Hybrid Estimation of Distribution Algorithm Using Local Function Approximations

Felipe Campelo[1], Frederico G. Guimarães[2,3], Jaime A. Ramírez[2], and Hajime Igarashi[1]

[1]Laboratory of Hybrid Systems, Hokkaido University, Sapporo 060-0814, Japan
[2]Departamento de Engenharia Elétrica, Universidade Federal de Minas Gerais, Belo Horizonte 31720-010, Brazil
[3]Departamento de Computação, Universidade Federal de Ouro Preto, Ouro Preto 35400-000, Brazil

In this paper, we introduce an approach for the design of electromagnetic devices based on the use of Estimation of Distribution Algorithms (EDAs), coupled with approximation-based local search around the most promising solutions. The main idea is to combine the power of EDAs in the solution of hard optimization problems with the faster convergence provided by the local search using local approximations. The resulting hybrid algorithm is tested on a numerical benchmark problem.

***Index Terms*—Estimation of Distribution Algorithms (EDAs), hybrid methods, local learning.**

I. Introduction

ESTIMATION of Distribution Algorithms (EDAs) are a family of techniques within the evolutionary computation framework characterized by the use of explicit probability distributions in optimization [1]. Instead of dealing with vectors of variables, EDAs operate directly on the underlying probabilistic model of the population. Algorithms following this paradigm are also usually capable of linkage learning [2], exploiting the dependencies among variables in a model built around superior solutions, while efficiently traversing the search space [3]. This ability gives EDAs an advantage over traditional evolutionary algorithms (EAs) when dealing with optimization problems where superior solutions tend to present dependencies among variables.

In general, the performance of evolutionary optimization techniques can be improved through the use of local search operators during the evolutionary cycle [4]. These hybrid algorithms combine the ability of the EAs to effectively discover high-performance regions of the search space with the fast convergence to the (locally) optimal point offered by local optimization methods. The major problem with these hybrid techniques is the large number of function evaluations usually required for the local improvement of the solutions, which can be prohibitive in expensive optimization problems, such as the numerical problems usually found in electromagnetic design problems.

The use of approximate models for local improvement has been proposed [4] as a way of reducing the computational cost of these operators. Instead of performing the local search directly over the objective function, local approximations are generated around the most promising solutions and used for the local search procedure. The optimal point found on this approximated space is then evaluated over the objective function and replaces the original solution, reducing the cost of the local search operator to just one extra function evaluation per application.

Manuscript received October 07, 2008. Current version published February 19, 2009. Corresponding author: F. Campelo (e-mail: pinto@em-si.eng.hokudai.ac.jp).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMAG.2009.2012741

In this paper, we couple the approximation-based local search operator mentioned previously with an EDA based on the Helmholtz Machine (HM). The objective is to obtain a hybrid technique with faster convergence and increased performance for the solution of numerical optimization problems in electromagnetics. Section II introduces the HM and Section III describes the approximation-based local search operator. These two components are put together in Section IV. Section V provides an example of application of the proposed methodology and Section VI offers some final considerations.

II. Helmholtz Machine

An HM is "a connectionist system with multiple layers of neuron-like binary stochastic processing units connected hierarchically by two sets of weights" [5]. These two sets of weights form the *Recognition* $(\mathbf{R})$ and *Generative* $(\mathbf{G})$ models. As the name implies, the goal of the $(\mathbf{R})$ model is to determine if a given input vector belongs or not to the same class of vectors used in the training of the model, while the $(\mathbf{G})$ model is responsible for generating patterns similar to those used in the training of the network.

These two models are trained using an unsupervised learning algorithm called *Delta-rule wake-sleep learning* [6], [7]. After the training, the $(\mathbf{G})$ weights model the probability distribution of the input data used. In this paper, the input data will correspond to superior solutions from an optimization process and the $(\mathbf{G})$ model will be used to generate vectors with similar characteristics to these solutions.

In order to illustrate the *wake-sleep* training methodology, suppose an HM with the topology shown in Fig. 1. This network contains two hidden layers and an input (visible) layer, connected by the top-down $(\mathbf{G})$ matrices and the bottom-up $(\mathbf{R})$ matrices, each containing the connection weights between the nodes of two layers, for each model.

Each neuron in a given layer computes a probability value based on the weighted sum of its inputs. For instance, for the middle layer of the $(\mathbf{R})$ model from Fig. 1

$$\mathbf{y} = \sigma(\mathbf{W}^{\mathbf{R}}[\mathbf{d}; 1]) \quad (1)$$

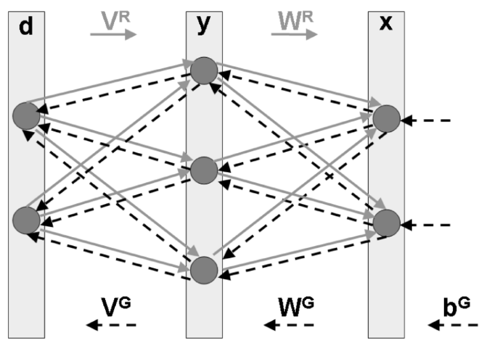

Fig. 1. Three-layer HM. The first layer ($\mathbf{d}$, at the left) is the input one, and the other two are hidden layers. The *Recognition* and *Generative* models are represented by gray solid and black dashed arrows, respectively. Example adapted from [7].

where $[\mathbf{d}; 1]$ represents the vector $\mathbf{d}$ with a 1 added after its last element. This is done for the sake of simplicity in the notation, so that the last column of the matrices can represent the bias values for each layer; σ is a sigmoid function, used to introduce nonlinearities in the network. Also, since this function returns values between 0 and 1, its output can be interpreted as the probability that a binary-valued neuron "fires," and this probability distribution can be sampled by means of a stochastic function $\mathbf{Sample}(\sigma(u))$, that yields a value of 1 with probability given by $\sigma(u)$, and 0 otherwise.

All weights for both the $(\mathbf{R})$ and the $(\mathbf{G})$ models are initially set to 0. The *wake-sleep* training process is used for setting these weights so that they can be used for recognition of input patterns or generation of new data. The two phases of the training algorithm are explained below.

A. Wake Phase

The *wake* phase starts by having the HM randomly sampling its input space. The input (represented by a binary vector) is then propagated through the $(\mathbf{R})$ network

$$\begin{aligned}\mathbf{d} &= \text{Sample}\ \{\text{input space}\}\\ \mathbf{y} &= \text{Sample}\ \{\sigma(\mathbf{V^R}[\mathbf{d};1])\}\\ \mathbf{x} &= \text{Sample}\ \{\sigma(\mathbf{W^R}[\mathbf{y};1])\}\end{aligned} \tag{2}$$

where $\mathbf{V^R}, \mathbf{W^R}$ are the weight matrices that implement the recognition model. The hidden signal is then propagated back through the $(\mathbf{G})$ network, and the computed probabilities are stored

$$\begin{aligned}\boldsymbol{\xi} &= \sigma\ (\mathbf{b^G})\\ \boldsymbol{\psi} &= \sigma\ (\mathbf{W^G}[\mathbf{x};1])\\ \boldsymbol{\delta} &= \sigma\ (\mathbf{V^G}[\mathbf{y};1])\end{aligned} \tag{3}$$

where $\mathbf{b^G}$ is the generative bias vector for the last hidden layer, and $\mathbf{V^G}, \mathbf{W^G}$ are the weight matrices that implement the generative model.

Finally, the generative weights are adjusted according to the delta rule

$$\begin{aligned}\mathbf{b^G} &+= \epsilon(\mathbf{x} - \boldsymbol{\xi})\\ \mathbf{W^G} &+= \epsilon(\mathbf{y} - \boldsymbol{\psi})[\mathbf{x};1]^T\\ \mathbf{V^G} &+= \epsilon(\mathbf{d} - \boldsymbol{\delta})[\mathbf{y};1]^T.\end{aligned} \tag{4}$$

B. Sleep Phase

In the sleep phase, we generate a "fantasy" output vector on the top-most layer of our network, by sampling the sigmoidal probability distribution determined by $\mathbf{b^G}$. We then propagate it through the $(\mathbf{G})$ network

$$\begin{aligned}\mathbf{x} &= \text{Sample}\ \{\sigma(\mathbf{b^G})\}\\ \mathbf{y} &= \text{Sample}\ \{\sigma(\mathbf{W^G}[\mathbf{x};1])\}\\ \mathbf{d} &= \text{Sample}\ \{\sigma(\mathbf{V^G}[\mathbf{y};1])\}.\end{aligned} \tag{5}$$

The "fantasy" input vector $\mathbf{d}$ is propagated back through the $(\mathbf{R})$ model

$$\begin{aligned}\boldsymbol{\psi} &= \sigma\{\mathbf{V^R}[\mathbf{d};1]\}\\ \boldsymbol{\xi} &= \sigma\{\mathbf{W^R}[\mathbf{y};1]\}.\end{aligned} \tag{6}$$

The *sleep* phase is then concluded, with the updating of the recognition weights

$$\begin{aligned}\mathbf{V^R} &+= \epsilon(\mathbf{y} - \boldsymbol{\psi})[\mathbf{d};1]^T\\ \mathbf{W^R} &+= \epsilon(\mathbf{x} - \boldsymbol{\xi})[\mathbf{y};1]^T.\end{aligned} \tag{7}$$

It can be shown that the iterative repetition of the *wake* and *sleep* phases approximates the parameters of both the $(\mathbf{R})$ and $(\mathbf{G})$ model to values that would be obtained through the expectation-maximization (EM) algorithm [6]. A more detailed analysis of the *wake-sleep* training, as well as the mathematical and statistical background on the HM, can be found elsewhere in the literature [5]–[7], and we refer the interested reader to these sources.

After training the HM using the procedure described before, it is possible to obtain vectors that follow the same probability distribution as the input vectors originally used. It is done by performing the steps described in (5) and (6), and taking the "fantasy" vector $\mathbf{d}$ as the output of the generative model.

III. Approximation-Based Local Search

The second component of our methodology is the local search operator. In order to reduce the computational cost of the local search, we employ an approximate model of the neighborhood $\mathcal{V}(\cdot)$ of the points selected for local improvement, as proposed in [4]. These local approximation models are learnt using samples produced during the normal progress of the optimization algorithm, and therefore do not contribute significantly to the overall computational cost in cases where the evaluation of the objective function is time consuming, e.g., electromagnetic design problems involving the solution of numerical models. Given the point to be subject to local search $\mathbf{p^{(i)}}$ and the data set of all points already visited by the evolutionary algorithm $\mathcal{N}(t)$ the local search step in our hybrid algorithm is composed by the following steps:

1) find the samples in $\mathcal{N}(t)$ that belong to the neighborhood $\mathcal{V}(\mathbf{p^{(i)}})$;
2) generate the local approximations for the objective and constraint functions using $\mathcal{V}(\mathbf{p^{(i)}})$;
3) solve the local optimization problem over the approximations;
4) update $\mathbf{p^{(i)}}$ with the locally improved solution.

The local approximations (one for each objective and constraint function in the problem) are learnt by a neural network with one hidden layer [10], and are given by

$$\varphi(p)=\sum_{i=1}^{m} w_i v\left(\sum_{j=1}^{k} s_{ij}p_j+s_{i0}\right)+b_l \tag{8}$$

where $v(\cdot)$ is the nonlinear transfer function of the neuron, and w_i, s_{ij} and b_l are network parameters, which are adapted in the learning phase of the neural network.

Local optimization is then performed over the model generated. The local optimization problem can be defined as

$$\begin{aligned}&\text{minimize: } \tilde{f}(p)\\ &\text{subject to: } p\in\mathcal{V}(p^{(j)})\cap\tilde{\mathcal{F}}\end{aligned} \tag{9}$$

with $\tilde{f}(p)$ being the local approximation of the objective function, and $\tilde{\mathcal{F}}$ the feasible set, defined by the locally approximated constraints.

Following the original formulation of this local operator [4], we use Sequential Quadratic Programming (SQP) [9] to solve the local optimization problem (9). The initial point of the SQP is the original solution $p^{(j)}$, and the locally optimal point obtained over the approximate model replaces the original one. For a more detailed description of this operator, we refer the reader to the bibliography [4].

IV. Hybrid Estimation of Distribution Algorithm

The HM and the approximation-based local search operator are combined to produce a hybrid EDA, using an algorithm structure known as *Bayesian Evolutionary Algorithm* (BEA) [8]. This method (already with the local search step included) can be described as follows:

```
1) generate a random initial population $\mathbb{X}$ composed of $n_{\text{pop}}$ vectors;
2) evaluate population: $\mathbb{F}\leftarrow\mathbf{Eval}(\mathbb{X})$;
3) WHILE (stop criterion not met):
    a) perform local search around the $\sigma$ current best solutions: $\{\mathbb{X},\mathbb{F}\}\leftarrow\mathbf{LS}_\sigma(\mathbb{X},\mathbb{F})$;
    b) select candidate solutions to build the probabilistic model: $\mathbb{D}\leftarrow\mathbf{Sel}(\mathbb{X},\mathbb{F})$;
    c) use selected solutions to train the probabilistic model: $\mathbb{H}\leftarrow\mathbf{Train}(\mathbb{D})$;
    d) generate a number of new candidate solutions using the model: $\tilde{\mathbb{X}}\leftarrow\mathbf{Generate}(\mathbb{H})$;
    e) evaluate the new candidate solutions: $\tilde{\mathbb{F}}\leftarrow\mathbf{Eval}(\tilde{\mathbb{X}})$;
    f) replace $\mathbb{X}$ (totally or partially) by $\tilde{\mathbb{X}}$: $\{\mathbb{X},\mathbb{F}\}\leftarrow\mathbf{Replace}(\mathbb{X},\mathbb{F},\tilde{\mathbb{X}},\tilde{\mathbb{F}})$.
```

In this paper, the selection is implemented as a binary tournament, performed until n_{pop} vectors are selected as input points for the training of the HM. Afterwards, the HM is used to generate $n_{\text{pop}}-1$ new vectors following the distribution of probabilities learned, and the next population is assembled using the best point from the previous population plus the points generated by the HM.

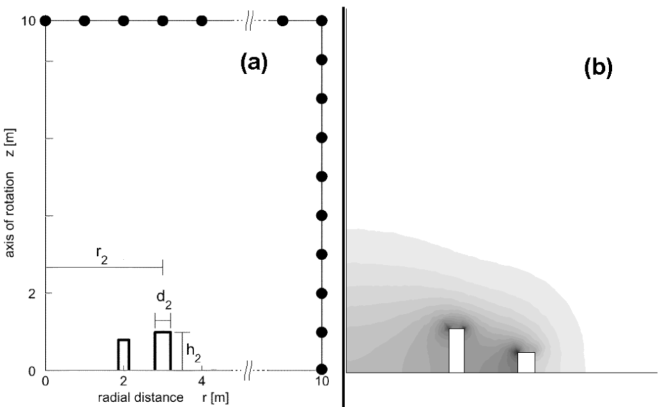

Fig. 2. TEAM Workshop Problem 22 (3 variables version): (a) parametric optimization model; (b) solution found by the hybrid EDA.

It is interesting to notice the two levels of learning performed by this hybrid EDA: the HM tries to uncover statistical relationships among the variables that result in solutions of good performance. The local approximation, on the other hand, tries to accurately represent the vicinity of a single selected point, in order to allow for the SQP algorithm to converge to a locally optimal point. This operator also contributes to the more global modeling performed by the HM, since a point with higher performance will be employed to train the statistical model, which will in turn be able to generate more points with the desired probabilistic distribution in the search space.

V. Results

The proposed methodology was applied for the solution of a benchmark problem in electromagnetic design. The TEAM Workshop benchmark problem 22 [11] consists on the optimization of a superconducting magnetic energy storage (SMES) system. The objective here is to minimize the strayed field while maintaining a prescribed level for the energy stored in the magnetic field generated. Another constraint deals with the maintenance of the superconducting state for the coils. Fig. 2(a) shows the upper half model of this device.

Table I gives the variable limits for the three design parameters r_2, h_2, and d_2, as well as some other relevant information to the problem. Mathematically, the optimization task is defined as in (10)

$$\begin{aligned}\text{minimize: } & f=\sqrt{\frac{\sum_{i=1}^{21}\left|B_S^{(i)}\right|^2}{21}}\\ \text{subject to: } & g=B_{\max}-4.92\le 0\\ & h=\frac{\text{Energy}}{E_{\text{ref}}}-1=0\end{aligned} \tag{10}$$

where $E_{\text{ref}}=180$ MJ is the reference value for the stored energy, and $B_{\max}$ is the maximum value of the magnetic field on the coils, which is involved in the maintenance of the superconducting state. The current density flowing through the coils is kept constant $|J_{\text{inner}}|=|J_{\text{outer}}|=22.5$ A/mm^2, as are the dimensions and position of the inner coil. More detailed information about this problem can be found in the literature [11].

The hybrid EDA was set to solve this problem with the learning factor set to $\epsilon=0.05$ and a three-layer topology for

TABLE I
PROBLEM SPECIFICATIONS

Parameter	r_2 $[m]$	h_2 $[m]$	d_2 $[m]$
min	2.6	0.204	0.1
max	3.4	1.1	0.4

the HM, similar to the one from Fig. 1. The input layer was composed of 30 neurons (corresponding to the coding of each variable by a 10-bit binary vector), and the hidden layers were set to 50 and 20 neurons, respectively. It is important to notice that this topology was arbitrarily chosen, and very likely does not represent the best compromise between representation capability and computational cost. It should, however, be good enough to demonstrate the performance of the hybrid EDA over the example used here.

The local search step of the algorithm was set to operate once every four iterations, only for the best current solution. Also, it was configured not to operate if the number of samples in $\mathcal{V}(p^{(j)})$ were inferior to $(n_{\text{vars}}+1)(n_{\text{vars}}+2)/2$, which is the minimum number of samples required to build the (i.e., quadratic) nonlinear model [4].

Also, while the HM used requires a binary input, the neural network used by the local search operator does not. Therefore, in order to reduce the dimensionality of the neural network used, the actual variable values of the candidate solutions were used for the local learning and search procedures.

The stop criterion was set to 500 evaluations of the objective function. After the optimization procedure, the hybrid EDA was able to return the solution shown in Fig. 2. This configuration produced a value of the objective function $f = 2.02$ mT, while maintaining energy stored at an acceptable level $E = 184$ MJ and not violating the $B_{\max}$ constraint.

In terms of computational time, it should be noted that the training of the HM statistical model tends to be an expensive procedure, taking much longer than any other component of the optimization algorithm. Therefore, this algorithm is recommended only for: 1) cases of expensive optimization problems, where the evaluation of the objective function demands a computational budget large enough to completely dominate the optimization time; or 2) problems where simpler methods (e.g., methods without statistical learning capabilities) have failed to obtain good results due to heavy correlations among the variables.

VI. CONCLUSION

In this paper we have presented a hybrid optimization algorithm, based on the statistical modeling performed by a Helmholtz machine, coupled with a neural network-based local search operator. The HM is used to produce the next generation of candidate solutions, based on the statistical characteristic of the superior solutions, while the local search operator is responsible for accelerating the convergence of the algorithm, as well as improving the quality of the solutions presented as input vectors to the HM. The use of a neural network to approximate the objective function and constraints in the vicinity of the points selected for local search also contributes to a faster convergence of the algorithm, since the high number of function evaluations needed by the local optimization can be substituted by the computationally cheap evaluations of the approximate model. While the results obtained for the test problem were not among the best available in the literature, this work has shown the possibility of application of the hybrid methodology in electromagnetic design. More testing is needed, however, in order to validate the proposed methodology for the design of electromagnetic devices.

ACKNOWLEDGMENT

This work was supported in part by the Ministry of Education, Culture, Sports, Science and Technology of Japan and by the National Counsel of Technological and Scientific Development of Brazil under CNPq Grant 306910/2006-3.

REFERENCES

[1] J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*. Berlin, Germany: Springer-Verlag, 2006.

[2] G. R. Harik and D. E. Goldberg, "Learning linkage," in *Foundations of Genetic Algorithms*, R. K. Belew and M. D. Vose, Eds. San Francisco, CA: Morgan Kaufmann, 1997, pp. 247–262.

[3] C. W. Ahn, R. S. Ramakrishna, and D. E. Goldberg, "Real-coded Bayesian optimization algorithm," *Studies Fuzziness Soft Comput.*, vol. 192, pp. 51–73, 2006.

[4] F. G. Guimarães, F. Campelo, H. Igarashi, D. A. Lowther, and J. A. Ramírez, "Optimization of cost functions using evolutionary algorithms with local learning and local search," *IEEE Trans. Magn.*, vol. 43, no. 4, pp. 1641–1644, 2007.

[5] P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel, "The Helmholtz machine," *Neural Comput.*, vol. 7, no. 5, pp. 889–904, 1995.

[6] R. M. Neal and P. Dayan, "Factor analysis using delta-rule wake-sleep learning," *Neural Comput.*, vol. 9, no. 8, pp. 1781–1803, 1997.

[7] K. G. Kirby, "A tutorial on Helmholtz Machines," 2007. [Online]. Available: http://www.nku.edu/~kirby/docs/HelmholtzTutorialKoeln.pdf

[8] S.-Y. Shin and B.-T. Zhang, "Bayesian evolutionary algorithms for continuous function optimization," in *Proc. 6th Int. Conf. Parallel Problem Solving From Nature*, Paris, France, 2000, pp. 16–20.

[9] R. Fletcher, *Practical Methods of Optimization*. New York: Wiley, 1987.

[10] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1999.

[11] F. G. Guimarães, F. Campelo, R. R. Saldanha, H. Igarashi, R. H. C. Takahashi, and J. A. Ramírez, "A multiobjective proposal for the TEAM benchmark problem 22," *IEEE Trans. Magn.*, vol. 42, no. 4, pp. 1471–1474, Apr. 2006.