PRIFYSGOL
glyndŵr
UNIVERSITY

Glyndŵr University Research Online

Journal Article

An Efficient Evolutionary Algorithm for Chance-constrained Bi-objective Stochastic Optimization

Liu, B., Zhang, Q., Fernández, F. and Gielen, G.

This article is published by IEEE. The definitive version of this article is available at http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6449314&tag=1

Recommended citation:

Liu, B., Zhang, Q., Fernández, F. and Gielen, G. (2013), 'An Efficient Evolutionary Algorithm for Chance-constrained Bi-objective Stochastic Optimization', IEEE Transactions on Evolutionary Computation, Vol.17, No.6, pp.786-796. doi: 10.1109/TEVC.2013.2244898

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

An Efficient Evolutionary Algorithm for Chance-constrained Bi-objective Stochastic Optimization

Bo Liu, Qingfu Zhang, *Senior Member, IEEE*, Francisco V. Fernández and Georges Gielen, *Fellow, IEEE*

***Abstract*—In engineering design and manufacturing optimization, the trade-off between a quality performance metric and the probability to satisfy all the performance specifications (yield) of a product naturally leads to a chance-constrained bi-objective stochastic optimization problem (CBSOP). A new method, called MOOLP (multiobjective uncertain optimization with ordinal optimization (OO), Latin Supercube sampling (LSS) and parallel computation), is proposed in this paper for dealing with the CBSOP. This proposed method consists of a constraint satisfaction phase and an objective optimization phase. In its constraint satisfaction phase, by using the OO technique, an adequate number of samples are allocated to promising solutions, and the number of unnecessary MC simulations for non-critical solutions can be reduced. This can achieve more than five times speed enhancement compared to the application of using an equal number of samples for each candidate solution. In its MOEA/D-based objective optimization phase, by using LSS, more than five times speed enhancement can be achieved with the same estimation accuracy compared to primitive MC simulation. Parallel computation is also used for speedup. A real-world problem of the bi-objective variation-aware sizing for an analog integrated circuit is used in this paper as a practical application. The experiments clearly demonstrate the advantages of MOOLP.**

***Index Terms*—Multiobjective optimization, uncertain optimization, parameter uncertainty, chance constraint, MOEA/D, yield optimization, process variation**

I. Introduction

In the engineering design area, design parameters suffer from process and environmental variations, which are often unavoidable. Therefore, uncertainty must be addressed at the design phase. Yield is the percentage of manufactured products that meet all the specifications under the presence of the process variations. A designer not only has to consider the yield, but also has to balance the yield and some other quality metrics of the design. For example, in analog integrated circuit (IC) design [1], a designer often has to balance the yield and the power consumption. These applications naturally lead to the following chance-constrained bi-objective stochastic optimization problem (CBSOP):

$$\text{maximize } Y(x) = Pr(g_1(x,\xi) \geq 0, \ldots, g_k(x,\xi) \geq 0) \quad (1a)$$

$$\text{maximize } Q(x) = E[h(x,\xi)] \quad (1b)$$

$$\text{subject to } Y(x) \geq \theta, \quad (1c)$$

$$x \in [a,b]^d \quad (1d)$$

where

- x is the d-dimensional vector of design variables,
- $[a,b]^d$ is the search space,
- $\xi \in \Omega$ is a vector of random variables with zero mean values that models the manufacturing and environmental variations. If ξ is replaced by its mean value ($\xi = 0$), it implies that manufacturing and environmental variations are ignored,
- $g_i(x,\xi)$ ($i = 1,\ldots,k$) are k performance functions. A design meets the design requirements if $g_i(x,\xi) \geq 0$ for all i. When $\xi = 0$, the constraints become $g_i(x,0) \geq 0$, which correspond to the design constraints under nominal manufacturing and environmental conditions,
- $h(x,\xi)$ is the quality index function, which may be one of the $g_i(x,\xi)$ functions,
- $Y(x)$ is the yield function,
- $Q(x)$ is the expected performance of the quality index,
- θ is the yield threshold.

The chance constraint $Y(x) \geq \theta$ in (1) [2] is important since the design is useless in practice when the yield $Y(x)$ is lower than a certain threshold, even if the design has a good performance under nominal manufacturing and environmental conditions. Chance constraints have been widely used in many real-world applications such as mechanical engineering [3], electrical engineering, reliability engineering [4] and others [2], [5].

A Pareto optimal solution to (1) is a best candidate for balancing $Q(x)$ and $Y(x)$. Let x and x' be two feasible solutions to (1), x is said to dominate x' if and only if $Q(x) \geq Q(x')$, $Y(x) \geq Y(x')$, and at least one of these two inequalities is strict. A solution x^* is Pareto-optimal if there is no other solution that dominates it. The set of all the Pareto-optimal solutions is called the Pareto set (PS) and the image of PS in the objective space (i.e., $Y-Q$ space) is the Pareto front (PF). A decision maker often wants to have an approximate PF for gaining more understanding of the problem and making his / her final decision.

B. Liu is with Department of Computing, Glyndwr University, Wrexham, U.K. He was with Technische Universitaet Dortmund, Dortmund, Germany, and Katholieke Universiteit Leuven, Leuven, Belgium. G. Gielen is with ESAT-MICAS, Katholieke Universiteit Leuven, Leuven, Belgium (e-mail: b.liu@glyndwr.ac.uk, {Georges.Gielen, Bo.Liu}@esat.kuleuven.be, liu_bo765@yahoo.com.cn).

Q. Zhang is with the School of Computer Science and Electronic Engineering, University of Essex, Colchester, U.K. (e-mail: qzhang@essex.ac.uk).

F. Fernández is with IMSE, CSIC and University of Sevilla, Sevilla, Spain. (e-mail: Francisco.Fernandez@imse-cnm.csic.es).

Copyright (c) 2013 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.

Copyright (c) 2013 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

In the engineering design optimization field, most reported approaches for dealing with $Y(x)$ and $Q(x)$ are based on single objective optimization. These approaches optimize $Y(x)$ or $Q(x)$ under some constraints. Such approaches cannot provide trade-off information between $Y(x)$ and $Q(x)$, which is of large practical interest in industrial applications. Some preliminary work has been done to consider $Y(x)$ in a multiobjective design optimization problem (e.g. [6], [7]). In the evolutionary computation field, generic multiobjective optimization in uncertain environments has been studied [8]. Many works focus on correctly identifying the optimal solutions in uncertain environments. Some of them aimed at handling a high noise level in uncertain multiobjective optimization [9]. [8] also introduces robust design problems [10], which consider process variations into the multiobjective optimization. A key solution method [11] is proposed. It uses robust measure (e.g. worst case measure) to reflect the degree of variation to the objectives due to the process variations. The selection criterion adopted is randomly based on either the conventional Pareto ranking or the robust measure to make a balance between the Pareto-optimality and the robustness [8]. The goal is to generate a set of robust Pareto-optimal designs. Some operators for efficiency enhancement to solve the robust design problem are proposed. Our approach, on the other hand, focuses on efficiency improvement for CBSOP, which explicitly trade off the objective and yield (In CBSOP, $Y(x)$ is an explicit objective.) Due to the stochastic value of ξ, evaluations of $Y(x)$ and $Q(x)$ require Monte-Carlo (MC) simulations, which could be very time-consuming. In many industrial applications, a single evaluation takes from a few seconds to minutes. Although the simulation of each sample does not take too long, many samples are needed to evaluate the objectives and constraints, hence, this costs a much longer time. Thus, it is of great practical interest to study how to reduce the total number of Monte-Carlo (MC) simulations in approximating the PF for CBSOP. To the best of our knowledge, however, very little effort has been made along this direction. This paper attempts to approximate the Pareto front of (1) with as few MC simulations as possible, so as to considerably increase the efficiency for solving CBSOP.

In engineering applications, most approaches use primitive Monte-Carlo (PMC) simulation for each candidate solution to estimate the values of $Y(x)$ and $Q(x)$, and apply a conventional multiobjective evolutionary algorithm (MOEA) [6], [7]. To reduce the computational cost, some approaches use some design of experiments (DoE) methods, e.g. Latin Hypercube sampling (LHS), to replace PMC simulation [12]. Shuffled Complex Evolution Metropolis [13], which is a Markov chain Monte-Carlo (MCMC) method to perform the samplings in noisy multiobjective optimization, was first introduced in [14]. These approaches have some advantages but also have some limitations. Firstly, LHS often cannot provide a sufficient speed enhancement with respect to PMC [15]. Secondly, MCMC-based advanced sampling methods are often sensitive to some parameters or settings (e.g. start distribution), which require problem-specific knowledge [13], [15]. Convergence diagnostic is very critical in MCMC-based approaches, but there are few general and robust methods to determine how many steps are needed to converge to the stationary distribution within an acceptable error [16]. A few approaches use offline surrogate model to predict $Y(x)$ and $Q(x)$ [2], [17]. Such approaches work well for small-dimensional problems, but are often not applicable to high-dimensional problems, since preparing the training data is computationally very expensive.

This paper focuses on efficient solution methods for CBSOP, which is also applicable to other uncertain multiobjective optimization problems including MC simulation [2]. Three main contributions are made in this paper to efficiently solve the uncertain optimization problem of (1). Firstly, an advanced MC sampling method, Latin Supercube sampling (LSS) [18], is applied, which is more efficient than LHS, and is more general and robust than MCMC methods [15]. Secondly, the common practice of using the same number of samples for all the candidate solutions is not adopted when handling stochastic chance constraints in this paper. Instead, a new mechanism is proposed that uses ordinal optimization (OO) [19] to adaptively assign the number of samples to different candidate solutions. OO was originally proposed for efficiently selecting the best candidate among a group of candidate solutions with uncertain parameters. In this work, we use it in iteration-based optimization, whose goal is to identify and obtain reasonably good estimations of feasible or almost feasible solutions and to decrease the computational effort spent in non-critical candidates. Lastly, a two-phase optimization mechanism is proposed. Although OO is very promising for efficiently selecting or ranking good candidates, its accuracy often cannot satisfy the requirement for objective optimization. Our proposed two-phase approach consists of a constraint satisfaction phase and an optimization phase. OO is applied in the constraint satisfaction phase using a small number of LSS samples. The purpose of this phase is to provide a well-distributed feasible initial population to the optimization phase. In the optimization phase, a large number of LSS samples are used to obtain more accurate function values. The MOEA/D [20] variant in [21] (MOEA/D-DE) is used as the optimization method. The new algorithm, multiobjective uncertain optimization with ordinal optimization, Latin Supercube sampling and parallel computation (MOOLP), has been proposed and extensively studied in this paper.

The remainder of the paper is organized as follows. Section II introduces the basic techniques used in this paper. Section III gives the details of the proposed approach. Section IV presents the test problems used in our experimental studies. Section V conducts experimental investigations of the major components of MOOLP. Section VI studies the hybridizations of the major components by a real-world analog integrated circuit design problem. Concluding remarks are given in Section VII.

II. Basic Techniques Used in MOOLP

This section provides a brief introduction to some basic techniques used in MOOLP.

A. Differential Evolution

Differential evolution is one of the most popular evolutionary algorithms for continuous optimization. Its distinct characteristics are its mechanisms for generating new solutions.

Copyright (c) 2013 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

There are several variants of DE operators. In our experiments, we use the DE/best/1 operator.

Let P be the current population and x^{best} be the best solution in P. Given a solution $x = (x_1, \ldots, x_d) \in P$, to generate an offspring $u = (u_1, \ldots, u_d)$ for x, the DE/best/1 operator first produces a donor vector:

$$v = x^{best} + SF \cdot (x^{r_1} - x^{r_2}) \tag{2}$$

where x^{r_1} and x^{r_2} are two different solutions randomly selected from P and also different from x^{best}. $SF \in (0, 2]$ is a control parameter, commonly known as the scaling factor [22]. Then the following crossover operator to produce u is applied:

1 Randomly select a variable index $j_{rand} \in \{1, \ldots, d\}$,

2 For each $j = 1$ to d, generate a uniformly distributed random number $rand$ from $(0, 1)$ and set:

$$u_j = \begin{cases} v_j, & \text{if } (rand \leq CR) | j = j_{rand} \\ x_j, & \text{otherwise} \end{cases} \tag{3}$$

where $CR \in [0, 1]$ is a constant called the crossover parameter.

DE is a robust algorithm and appropriate settings of its parameters (the scaling factor and the crossover rate) are detailed in [22].

B. Ordinal optimization

Suppose that we have a quality function:

$$J(x) = E(L(x, \xi)) \tag{4}$$

where x can have M different design values: $x^1, \ldots, x^M$, and ξ is a random variable. We assume that $L(x, \xi)$ is very complex so that $J(x)$ can only be estimated by Monte-Carlo simulation. A simulation for $L(x^i, \xi)$ implies to sample a value of ξ for x^i and compute the value of $L(x^i, \xi)$. Within a fixed simulation budget (i.e., a fixed number of simulations), the goal of OO is to allocate the simulation budgets among different x^i in an intelligent way for finding design values with high quality values.

Given n samples of ξ for x: $\xi_1, \ldots, \xi_n$. $J(x)$ can be estimated as:

$$\bar{J}(x) = (\sum_{j=1}^{n} L(x, \xi_j))/n \tag{5}$$

The basic idea behind OO is that, to select good designs, one only needs a reliable order of $J(x^i)$ instead of very accurate estimation of their values. OO allocates a large portion of simulations to promising x^i (often called critical solutions in the OO literature). Let σ_i^2 stand for the variance of $L(x^i, \xi)$. In OO, it can be replaced by its sample variance, and the sample variance of x is:

$$\sigma^2(x) = \frac{1}{n-1} \sum_{j=1}^{n} (L(x, \xi_j) - \bar{J}(x))^2 \tag{6}$$

Let x^b be the best design value, i.e. the one with the largest $\bar{J}$ value, and $\delta_{bj} = \bar{J}_b - \bar{J}_j$. Let n_i denote the number of simulations allocated to x^i, an asymptotic optimal budget allocation should have the following property [23]:

$$\begin{aligned} &n_b = \sigma_b (\textstyle\sum_{i=1, i \neq b}^{M} n_i^2 / \sigma_i^2)^{1/2} \\ &n_i / n_j = (\frac{\sigma_i / \delta_{bi}}{\sigma_j / \delta_{bj}})^2, i, j = 1, \ldots, M;\ i \neq j \neq b. \end{aligned} \tag{7}$$

An OO algorithm based on (7) works as follows [23]:

1 Set $l = 0; n_1^l = \ldots = n_M^l = n_0$. Perform n_0 simulations for each x^i. Estimate $\bar{J}(x^i)$ and σ_i^2. Find x^b and compute δ_{bj}.

2 If $TO1$, the total number of simulations performed so far, exceeds the total budget TO, stop.

3 Increase the simulation budget by Δ and compute the new budget allocation $n_1^{l+1}, \ldots, n_M^{l+1}$ which satisfy (7) and the constraint:

$$n_1^{l+1} + \ldots + n_M^{l+1} = TO1 + \Delta.$$

If $n_i^{l+1} \geq n_{max}$, then reset $n_i^{l+1} = n_{max}$.

4 Perform additional $max(0, n_i^{l+1} - n_i^l)$ simulations for x^i, $i = 1, \ldots, M$. Use all the simulations performed so far to update $\bar{J}(x^i)$, σ_i^2, x^b and δ_{bj}. Set $l = l + 1$ and go to Step 2.

n_0, TO, n_{max} and Δ are control parameters. In Step 1, n_0 simulations for each x^i are performed to provide a very rough estimate of $J(x^i)$. Based on the simulations made so far, Step 3 computes, by using (7), the optimal budget allocation with additional Δ simulation budget. n_{max} is used to bound the simulation number for each x^i. Step 4 conducts additional simulations and updates $\bar{J}(x^i)$ and other statistics, which can hopefully make the rank of $\bar{J}(x^i)$ more accurate.

In our constraint satisfaction phase, we need to estimate and compare the yield (i.e. $Y(x)$) values of a set of candidate solutions. Let

$$I(x, \xi) = \begin{cases} 1, & \text{if all the } g_i(x, \xi) \geq 0 \text{ for } i = 1, \ldots, k; \\ 0, & \text{otherwise.} \end{cases}$$

Note that

$$Y(x) = E(I(x, \xi)),$$

we can treat $I(x, \xi)$ as $L(x, \xi)$ and then apply the above OO method to estimate and compare the $\bar{Y}$ values of these solutions.

C. Latin Supercube sampling (LSS)

In our proposed method, we need to estimate $Y(x)$ and $Q(x)$ by sampling. Both of them can be regarded as specific cases of (4). Assume that ξ in (4) is uniformly distributed in the s-dimensional supercube $[0, 1)^s$ (otherwise, we can use a variable transformation to make this assumption true). Suppose that we use (5) to compute $\bar{J}(x)$. The estimation error can be bounded by the Koksma-Hlawka inequality [24]:

$$|\bar{J}(x) - J(x)| \leq D^*(\xi_1, \ldots, \xi_n) V_{HK}(L) \tag{8}$$

where $D^*(\xi_1, \ldots, \xi_n)$ is the so-called star discrepancy of $\xi_1, \ldots, \xi_n$ in $[0, 1)^s$: more uniformly distributed samples have lower D^*. $V_{HK}(L)$ is the total variation of $L(x, \cdot)$ in the sense

Copyright (c) 2013 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

of Hardy and Krause. LSS tries to generate a sequence of points in $[0,1)^s$ with a low star discrepancy.

As a first step, LSS divides ξ into several subvectors. A lower-dimensional (randomized) quasi-Monte Carlo ((R)QMC) method is used with each subvector to generate a set of (R)QMC points. Reference [15] shows that QMC can achieve a 2-50 times speed enhancement compared to PMC simulation (using computer generated random numbers). (R)QMC uses low-discrepancy sequences (LDS) to generate more uniformly distributed samples, so as to obtain a much smaller D^* compared to the fake random numbers generated by the computer. LDS is a deterministic infinite sequence of s-dimensional points. The goal of the constructed sequences is to fill the space as geometrically equidistant as possible. There are different methods to generate a LDS, e.g. *Halton* set, *Sobol* set, etc. In MOOLP, we use RQMC samples generated by the *Sobol* set with a skip of $2^{\lfloor logn/log2 \rfloor}$ points. More details can be found in [18], [24], [25].

The next step of LSS is to present these (R)QMC points in a random order within subvectors to produce a set of points in $[0,1)^s$. The purpose is to address high dimensional sampling. Suppose ξ is a s-dimensional input sample, which is divided into k_g groups ($s = k_g \times s_g$). $\xi_i^j, (i = 1,\ldots,n,\ j = 1,\ldots,k_g)$ is a s_g-dimensional (R)QMC point set. The i^{th} LSS sample is:

$$\xi_i = (\xi^1_{\pi_1(i)}, \ldots, \xi^{k_g}_{\pi_{k_g}(i)}), i = 1,\ldots,n \tag{9}$$

where π_j are independent uniform random permutations of $1,\ldots,n$. The purpose of this random permutation is the same as in LHS, i.e. to make the projection of each coordinate of the samples more uniform so as to reduce the variance. The details of LSS can be found in [18].

III. The MOOLP Algorithm

A. Main Idea

To estimate $Y(x)$ and $Q(x)$ in (1), one has to do computer simulations which are often computationally expensive in practice. Thus, our goal is to reduce the number of simulations by as many as we can. We have the following considerations:

- Constraints are more important than objectives. When a candidate solution does not meet the constraints, we do not deliver it as a final solution to the decision maker. Therefore, it is unnecessary to spend much effort to estimate the objective function values of an infeasible solution.
- In most engineering applications, if a solution does not meet the following deterministic constraints:

  $$g_i(x,0) \geq 0 \text{ for } i = 1,\ldots,k, \tag{10}$$

  which is the constraint without variations, then its $Y(x)$ should be zero or very low, and is unlikely to meet the chance constraint:

  $$Y(x) \geq \theta$$

  for practical values of θ. Therefore, it is unnecessary to estimate $Y(x)$ when x is infeasible in terms of the deterministic constraints.
- It is relatively easy to find a set of (nearly) feasible solutions compared with the optimization problem itself. If most initial solutions are feasible and of good diversity in an EA, then its computational effort should be significantly reduced.

Based on the above considerations, we divide our search into two phases: constraint satisfaction and optimization. The constraint satisfaction phase tries to find a set of nearly feasible solutions which have a good diversity. The optimization phase takes the output of the constraint satisfaction phase as its initial population and searches for an approximate PF.

In the constraint satisfaction phase, we do not need to estimate $Q(x)$, and only estimate $Y(x)$ when necessary. More importantly, the goal is to provide an initial population to the optimization phase, so we only need to have a rough estimate of $Y(x)$, particularly for those solutions which are unlikely to be feasible since those solutions would be discarded later. Thus, it is very reasonable to use the OO method in this phase.

In the optimization phase, since its initial population is almost feasible, thus we do not need to employ very sophisticated constraint handling techniques in this phase. For simplicity, we use the penalty function approach to maintain the feasibility. Because this phase produces the final solutions to the decision maker, we allocate a good number of simulations to each candidate solution to estimate its objective function values.

Since the LSS is much more efficient than other sampling techniques, we use it in both the constraint satisfaction and optimization phase. In addition, both, the evaluations of individuals in each iteration of evolutionary algorithms and their MC simulations are independent from each other, and, hence, parallel computation can be easily applied.

B. Phase 1: constraint satisfaction

The goal of the constraint satisfaction phase is to provide a good initial population to the optimization phase. We set the following two requirements for its output population Pop:

- For each solution x in Pop, $Y(x) \geq 0.9\theta$. The reason why we relax the chance constraint is that nearly feasible solutions can often provide very useful information for the optimization phase, and those solutions can be also helpful to increase the diversity of Pop.
- Pop should be of good diversity in the decision space. We use the following metrics to measure the uniformness of Pop:

  $$Uni(Pop) = \sum_{x \in Pop} (\overline{dis} - dis(x, Pop\backslash\{x\})^2 \tag{11}$$

  where $Pop\backslash\{x\}$ denotes the whole population Pop except the individual x and $dis(x, Pop\backslash\{x\})$ is the Euclidean distance between x and the individual of $Pop\backslash\{x\}$ closest to x, i.e.

  $$dis(x, Pop\backslash\{x\}) = min_{x' \in Pop\backslash\{x\}} ||x - x'||,$$

  and $\overline{dis}$ is the average of all these distances. The smaller the Uni value, the more uniform is the distribution of the points in Pop.

Copyright (c) 2013 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

The constraint satisfaction starts with a population of N solutions randomly generated from $[a,b]^d$. At each generation, it produces one offspring solution for each solution in the current population by using the DE operators (Section II (A)). To evaluate the qualities of these N new solutions, we split them into two groups:

- Group A: it contains all the new solutions which do not meet the deterministic constraints $g_i(x,0)$,
- Group B: it contains all the other new solutions.

For a solution x in Group A, we define its deterministic constraint violation $DCV(x)$ as the sum of the violations of each constraint. We do not need to do more simulations to estimate its Y value and simply set it to zero. We apply OO on Group B to estimate the Y values of its solutions. The main goal is to obtain a good ranking for the candidate solutions to make sure the selection is correct, while decreasing the necessary number of simulations as much as possible.

To update the population, we compare each solution x and its offspring o. o replaces x if and only if one of the following conditions is true:

- when x violates some deterministic constraints and $DCV(o) < DCV(x)$,
- when $DCV(x) = 0$, $Y(x) < 0.9\theta$ and $Y(x) < Y(o)$,
- when $Y(x) \geq 0.9\theta$, $Y(o) \geq 0.9\theta$, and Uni value of Pop will decrease if x is replaced by o.

We use the estimated Y values in the above comparison. Due to the nature of OO, for those solutions which severely violate the chance constraint, they will be allocated a very small number of simulations and thus their estimated Y values are not very accurate. However, it will not have a large effect since these solutions should be replaced during the constraint satisfaction, whereas the (nearly) feasible solutions for the initial population of the optimization phase have a good estimation accuracy because much more samples are allocated to them.

For simplicity, we always use the current population Pop as the baseline when we check if the Uni value decreases by replacing x with o. We adopt the generational DE in which all the replacements are conducted in parallel. Thus, a replacement based on the Uni change computed in our approach may not lead to the Uni decrease in some cases. However, since only a few solutions are replaced at each generation when the candidates move to the feasible space, our approach works very well in practice as confirmed in our pilot experimental studies.

We have observed that the Uni value of the current population increases first as the population moves towards the feasible region and then decreases as Uni becomes the major driving force. The constraint satisfaction phase stops when the Uni value is smaller than that of the initial population or the number of iterations used exceeds a predefined number. The constraint satisfaction phase is summarized as follows:

1 Randomly sample N candidate solutions uniformly distributed in $[a,b]^d$ to form the initial population Pop. Compute the DCV values of all the solutions, divide them into two groups and set/estimate their Y values accordingly. If no solution in Pop meets all the the deterministic constraints, the best solution is that with the smallest DCV value. Otherwise, the best solution is that with the largest Y value.

2 Use the DE operators to create an offspring for each solution in Pop. Compute the DCV values of all the offsprings, divide them into two groups and set/estimate their Y values accordingly.

3 Compare each solution and its offspring, and update Pop.

4 If the stopping condition is met, output Pop. Otherwise, go to Step 2.

C. Phase 2: objective optimization

MOEA/D is used for multiobjective optimization in the optimization phase. A general framework of MOEA/D is proposed in [20]. By using a (linear or nonlinear) weighted aggregation method, the approximation of the PF can be decomposed into a number of single objective optimization subproblems. MOEA/D defines neighbourhood relations among these subproblems based on the distances among their weight vectors. Each subproblem is optimized in MOEA/D by using information mainly from its neighbouring subproblems. In our experiments, we employ the Tchebycheff approach and a penalty function technique for transforming (1) into N unconstrained single objective optimization problems. More specifically, the k-th subproblem is formulated as the minimization of:

$$\begin{aligned} F^k(x) &= max\{\frac{k}{N-1}|Y(x)-Y^*|, \frac{N-k-1}{N-1}|Q(x)-Q^*|\} \\ &\quad +\rho \times (max\{\theta - Y(x), 0\}) \end{aligned} \tag{12}$$

where Q^* and Y^* are the largest values for $Q(x)$ and $Y(x)$ in the feasible region.

It is worth noting that as Q^* and Y^* are usually unknown before the search, the algorithm uses the largest Q and Y values found during the search to substitute them. We generate the same number of LSS samples for each candidate solution to estimate its Y and Q values. The number of the LSS samples should be large enough to obtain a good accuracy.

There are a number of different variants of MOEA/D. In our experiments, we adopt MOEA/D-DE proposed in [21]. During the search, MOEA/D maintains:

- a population of N solutions $x^1,\ldots,x^N \in [a,b]^d$, where x^i is the current solution to the i-th subproblem; and their estimated Q and Y values,
- Q^* and Y^*, the largest Q and Y values found so far, respectively.

Let $B(i), i \in \{1,\ldots N\}$ contain the indices of the T closest neighbours of x^i, defined by the Euclidean distances between other weight vectors and i. The k-th subproblem is a neighbour of the i-th problem if $k \in B(i)$. Neighbouring problems have similar objective functions, so their optimal solutions should be similar too. Before applying MOEA/D, we should set the value of the neighbourhood size T. We also need to set the value of n_r, the maximal number of solutions allowed to be replaced by a single new solution.

The algorithm is summarized as follows:

Copyright (c) 2013 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

1. **Initialization:**

1.1: Set the output of Phase 1 as the initial population. Use n_{ac} LSS samples to estimate the Y and Q values of each individual.

1.2: Assign the solutions in the initial population to the subproblems in a random way.

1.3: Initialize Q^* and Y^* as the largest Q and Y function values, respectively, in the initial population.

2. **Update:**

For $i = 1, \ldots, N$

2.1 Selection of the mating pool:

Generate a uniformly distributed random number $rand$ in (0,1]. Set

$$P_{nb} = \begin{cases} B(i), & \text{if } rand < \delta, \\ \{1, \ldots, N\}, & \text{otherwise.} \end{cases} \quad (13)$$

2.2 Reproduction:

Set $x^{best} = x^i$, randomly select two indexes, r_1 and r_2, from P_{nb}, and generate a new solution $\overline{y}$ using the DE operator introduced in Section II (A). Then, perform a polynomial mutation [26] on $\overline{y}$ with probability p_m to produce a new solution y.

2.3 Use LSS sampling to estimate $Q(y)$ and $Y(y)$.

2.4 Update Q^* and Y^*:

If $Q(y) > Q^*$, set $Q^* = Q(y)$. If $Y(y) > Y^*$, set $Y^* = Y(y)$.

2.5 Update of Solutions: Set $c = 0$ and then do the following:

(1) If $c = n_r$ or P_{nb} is empty, go to Step 3. Otherwise, randomly pick an index j from P_{nb}.

(2) If $F^i(y) < F^i(x^j)$, then set $x^j = y$, and $c = c + 1$.

(3) Remove j from P_{nb} and go to (1).

End

3. **Stopping**

If the stopping criteria (e.g. a certain number of iterations) are satisfied, then stop and output $\{x^1, \ldots, x^N\}$, $\{Y(x^1), \ldots, Y(x^N)\}$ and $\{Q(x^1), \ldots, Q(x^N)\}$. Otherwise go to Step 2.

IV. Test Problems

We have experimentally studied our proposed method on constructed mathematical test problems for the main components of MOOLP and a real-world engineering design problem for the hybridization of the main components (the full MOOLP algorithm).

A. Variation-aware analog sizing problems

This paper uses the proposed method to solve a real-world analog integrated circuit design problem, the design of a two-stage fully-differential folded-cascode amplifier with common-mode feedback, shown in Fig. 1. The circuit is designed in a 90nm CMOS process with 1.2V power supply. The specifications are:

- DC gain$\geq$ 60dB,
- gain bandwidth product$\geq$ 45MHz,
- phase margin$\geq$ 60°,
- output swing$\geq$ 2V,
- power$\leq$ 2.5mW,
- $\sqrt{area} \leq 180\mu m$.

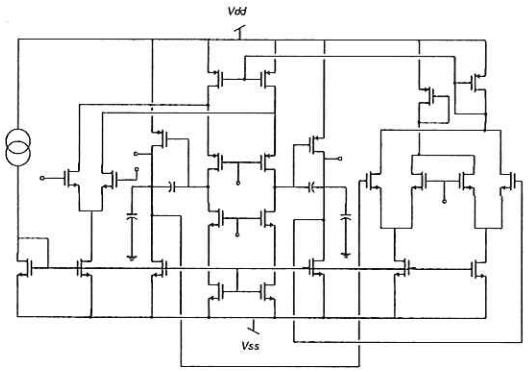

Fig. 1. Two-stage fully differential folded-cascode amplifier

$Y(x)$ is the probability that a design meets all the above specifications. The chance constraint is $Y(x) \geq 85\%$. The goal is to maximize $Y(x)$ and minimize the power consumption ($Q(x)$). There exist 21 design variables, including transistor widths W_i, transistor lengths L_i, the compensation capacitance Cc and the biasing current I_b. The bounds of the design variables are: $0.12\mu m \leq W_i \leq 800\mu m$, $0.1\mu m \leq L_i \leq 20\mu m$, $0.1pF \leq Cc \leq 50pF$, and $0.05mA \leq I_b \leq 50mA$. All transistors must be in the saturation region. The number of process variation variables (noise variables) is 52, which are all normally distributed. To estimate $Y(x)$ and $Q(x)$, all these 52 variation variables need to be considered. Note that in this application, the uncertain parameters and the design parameters do not have a one-to-one correspondence. The design parameters are width and length for each transistor, but there are several inter-die and intra-die variation variables, which need to be considered when evaluating the performance of a circuit.

B. Problems for testing the two-phase optimization mechanism

One of the main characteristics of MOOLP is the two-phase optimization mechanism. To test the effectiveness of this mechanism to handle constrained MOPs, some problems are constructed. Since the uncertainty handling is not the major purpose of this test (tested by using the real-world engineering problem), we use the deterministic unconstrained test problems UF1 to UF4 and UF7 to UF10 from the CEC09 competition suite [27] and add constraints to some objectives. Although they are deterministic problems, they have some similarity to (1), since some objectives also serve as constraints. For the two-objective problems, the added constraints are

$$f_1(x) \leq 0.6$$

and

$$f_2(x) \leq 0.8$$

Copyright (c) 2013 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

TABLE I
X_t OF 11 SETS OF TEST POINTS FOR OO

group	X(1)	X(2)	X(3)	X(4)	X(5)	X(6)
1	13.7	14.0	14.9	15.1	15.7	16.3
2	13.4	13.7	14.5	15.1	15.7	16.9
3	13.4	14.8	15.5	15.9	16.2	16.7
4	13.3	13.4	15.1	15.3	15.5	16.2
5	13.1	13.3	13.5	13.6	15.5	16.8
6	13.0	13.1	13.8	14.4	14.5	16.9
7	13.8	14.2	14.4	14.7	16.1	16.7
8	13.9	14.2	14.4	14.8	15.5	16.8
9	13.4	13.9	14.2	14.7	14.8	16.6
10	13.4	13.5	14.0	14.2	14.9	15.3
11	13.7	15.0	15.7	15.9	16.1	16.7

For the three-objective problems, the added constraint is

$$f_3(x) \leq 0.7$$

Problems UF5 and UF6 are not considered because their Pareto sets are discrete. The number of decision variables is 20. In the next section, the two-phase optimization mechanism will be compared with a single-phase approach in which MOEA/D is used with penalty functions.

C. Problems for testing OO

The goal of OO is to select (nearly) feasible solutions for a chance constraint with as few simulations as possible. It can also provide relatively accurate estimations to those (nearly) feasible solutions. A chance constraint problem is defined as

$$Y(k) = Pr\{X(k) + Nrand \leq 15\} \geq 75\%$$

where $X(1), \ldots, X(6)$ is a group of six numbers. $Nrand$ represents the sum of 7 standard normally distributed random numbers. The task is to find all the k among $\{1, \ldots, 6\}$ that meet this constraint. Each row in Table I represents such a group of numbers. In other words, this table gives 11 test instances, which are randomly generated around 15. In the next section, OO will be compared with the traditional method which assigns the same number of samples to each candidate.

V. EXPERIMENTAL STUDIES OF THE MAIN COMPONENTS OF MOOLP

This section experimentally studies the main components of MOOLP: the two-phase optimization mechanism, OO for handling chance constraint, separately. Another main component of MOOLP, the LSS simulation, has been studied in literature. The advantage on convergence speed of LSS compared to PMC is detailed in [15], [18]. Hence, we will study the advantage of LSS in handling CBSOP in the next section based on the real-world problem.

A. The two-phase optimization mechanism

This subsection studies the effect of the two-phase optimization mechanism to handle constraints and compares it with a conventional single-phase method based on MOEA/D and penalty functions. The test problems used in this subsection were given in Section IV (B). In our experiments, the population size is 200 for two-objective problems, and 400 for three-objective problems. In the constraint satisfaction phase, the severity of the constraints can be assessed. Problems with severe constraints require more iterations. Based on the severity of constraints and the number of objectives, we classify the test problems into 5 groups, which are shown in Table II.

TABLE II
CLASSIFICATION OF TEST PROBLEMS

Class	Problems	G(init)	Severity	N(obj)
A	UF7	40	Low	2
B	UF1,UF2	100	Medium	2
C	UF3,UF4	150	High	2
D	UF8	50	Low	3
E	UF9,UF10	200	High	3

In Table II, $G(init)$ is the average number of iterations (rounded to tens) among 10 independent runs of the constraint satisfaction phase with the stopping criterion described in section III (B). $G(init)$ can be considered as an indicator of the severity of the constraints. $G(MO)$ in Table III is the number of iterations used in the optimization phase in our proposed approach. The number of iterations assigned to the single-phase MOEA/D with penalty functions to handle constraints is $G(init) + G(MO)$. The penalty coefficient ρ is set to 10, the neighbourhood size T is 10% of the population size and the replacement boundary n_r is 10% of T for both approaches. In the constraint satisfaction phase of our proposed approach, following the suggestions from [22], the scaling factor $SF1$ is set to 0.8 and the crossover rate $CR1$ is set to 0.8 in the DE operators. In the optimization phase of our approach and the single-phase MOEA/D, following the suggestions from [21], the crossover rate $CR2$ is set to 1 and the scaling factor $SF2$ is set to 0.5 in their DE operators.

The inverted generational distance (IGD) [28] is used to assess and compare the performance of the algorithms. Let P^* be a set of uniformly distributed points in the objective space along the true PF. Let A be an approximation to the PF, the inverted generational distance from P^* to A is defined as:

$$IGD(A, P^*) = \frac{\sum_{v_p \in P^*} d(v_p, A)}{|P^*|} \quad (14)$$

where $d(v_p, A)$ is the minimum Euclidean distance between v_p and the points in A. The average IGD values in 20 runs for the benchmark problems are shown in Table III.

From Table III, it is evident that the two-phase approach is not worse than the single-phase MOEA/D with penalty function for all the test problems in terms of the IGD values. A Wilcoxon signed rank test [29] with a significance level of 0.05 has been used to compare the IGD values in 20 runs obtained by the two algorithms. It confirms that our two-phase approach is significantly better on all the test instances except UF2 and UF7. Table III reveals that:

- For the two-objective problems with low and medium severity constraints (classes A and B), our two-phase approach leads to moderate improvement;

Copyright (c) 2013 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

TABLE III
THE IGD STATISTICS BASED ON 20 RUNS OF OUR PROPOSED TWO-PHASE APPROACH (TWO PHASE) AND MOEA/D WITHOUT CONSTRAINT SATISFACTION PHASE (ONE PHASE)

Problems	Class	Two Phase	One phase	G(MO)
UF1	B	0.0024	0.0032	800
UF2	B	0.0039	0.0039	800
UF3	C	0.0257	0.0448	800
UF4	C	0.0532	0.0735	800
UF7	A	0.00075	0.00076	1200
UF8	D	0.0386	0.0432	1500
UF9	E	0.0805	0.1629	2000
UF10	E	0.0854	0.1795	2000

TABLE IV
Y_t FOR TEST INSTANCE 1

$X(k)$	13.7	14.0	14.9	15.1	15.7	16.3
$Y(k)(\%)$	83.77	77.53	52.98	46.92	29.75	16.31

- For the two-objective problems with high severity constraints (class C) or three-objective problems with low severity constraints (class D), the improvement is good;
- For the three-objective problems with high severity constraints (class E), the improvement is very significant.

Hence, it can be concluded that our two-phase optimization approach is effective and efficient for dealing with constrained optimization problems.

B. OO for handling chance constraint

This subsection experimentally studies the ability of OO to handle chance constraints and compares OO with the standard method in which all the candidates are assigned the same number of samples (i.e. simulations). The eleven test instances are given in Section IV (C). In our experiments, PMC simulation is used in both the OO and standard method. We first compare the OO and standard method on the first instance (its data are given in the first row of Table I). Then, we conduct significance tests on the other ten instances (their data are given from row 2 to 10 of Table I). The true values of $Y(k)$ for the first instance are given in Table IV. Thus, the solution to this instance is $k = 1, 2$.

For the first instance, we use different number of samples to estimate $Y(k)$. 100 independent runs for each experiment have been conducted in both methods. The results are presented in Table V. In Table V, the total number of samples in each experiment is $(6 \times n_{sam})$. For the case of the standard method, n_{sam} samples are used for estimating each $Y(k)$. In this table:

- The data in the MOO column are the error rates of using OO, i.e., the percentages of the runs which fail to find the correct k values by the OO method,
- The data in the MTR column are the percentages of the runs which fail to find the correct k values by the traditional method.

From Table V, it is clear that the error rates of the OO are much lower than those of the traditional method. For example, in the case of using 6×390 samples, OO can find the right values of k in all the 100 runs, while the traditional method fails in 11% of the runs. To reduce the error rate below 1%, the traditional method needs 6×2500 samples. Therefore, a 6.5 times speedup can be achieved by the OO compared with the traditional method.

TABLE V
COMPARISON ON IDENTIFYING CRITICAL SOLUTIONS BETWEEN THE OO AND TRADITIONAL METHOD ON THE FIRST INSTANCE

n_{sam}	MTR	MOO
90	36%	16%
120	23%	9%
200	14%	6%
350	11%	2%
390	11%	0
2200	1%	0
2500	0	0

TABLE VI
COMPARISON ON ESTIMATION ACCURACY BETWEEN THE OO AND THE TRADITIONAL METHOD

Instances	$N_{critical}$	$N_{p-value\in(0,0.05]}$	$N_{p-value\in(0.05,0.1]}$	$N_{p-value>0.1}$
2	2	1	0	1
3	1	1	0	0
4	2	2	0	0
5	4	2	0	2
6	3	2	1	0
7	1	1	0	0
8	2	1	0	1
9	2	1	0	1
10	3	2	0	1
11	1	1	0	0

The significance tests of the error rates (like in Table V) are performed for the the second to eleventh instances in Table I. n_{sam} is set to 200. 10 error rates are generated for each method on each instance, which is obtained by 100 runs. Results show that in all of the test instances, OO is significantly better than the traditional method with a significance level of 0.05.

Besides the correct selection for handling chance constraints studied in the previous experiments, the accuracy of the estimations using the two methods are also compared. The second to eleventh instances in Table I are used. n_{sam} is set to 200. 500 runs are performed for each method on each instance. For each instance, the candidates that meet the chance constraint (indicated by the k values) are called critical candidates. Their corresponding $Y(k)$ values need good estimations. The Wilcoxon signed rank test is performed to compare the two methods on their estimated $Y(k)$ at these critical candidates. The results are summarized in Table VI. Column two shows the number of critical candidates in the ten test instances. Column three to column five show the number of critical candidates falling in the three p-value intervals from the Wilcoxon signed rank test, respectively.

It can be seen that under a significance level of 0.05, the difference of the estimated $Y(k)$ between these two methods is significant and the results of OO are much closer to the correct values for at least 50% of the critical candidates in all the test instances. Nevertheless, for other critical candidates, the two estimations show no significant difference.

Copyright (c) 2013 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

VI. Experimental Studies of the Hybridization on a Real World Application Problem

In this section, we use the proposed approach to solve the real-world process variation-aware analog circuit sizing problem described in Section IV (A). The goal is to test the hybridization of the main techniques in the complete MOOLP algorithm. We also compare our proposed method with the single-phase MOEA/D using PMC samples. At last, the hybridization of OO and LSS is studied by experiments based on this real-world problem.

The experiments were implemented in Matlab running on a PC with an 8-core CPU, 4GB RAM and the Linux operating system. The electrical simulator HSPICE is used as the circuit performance evaluator.

In our constraint satisfaction phase, the parameter settings are:

- $N = 50$.
- $SF1 = 0.8$ and $CR1 = 0.8$
- $n_0 = 20$, $n_{max} = 600$
- $TO = sim_{ave} \times M1$ where $sim_{ave} = 60$ and $M1$ is the number of nearly feasible solutions defined in section III, $\Delta = 0.2 \times TO$

The parameter settings in the optimization phase are:

- $N = 50$
- $T = 5$, $n_r = 3$, $\delta = 0.9$, $SF2 = 0.5$, $CR2 = 1$
- $n_{ac} = 1000$
- $\rho = 10$
- the number of iterations is 300

The wall clock time spent in this problem is 112 hours.

For each of the final 50 solutions, we have conducted 5000 LSS samples (i.e. simulations) to obtain good estimates of their final Q and Y values. In all experiments using this problem, we round the estimated Y value to 1% scale and the Q value to 0.001mW. Because the accuracy of the estimations improves when the number of samples is increased (the sample variance approximates the population variance), the method of rounding to a certain scale is a trade-off between the necessary number of LSS samples and the density of the points in the PF. For $Y(x)$ in this application, 1% is a sufficient accuracy and we found that $n_{ac} = 1000$ LSS samples is often enough to achieve 1% accuracy. Due to the rounding up, the number of solutions in the PF is at most 16 ($Y(x) \geq 85\%$). 15 final non-dominated solutions in the objective space are obtained. In addition, following the practice in robust circuit optimization [30], the estimated $Q(x)$ value is replaced by $\overline{Q(x)} + 3\sigma_{Q(x)}$ in the optimization process, where $\overline{Q(x)}$ is the average value of the available samples and $\sigma_{Q(x)}$ is the standard deviation of them.

For comparison, we also used MOEA/D with PMC simulation and only one optimization phase for solving this problem. n_{ac} PMC samples are used for each candidate solution to estimate their Y and Q values. All the parameter settings are exactly the same as those in our proposed MOOLP approach. We found that this method could not produce any feasible solutions after 112 hours of wall clock time if we just use a penalty function for the chance constraint $Y(x) \geq 85\%$. This result shows that the two-phase approach is necessary for this real-world problem. Then, we included $g_i(x, 0) \geq 0$ (i.e. the specifications without considering process variations) in the constraints and add them together with the chance constraint to the penalty function. After 112 hours, the solutions generated and post-processed like in MOOLP and the approximate PF obtained by MOOLP are shown in Fig. 2. It is clear that the approximate PF by the traditional method is much worse than that generated by MOOLP:

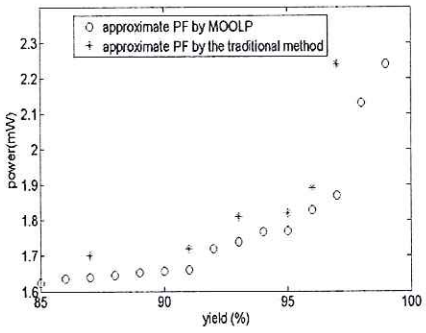

Fig. 2. Comparison of the power-yield trade-off of the amplifier by MOOLP and the traditional method

- the largest obtained yield is lower than that obtained by MOOLP,
- there are many gaps in the PF compared with that obtained by MOOLP,
- all points in the PF are dominated by those provided by MOOLP.

These performance differences are due to the following reasons:

- without the constraint satisfaction phase, many samples are wasted to infeasible solutions, and less computational effort is allocated to important solutions.
- LSS can provide better samples than PMC, which makes that MOOLP can estimate the Y and Q values more accurately.

Using MOEA/D with PMC simulation to obtain comparable results of MOOLP may take too long time. Here, it is roughly estimated. In the constraint satisfaction phase, MOOLP takes 3.4 hours. According to previous experiments in section V, we can safely estimate that OO and LSS speed up this phase by at least 5 times. In the optimization phase, MOOLP takes 109 hours. To estimate the speedup brought by LSS, 50 individuals generated in the MOOLP optimization are randomly selected. We have compared the estimation quality of Y of these points by using PMC and LSS. The experiments show that to obtain about the same variance (estimated by 5 runs), PMC needs at least 5 times more samples than LSS. Therefore, without using OO and LSS, it may take about a month to obtain a PF of the same quality compared to MOOLP, which is often unbearable in practice.

OO has been traditionally used with PMC simulation whereas we have applied OO with LSS simulation. An interesting issue is to investigate if OO provides the same or even more advantages with LSS than with PMC. To address this issue, we have run the constraint satisfaction phase of this integrated circuit example with LSS and with PMC for 5 times.

Copyright (c) 2013 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

TABLE VII
COMPARISON OF THE PERCENTAGES OF THE FEASIBLE SOLUTIONS IN THE CONSTRAINT SATISFACTION PHASE WITH LSS AND THAT WITH PMC

Seed	OO+LSS	OO+PMC
1	86%	74%
2	64%	54%
3	84%	86%
4	62%	4%
5	76%	68%

Each time both implementations start from the same randomly generated population (same seed) and are allocated the same number of simulations. In each run the total number of LSS or PMC simulations is set to 100,000. Note that 100,000 samples have substantial contributions to the constraint satisfaction phase but it is usually an insufficient number to generate the output population for this high-performance analog circuit sizing problem. Hence, we can clearly observe the differences of the convergence speed from the number of obtained feasible candidates. The parameter settings in the constraint satisfaction phase have been listed in section VI. The yield of the candidates obtained by both methods are re-evaluated by using 5000 LSS samples and the percentage of feasible solutions can be calculated (the constraint in this phase is $Y(x) \geq 0.9 \times 85\%$, see section III). Table VII lists the percentages of feasible solutions among all the final solutions obtained by the two techniques in five runs. It is clear that the constraint satisfaction phase with LSS works better than the one with PMC on this real-world problem.

VII. CONCLUSIONS

In this paper, the MOOLP algorithm has been proposed for solving chance-constrained bi-objective stochastic optimization problems. A two-phase optimization mechanism is proposed. Different speed enhancement methods are used in these two phases. Both the solution feasibility and the diversity of the current population are considered in the constraint satisfaction phase. By using the ordinal optimization technique with a small number of LSS samples, nearly feasible solutions can be correctly identified with reasonably accurate estimations, while the computational cost on infeasible solutions can be greatly reduced. In the optimization phase, by using LSS with a large number of samples, a significant speed enhancement can be achieved compared with primitive MC simulation. Parallel computation is also included to speed-up the computation. Experimental studies on the major components of the proposed approach have been conducted. Their hybridization, the complete algorithm, has been applied to a real-world process variation-aware analog circuit sizing problem. From the tests, high speed enhancement and high quality results have been observed. We can conclude that the proposed MOOLP is efficient and effective for dealing with CBSOP, and thus provides a practical solution for computationally expensive manufacturing engineering problems.

ACKNOWLEDGEMENT

This research has been supported by the Humboldt research fellowship, Germany and a scholarship of Katholieke Universiteit Leuven, Belgium.

REFERENCES

[1] T. McConaghy, P. Palmers, M. Steyaert, and G. Gielen, "Variation-aware structural synthesis of analog circuits via hierarchical building blocks and structural homotopy," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, no. 9, pp. 1281–1294, 2009.

[2] B. Liu, "Uncertain programming," *Uncertainty Theory*, pp. 81–113, 2010.

[3] L. Mercado, S. Kuo, T. Lee, and R. Lee, "Analysis of rf mems switch packaging process for yield improvement," *Advanced Packaging, IEEE Transactions on*, vol. 28, no. 1, pp. 134–141, 2005.

[4] H. Chan and P. Englert, *Accelerated stress testing handbook*. IEEE Press, 2001.

[5] C. Poojari and B. Varghese, "Genetic algorithm based technique for solving chance constrained problems," *European journal of operational research*, vol. 185, no. 3, pp. 1128–1154, 2008.

[6] M. Chu and D. Allstot, "Elitist nondominated sorting genetic algorithm based rf ic optimizer," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 52, no. 3, pp. 535–545, 2005.

[7] S. Ali, R. Wilcock, P. Wilson, and A. Brown, "A new approach for combining yield and performance in behavioural models for analogue integrated circuits," in *Proceedings of the conference on Design, automation and test in Europe*. ACM, 2008, pp. 152–157.

[8] C. Goh and K. Tan, *Evolutionary multi-objective optimization in uncertain environments: issues and algorithms*. Springer Verlag, 2009, vol. 186.

[9] ——, "An investigation on noisy environments in evolutionary multiobjective optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 3, pp. 354–381, 2007.

[10] H. Gupta and K. Deb, "Handling constraints in robust multi-objective optimization," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 1. IEEE, 2005, pp. 25–32.

[11] C. Goh and K. Tan, "Evolving the tradeoffs between pareto-optimality and robustness in multi-objective evolutionary algorithms," *Evolutionary Computation in Dynamic and Uncertain Environments*, pp. 457–478, 2007.

[12] Q. Zhang, J. Liou, J. McMacken, J. Thomson, and P. Layman, "Development of robust interconnect model based on design of experiments and multiobjective optimization," *Electron Devices, IEEE Transactions on*, vol. 48, no. 9, pp. 1885–1891, 2001.

[13] C. Andrieu, N. De Freitas, A. Doucet, and M. Jordan, "An introduction to mcmc for machine learning," *Machine learning*, vol. 50, no. 1, pp. 5–43, 2003.

[14] A. Nazemi, X. Yao, and A. Chan, "Extracting a set of robust pareto-optimal parameters for hydrologic models using nsga-ii and scem," in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*. IEEE, 2006, pp. 1901–1908.

[15] A. Singhee and R. Rutenbar, *Novel algorithms for fast statistical analysis of scaled circuits*. Springer Verlag, 2009.

[16] M. Cowles and B. Carlin, "Markov chain monte carlo convergence diagnostics: a comparative review," *Journal of the American Statistical Association*, pp. 883–904, 1996.

[17] P. Lv and P. Chang, "Rough programming and its application to production planning," in *Risk Management & Engineering Management, 2008. ICRMEM'08. International Conference on*. IEEE, 2008, pp. 136–140.

[18] A. Owen, "Latin supercube sampling for very high-dimensional simulations," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 8, no. 1, pp. 71–102, 1998.

[19] Y. Ho, Q. Zhao, and Q. Jia, *Ordinal optimization: Soft optimization for hard problems*. Springer-Verlag New York Inc, 2007.

[20] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 6, pp. 712–731, 2007.

[21] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 2, pp. 284–302, 2009.

[22] K. Price, R. Storn, and J. Lampinen, *Differential evolution: a practical approach to global optimization*. Springer Verlag, 2005.

[23] C. Chen, J. Lin, E. Yucesan, and S. Chick, "Simulation budget allocation for further enhancing the efficiency of ordinal optimization," *Discrete Event Dynamic Systems*, vol. 10, no. 3, pp. 251–270, 2000.

[24] H. Niederreiter, "Random number generation and quasi-monte carlo methods, volume 63 of siam cbms-nsf regional conference series in applied mathematics," *SIAM, Philadelphia*, 1992.

Copyright (c) 2013 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

[25] J. Matousek, "On the l 2-discrepancy for anchored boxes," *Journal of complexity*, vol. 14, no. 14, pp. 527–556, 1998.

[26] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, 2002.

[27] Q. Zhang, A. Zhou, S. Zhao, P. Suganthan, W. Liu, and S. Tiwari, "Multiobjective optimization test instances for the cec 2009 special session and competition," *University of Essex and Nanyang Technological University, Tech. Rep. CES-487*, 2008.

[28] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 2, pp. 117–132, 2003.

[29] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[30] B. Liu, F. V. Fernández, and G. Gielen, "Efficient and accurate statistical analog yield optimization and variation-aware circuit sizing based on computational intelligence techniques," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 30, no. 6, pp. 793–805, 2011.

Francisco V. Fernández received the Physics-Electronics degree from the University of Seville, Seville, Spain, in 1988, and the Ph.D. degree in 1992.

In 1993, he was a Post-Doctoral Research Fellow with Katholieke Universiteit Leuven, Leuven, Belgium. From 1995 to 2009, he was an Associate Professor with the Department of Electronics and Electromagnetism, University of Seville, where he was promoted to Full Professor in 2009. He is currently a Researcher with IMSE-CNM, CSIC, University of Seville. He has authored or edited three books and has coauthored more than 100 papers in international journals and conferences. His current research interests include the design and design methodologies of analog and mixed-signal circuits.

Dr. Fernández is currently the Editor-in-Chief of Integration, the VLSI Journal (Elsevier). He regularly serves at the program committees of several international conferences. He has participated as a researcher or a main researcher in several national and European research and development projects.

Bo Liu received the B.S. degree from Tsinghua University, P. R. China, in 2008. He received his Ph.D. degree at the MICAS laboratories of the Katholieke Universiteit Leuven, Belgium, in 2012. Since October 2012, he is a Humboldt research fellow and is working with Technical University of Dortmund and Technical University of Munich, Germany. Since April 2013, he is a lecturer (Assistant Professor) in Glyndwr University, U.K.

His research interests lie in evolutionary computation, machine learning, fuzzy logic and design automation methodologies of analog / RF integrated circuits and antennas. He has authored or coauthored 1 book and more than 20 papers in international journals and conference proceedings. He was a technical consultant of Accelicon Technologies Inc. He was a session organizer on computationally expensive optimization and optimization in uncertain environments in IEEE World Congress on Computational Intelligence 2012. He is in the editorial board or as a reviewer in artificial intelligence and electronic design automation fields, such as IEEE Transactions on Evolutionary Computation, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Journal of Engineering. He is also a book reviewer of Elsevier and Bentham Science Publishers.

Georges G. E. Gielen (S'87-M'92-SM'99-F'02) received his M.Sc. and Ph.D. degrees in Electrical Engineering from the Katholieke Universiteit Leuven, Belgium, in 1986 and 1990, respectively. In 1990, he was appointed as a postdoctoral research assistant and visiting lecturer at the department of Electrical Engineering and Computer Science of the University of California, Berkeley. In 1993, he was appointed assistant professor at the Katholieke Universiteit Leuven, where he was promoted to full professor in 2000.

His research interests are in the design of analog and mixed-signal integrated circuits, and especially in analog and mixed-signal CAD tools and design automation (modeling, simulation and symbolic analysis, analog synthesis, analog layout generation, analog and mixed-signal testing). He has authored or coauthored two books and more than 400 papers in edited books, international journals and conference proceedings. He received the 1995 Best Paper Award in the John Wiley international journal on Circuit Theory and Applications, and was the 1997 Laureate of the Belgian Royal Academy on Sciences, Literature and Arts in the discipline of Engineering. He received the 2000 Alcatel Award from the Belgian National Fund of Scientific Research for his innovative research in telecommunications, and won the DATE 2004 Best Paper Award. He is a Fellow of the IEEE, served as elected member of the Board of Governors of the IEEE Circuits And Systems (CAS) society and as chairman of the IEEE Benelux CAS chapter. He served as the President of the IEEE Circuits And Systems (CAS) Society in 2005.

Qingfu Zhang (M'01-SM'06) received the BSc in mathematics from Shanxi University, China in 1984, the MSc in applied mathematics and the PhD in information engineering from Xidian University, China, in 1991 and 1994, respectively.

He is currently a Professor with the School of Computer Science and Electronic Engineering, University of Essex, UK. From 1994 to 2000, he was with the National Laboratory of Parallel Processing and Computing, National University of Defence Science and Technology, China, Hong Kong Polytechnic University, Hong Kong, the German National Research Centre for Information Technology (now Fraunhofer-Gesellschaft, Germany), and the University of Manchester Institute of Science and Technology, Manchester, U.K. He holds two patents and is the author of many research publications. His main research interests include evolutionary computation, optimization, neural networks, data analysis, and their applications.

Dr. Zhang is an Associate Editor of the IEEE Transactions on Evolutionary Computation and the IEEE Transactions on Systems, Man, and CyberneticssCPart B. He is also an Editorial Board Member of three other international journals. MOEA/D, a multobjevitve optimization algorithm developed in his group, won the Unconstrained Multiobjective Optimization Algorithm Competition at the Congress of Evolutionary Computation 2009, and was awarded the 2010 IEEE Transactions on Evolutionary Computation Outstanding Paper Award.

Copyright (c) 2013 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.