expert knowledge or even intuition) and store a number of initial harmonies in the HM:

$$\mathbf{HM} = \left[\begin{array}{cccc|c} D_1^1 & D_2^1 & \cdots & D_n^1 & f(\mathbf{D}^1) \\ D_1^2 & D_2^2 & \cdots & D_n^2 & f(\mathbf{D}^2) \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ D_1^{HMS} & D_2^{HMS} & \cdots & D_n^{HMS} & f(\mathbf{D}^{HMS}) \end{array}\right] \tag{3}$$

where D_i^j is the *i*-th decision variable in the *j*-th solution vector, which for the original integer formulation has one discrete value out of a candidate set $\{D_i(1), D_i(2), \ldots, D_i(k), \ldots, D_i(K_i)\}$; $f(\mathbf{D}^j)$ is the objective function value for the *j*-th solution vector, and HMS is the harmony memory size (i.e. the number of solution vectors stored in the HM). The number of random harmonies should be at least HMS. However, the number can be more than HMS, such as twice or three times as many as HMS. Then, top-HMS harmonies are selected as starting vectors [15]. In the case of simulation optimization the inclusion of this step should be weighed against the extra replications needed.

B. Improvisation of a new harmony

Using the vectors stored in the HM, new solution vectors are produced based on the application of three operations:

- Random selection
- Memory consideration and
- Pitch adjustment

During the random selection process, as its name implies, a new value is chosen randomly out of a candidate set with a probability (1-HMCR) (see next paragraph for the definition of HMCR), as a direct analogy to a musician playing any possible pitch within the range of its musical instrument

$$D_i^{New} \leftarrow D_i(k), D_i(k) \in \{D_i(1), D_i(2), \ldots, D_i(K_i)\} \tag{4}$$

In memory consideration, we resort to the already stored harmonies within the HM, picking a value that is already there, with a probability equal to the harmony memory considering rate (HMCR), as a musician plays any pitch of the preferred pitches stored in his memory

$$D_i^{New} \leftarrow D_i(l), D_i(l) \in \{D_i^1, D_i^2, \ldots, D_i^{HMS}\} \tag{5}$$

In pitch adjustment a value that has been selected in the previous step of memory consideration is altered further a bit, turning into one of its neighboring values with a probability equal to the pitch adjusting rate (PAR)

$$D_i^{New} \leftarrow D_i(l \pm 1), D_i(l) \in \{D_i^1, D_i^2, \ldots, D_i^{HMS}\} \tag{6}$$

In the case of constrained optimization problems, if the newly improvised harmony $\mathbf{D}^{New}$ violates any of the constraints, HS abandons it or still keeps it by adding a penalty to the objective function value, just like musicians sometimes still accept a rule-violating harmony.

C. Update of harmony memory

After the generation of a new vector $(D_1^{New}, D_2^{New}, \ldots, D_n^{New})$ a selection process takes place: if the new harmony is better than the worst vector in HM with respect to the objective function, the former takes the place of the latter.

Note: For the diversity of harmonies in HM, other harmonies (in terms of least-similarity) can be considered.

Also, maximum number of identical harmonies in HM can be considered in order to prevent premature convergence.

After the HM update and if the maximum number of iterations or a desired performance has not been reached, the algorithm continues to generate new harmonies.

D. Harmony search for real valued variables

In the case of real valued variables what changes is the random selection and the pitch adjustment while the memory consideration remains the same.

In the random selection we now select a value within the admissible range of values for the corresponding variable

$$D_i^{New} \leftarrow Range_i \cdot U(0,1) \tag{7}$$

Where $Range_i$ is the range of values of variable i and $U(0,1)$ is a uniform random generator between 0 and 1, meaning that we sample uniformly within the acceptable range of values.

In the pitch adjustment the new value is produced by randomly sampling around the value that was selected in the memory consideration step, by:

$$\begin{gathered} D_i^{New} \leftarrow D_i(l) + bw_i\left(2 \cdot U(0,1) - 1\right), \\ D_i(l) \in \left\{D_i^1, \ldots, D_i^{HMS}\right\} \end{gathered} \tag{8}$$

where bw_i is an arbitrary distance bandwidth for the continuous design variable

IV. Optimal Computing Budget Allocation

Optimal Computing Budget Allocation was originally proposed by Chen [16] as a procedure to optimally allocate a predefined number of trials/replications in order to maximize the probability of selecting the best system/design. Since its introduction many variants have been proposed [17], [18] and while at the beginning it was meant to deal with a predefined number of designs, lately it has been coupled with (global) search algorithms to deal with large scale optimization problems both for discrete and continuous search spaces [19]-[25].

The main philosophy of OCBA dictates: Allocate replications not only based on the variance of the different designs but also taking into account the respective means.

According to OCBA if we have a total budget of T replications, $T = \sum_{i=1}^{k} N_i$ then we try to (asymptotically) maximize the probability of Correct Selection $P\{CS\}$ (the probability of actually selecting the best b among the k designs) [18] (for the case of a minimization problems):

$$P\{CS\} = P\left\{\bigcap_{i=1, i\neq b}^{k} \left(J_b < J_i\right)\right\} \tag{9}$$

As it was pointed out in Section II we can only have estimates of J_i using the sample mean based on simulation outputs. Even using estimates the $P\{CS\}$ itself, for the general case, can be only estimated resorting to a Monte Carlo simulation procedure which is time consuming. Therefore instead of estimating the $P\{CS\}$ one resorts to a much easier to compute lower bound which is called the Approximate Probability of Correct Selection (APCS).

APCS comes in two forms [18]:

APCS-B which is in a summation form derived using the Bonferroni inequality:

$$APCS - B \equiv 1 - \sum_{i=1, i\neq b}^{k} P\left\{\bar{J}_b > \bar{J}_i\right\} \leq P\{CS\} \tag{10}$$

APCS-P which comes as a product form:

$$APCS - B \equiv \prod_{i=1, i\neq b}^{k} P\left\{\bar{J}_b < \bar{J}_i\right\} \leq P\{CS\} \tag{11}$$

For a more thorough treatment of the subject the interested reader is referred to the original publication [16]. Note: In Chen's formulation normal distributions are assumed for the estimation of the APCS.

In order to (asymptotically) maximize APCS and as a result $P\{CS\}$ the following relationship between two non-best designs $\left(N_i, N_j\right)$ should hold

$$\frac{N_i}{N_j} = \left(\frac{\sigma_i / \delta_{b,i}}{\sigma_j / \delta_{b,j}}\right)^2, \text{ for all } i \neq j \neq b \tag{12}$$

and the number of simulation replications for the best design is given as

$$N_b = \sigma_b \sqrt{\sum_{i=1, i\neq b}^{k} \frac{N_i^2}{\sigma_i^2}} \tag{13}$$

where, μ_i, σ_i, are the mean and standard deviation of the i-th design, $\delta_{b,i} = \mu_i - \mu_b$, and b is the best design.

The above equations show that the noisier the simulation output (larger variance), the more replications are allocated. More replications are also given to the design of which the mean is closer to that of the best design.

The most common implementation of the OCBA is based on a sequential approach that allocates Δ extra replications per step. The procedure is summarized as follows [18]:

OCBA Procedure:

Input: k, P^*, Δ, n_0 ($n_0 \geq 5$)

Initialize: $l \leftarrow 0$

Perform n_0 simulation replications for all designs, $N_1^l = N_2^l = \ldots = N_k^l = n_0$

Loop.

Update: Calculate sample means $\hat{\mu}_i$ and sample standard deviations $S_i\left(N_i^l\right)$, $i=1,2,\ldots,k$ using the new simulation output. Find $b = \arg\min_i \hat{\mu}_i$; calculate APCS

Check: If $APCS \geq P^*$ stop; Otherwise, continue

Allocate: Increase the computing budget by Δ and calculate the new budget allocation $N_1^{l+1}, N_2^{l+1}, \ldots, N_k^{l+1}$, according to

$$\frac{N_i^{l+1}}{N_j^{l+1}} = \left(\frac{S_i(N_i^l)\left(\hat{\mu}_j - \hat{\mu}_b\right)}{S_j(N_j^l)\left(\hat{\mu}_i - \hat{\mu}_b\right)} \right)^2 \text{, for all } i \neq j \neq b \text{ and}$$

$$N_b^{l+1} = S_b(N_b^l)\sqrt{\sum_{i=1, i\neq b}^{k} \left(\frac{N_i^{l+1}}{S_i(N_i^l)} \right)^2}$$

Simulate: Perform additional max($N_i^{l+1} - N_i^l$,0) simulations for design i, $i=1,2,\ldots,k$;
$l \leftarrow l+1$

End of loop

We must note that for practical implementations the allocation of replications is terminated at some point even without reaching the predefined confidence level. This is especially useful in case of large spaces and/or expensive simulation runs. In the extreme case $\Delta = 1$ and one replication is allocated at a time.

V. HS-OCBA

In the HS procedure during the step of memory update a decision has to be made on whether the new harmony will be included in the HM substituting the worst member of the memory or not.

Therefore the OCBA procedure can be used to find the worst among the HMS+1 harmonies (where HMS is the size of the memory). Moreover towards the end of the search, OCBA can be used to select the best among the members of the HM. Therefore in general if we are dealing with a minimization problem, during the memory update process we apply the OCBA procedure as described in section III using $-\hat{\mu}_i$ instead of $\hat{\mu}_i$ (turning a maximization problem into a minimization one) while during the final selection of the best solution we apply the OCBA procedure using $\hat{\mu}_i$.

VI. Results

In this preliminary study we experimented mainly with scenarios having a rather "small" number of replications since our approach aims to be used primarily as part of simulation optimization solution where each simulation is "expensive" in terms of computational time. The OCBA loop was terminated either if the number of replications was consumed or if the APCS exceeded a predefined threshold. The parameters used are summarized in the following Table I. We must note that at this stage no "optimal" selection of the HS parameters was sought.

For each one of the following test functions, Gaussian noise was added with zero mean and variance equal to one:

$$g(x) = f(x) + N(0,1) \qquad (14)$$

In this study the dimension d was set equal to 2 and the search space in each dimension to [-5,5] for all test functions. For each test function the experiments were repeated 20 times and the results were averaged. Three commonly used test functions were involved and the results are depicted in Figures 1 to 3. For the case of equal allocation scheme the replications were also allocated one by one and the procedure was stopped either if the number of replications was exceeded or the APCS exceeded the predefined threshold.

TABLE I
Parameter Settings

Parameter name	*Parameter Value*
Total number of replications *T-total*	5000
number of replications T	100
Initial number of replications n_0	5
Δ	1
APCS threshold	0.8
HMCR	0.9
PAR	0.3
bw_i	$Range_i$

A. Rosenbrock

$$f(x) = \sum_{i=1}^{d-1}\left(100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right) \qquad (14)$$

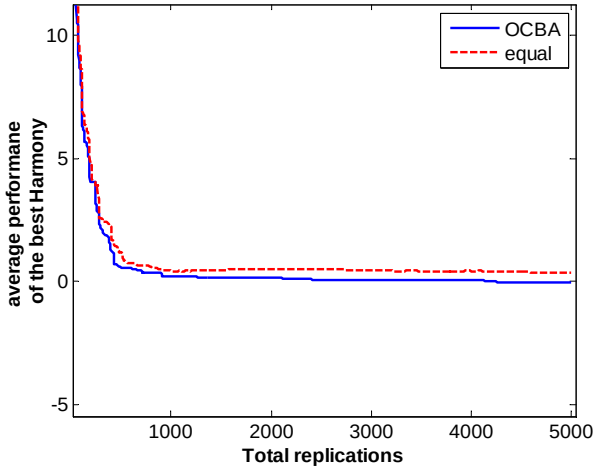

Fig. 1. Average performance of the best Harmony for the case of noisy Rosenbrokc function

B. Rastrigin

$$f(x) = 10d + \sum_{i=1}^{d}\left(x_i^2 - 10\cos\left(2\pi x_i\right)\right) \qquad (15)$$

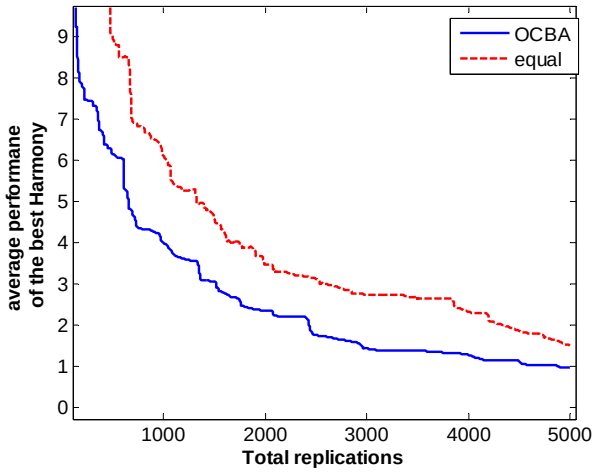

Fig. 2. Average performance of the best Harmony for the case of noisy Rastrigin function

C. Ackley

$$f(x) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) \quad (15)$$

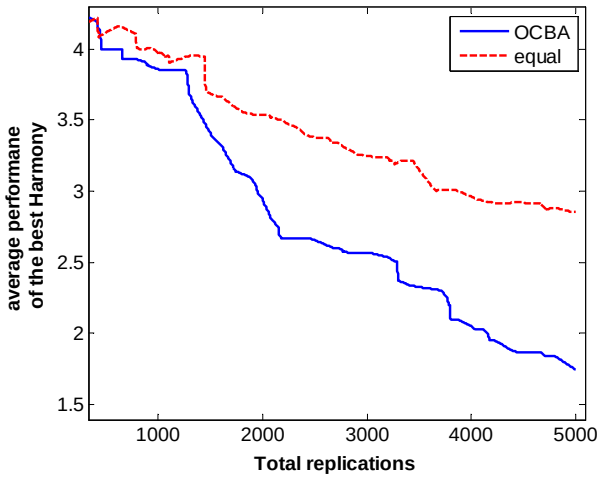

Fig. 3. Average performance of the best Harmony for the case of noisy Ackley function

VII. Conclusions

In this work, we present a scheme for integrating a popular metaheuristic algorithm with a mechanism for allocating replications under the presence of uncertainty. Harmony Search is augmented with Optimal Computing Budget Allocation, so as to be able to tackle noisy optimization problems that arise frequently in real life applications.

Our preliminary results that are presented here, suggest that for a fairly small computational budget and moderate levels of noise the proposed scheme seems to be more effective compared to a scheme that blindly allocates equal budget to all candidate solutions. To be more specific, by implementing OCBA, more alternatives were investigated within a given computational budget, letting further exploration of the search space. However more tests need to take place before reaching overall conclusions regarding the settings at which the OCBA procedure can outperform with certainty the simpler equal-allocation approach.

One of our preliminary observations also suggests that as the budget increases the difference between the two approaches becomes smaller and at some point is negligible. Therefore the HS-OCBA scheme should probably be reserved for situations where only few and time consuming replications are available as in the case of DES application. Our ongoing research is going to include a more elaborate set of experiments involving a real life DES model.

During our experiments, we found out that in some cases we came across a situation where less promising solutions were "accidentally" saved in the HM. This occurred especially under a setting with higher noise levels and small number of initial replications. Under these circumstances a solution could be assigned a better performance value which is in fact an anomaly/outlier and since the comparison that takes place seeks for the worst solution, the solution with the "mistakenly" perceived better performance value can continue being held in the HS memory only to be revealed that it is actually inferior during the final OCBA stage that seeks for the best among the solutions stored in the memory. This final stage helps us to more accurately estimate the value of the best solution but is not able to prevent the situation just described.

Therefore in future work, we will investigate whether such a stage or a stage of equal sampling applied at some point during the search process and not just during the final stage could be beneficial for the overall efficiency of the algorithm.

Moreover in future work we will also test the use of an indifference zone formulation [26], in other words a formulation that will stop sampling if all competitors for the best/worst are "good enough" in order to avoid unnecessary replications just to achieve a trivially better solution. Such a mechanism can be of great use within a simulation-optimization framework.

Acknowledgment

This work was supported by the E.U. FP7–PEOPLE–IAPP–2009, Grant Agreement No. 251589, Acronym: SAIL.

References

[1] M. Fu, C. Chen, and L. Shi, "Some topics for simulation optimization" In *Winter Simulation Conference*, pp 27–38, 2008.

[2] J. April, F. Glover, J. P. Kelly, and M. Laguna, "Practical introduction to simulation optimization" In *Winter Simulation Conference*, vol. 1, pp. 71-78, 2003.

[3] S. Olafsson and J. Kim. Simulation optimization. In E. Yucesan, C.-H. Chen, J. Snowdon, and J. Charnes, editors, *Winter Simulation Conference*, 2002.

[4] M. C. Fu, F. W. Glover, and J. April., Simulation Optimization: A Review, New Developments, and Applications, in. *Winter Simulation Conference*, pp. 83-95, 2005.

[5] G. Giani, G. Laporte and R. Musmanno, *Introduction to Logistics Systems Planning and Control*, Willey, 2004

[6] D. J. Power, and R. Sharda. Model-driven decision support systems: Concepts and research directions. *Decision Support Systems* vol. 43, no. 3, pp. 1044-1061, 2007.

[7] F. Glover, F., Kelly, J.P., & Laguna, M. "The OptQuest approach to Crystal Ball simulation optimization". *Decisioneering White Paper.(http://www. decisioneering. com/articles/article_index. html),* 2000.

[8] J. Xu, B. L., Nelson, and J.E.F.F. Hong, "Industrial strength COMPASS: A comprehensive algorithm and software for optimization via simulation", *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 20, no. 1, pp 3.1-3:29, 2010.

[9] M. C. Fu, J. Hu, and S. I. Marcus, "Model-based randomized methods for global optimization" In *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems*, pp. 355-363, 2006.

[10] G. S. Piperagkas, G. Georgoulas, K. E. Parsopoulos, C. D. Stylios, and A. C. Likas "Integrating particle swarm optimization with reinforcement learning in noisy problems" *In Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, pp. 65-72, 2012.

[11] Z. W., Geem, J.-H. Kim, and, G. V. Loganathan, "A new heuristic optimization algorithm: harmony search", *Simulation,* vol. 76, no 2, pp. 60– 68, 2001.

[12] K. S. Lee, and Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and

practice" *Computer Methods in Applied Mechanics and Engineering,* vol. 194, no 36–38, pp. 3902–3933, 2005.

[13] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*, 2nd ed, Luniver Publisher, 2010.

[14] L. Wang, Y. Mao, Q. Niu and M. Fei "A Multi-Objective Binary Harmony Search Algorithm" *Advances in Swarm Intelligence Lecture Notes in Computer Science,* vol. 6729, pp. 74-81, 2011.

[15] S. O. Degertekin, "Optimum design of steel frames using harmony search algorithm" *Structural and Multidisciplinary Optimization,* vol. 36, no. 4, pp. 393-401, 2008.

[16] C. H. Chen, "A lower bound for the correct subset-selection probability and its application to discrete-event system simulations". *IEEE Transactions on Automatic Control,* vol. 41*,* no. 8, pp. 1227-1231, 1996.

[17] C. H. Chen, J. Lin, E. Yücesan, and S. E. Chick, "Simulation Budget Allocation for Further Enhancing the Efficiency of Ordinal Optimization," *Journal of Discrete Event Dynamic Systems: Theory and Applications* , vol. 10, pp. 251-270, July 2000.

[18] C .H. Chen, and L. H. and Lee, *Stochastic Simulation Optimization: An Optimal Computing Budget Allocation*, World Scientific Publishing Co., 2010.

[19] T. Bartz-Beielstein, D., Blum, and J. Branke, "Particle swarm optimization and sequential sampling in noisy environments". In Hartl, R. and Doerner, K., editors, *Proceedings 6th Metaheuristics International Conference (MIC2005)*, Vienna, Austria, pp. 89-94, 2005.

[20] T. Bartz-Beielstein, D., Blum, and J. Branke, "Particle swarm optimization and sequential sampling in noisy environments". In *Metaheuristics* Springer US., pp. 261-273, 2007.

[21] S. C., Horng, F. Y., Yang, and S .S. Lin, "Applying PSO and OCBA to Minimize the Overkills and Re-Probes in Wafer Probe Testing". *IEEE Transactions on Semiconductor Manufacturing,* vol. *25,* no. 3, pp. 531-540, 2012.

[22] S. Zhang, P., Chen, L. H., Lee, C. E., Peng, and C. H. Chen, "Simulation optimization using the particle swarm optimization with optimal computing budget allocation". In *Winter Simulation Conference,* pp. 4303-4314, 2011.

[23] H. Pan, L., Wang, and B. Liu, "Particle swarm optimization for function optimization in noisy environment" *Applied Mathematics and Computation*, vol. 181*,* no. 2, pp. 908-919, 2006.

[24] B. Liu, X. Zhang, and H. Ma, "Hybrid differential evolution for noisy optimization". In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence),* pp. 587-592, 2008.

[25] C. Schmidt, J, Branke, and S. E. Chick, "Integrating techniques from statistical ranking into evolutionary algorithms". In *Applications of Evolutionary Computing* , pp. 752-763, 2006.

[26] J. Branke, S.E. Chick, and C. Schmidt, "New developments in ranking and selection: an empirical comparison of the three main approaches". In *Winter Simulation Conference,* pp. 708-717, 2005.