Design Space Exploration using Self-Organizing Map Based Adaptive Sampling

Keiichi Ito[a,b,*], Ivo Couckuyt[a], Roberto d'Ippolito[b], Tom Dhaene[a]

[a]*Ghent University - iMinds, Department of Information Technology (INTEC), Gaston Crommenlaan 8 bus 201, 9050 Ledeberg - Ghent, Belgium*
[b]*Noesis Solutions, Gaston Geenslaan 11 B4, 3001 Leuven, Belgium*

Abstract

In engineering design, a set of potentially competitive designs is conceived in the early part of the design process. The purpose of this research is to help such a process by investigating algorithm that enables approximate identification of a set of inputs of real variables that return desired responses from a function or a computer simulation. We explore sequential or adaptive sampling methods based on Self-Organizing Maps (SOM). The proposed method does not rely on parametrized distributions, and can sample from multi-modal and non-convex distributions. Furthermore, the proposed merit function provides infill characteristics by favoring sampling points that lay farther from existing points. The results indicate that multiple feasible solutions can be efficiently obtained by the new adaptive sampling algorithm. The iterative use of the SOM in density learning to identify feasible or good designs is our new contribution and it shows a very rapid increase in number of feasible solutions to total number of function evaluation ratio. Application examples to planing hull designs (such as in powerboats and seaplanes) indicate the merits of the feasible region approach to observe trends and design rules. Additionally, the well distributed sampling points of the proposed method played favorable effect in improving the prediction performance of a classification problem learned by Support Vector Machine.

Keywords: Optimization, Simulated-Annealing, Self-Organizing Map, Density Learning, Feasible Region

1. Introduction

In today's engineering, it is common practice to use computer simulations to understand the behavior of complex systems and optimize their parameters to obtain satisfactory designs before building actual physical prototypes. The goal of an engineer is not only to optimize the systems, but also to understand what makes a design good. Engineers may also need to deal with simulation models that may not represent all the effects necessary to make a right decision. The question - particularly in the early stage of the design process - is often not about finding the best parameter values, but establishing what parameter values would generate a satisfactory design. At a more pragmatic simulation level, engineers often confine the simulation runs to parameter settings for which results are trustworthy [1]. For example, a simulation may crash or its solution may not converge. Such information may not be available until the simulation is running. Furthermore, there are widespread incentives to reduce the number of simulation runs since high fidelity simulations often require a lot of time and computational resources as evidenced in the research of surrogate model based optimization [2, 3, 4] and multifidelity optimization [5, 6] methods. Our research is motivated by situations in which efficient identification of diverse designs satisfying minimum specifications and understanding about the problem are more important than finding an accurate single optimal solution.

We propose a new adaptive sampling algorithm that uses a Self-Organizing Map (SOM) [7, 8] to perform importance sampling: Self-Organizing Map Based Adaptive Sampling (SOMBAS). The fundamental idea is an algorithm that learns to select new samples in the region of interest, using the density learning mechanism in SOM. It is similar to Monte Carlo Optimization methods such as Probability Collectives [9] and Cross-Entropy Methods [10] or Subset Optimization methods [11, 12]. However, we do not use parameterized probability density functions to represent solutions. Instead, SOM is used to obtain a set of solutions as rep-

*Corresponding Author
Email address: keiichi.ito@intec.ugent.be (Keiichi Ito)

resented by the weight vectors in each cell of the map. SOM represents a densely sampled region as a larger area on the map than a sparsely sampled region. Therefore, a weight vector can be considered as being analogous to an instance of a random vector drawn from a probability density distribution. Furthermore, these vectors are mutated to improve diversity. The training set can be iteratively enriched with, or replaced by, new samples that exhibit desirable responses (or objective values). SOM is retrained in each iteration with the updated training set. To enhance uniform sampling in the region of interest, a new merit function is also proposed. The flowchart of SOMBAS is shown in Fig. 1.

The idea of using the weight vectors as candidate solutions can also be seen in Liukkonen et al. [13]. They train SOM from a set of experiments and look for a best candidate solution from the SOM weight vector. The chosen weight vectors are taken as representing an average solution in their respective neighborhoods of good solutions. However, their method does not entail further sampling, and does not have the density learning notion. Our new method substantially extends the idea by making the search process adaptive (i.e. iterates to further explore good solutions).

A preliminary version of SOMBAS was presented in Ito et al. [14]. Current work investigates the scalability of SOMBAS to high-dimensional problems and application to a conceptual design example giving extra insights of the parameters on the results.

SOMBAS does not entail any modification of SOM just as in Kita et al. [15]. Therefore, different implementations of SOM or other density learning algorithms can be used instead. It focuses on interesting regions of the input space by modifying the sample densities. While Kita et al. use SOM to do clustering (niching), we use SOM to generate new samples according to the density of its training samples. Their paper shows its advantage in multimodal functions and relative weakness in non-separable unimodal functions. Our algorithm, on the other hand, shows no such tendency.

Couckuyt et al. [16] and Gorissen et al. [17] have proposed a sequential sampling approach that samples uniformly from the region of interest specified by upper and lower bounds on the output. They extended the Efficient Global Optimization (EGO) [18] and used prediction variances to determine new sampling points that would likely produce an output in the desired range and away from existing samples. The fundamental difference between their work and this paper is that we do not fit interpolating surrogate models that require optimization of the surrogate model hyperparameters. In SOMBAS, no optimality on SOM training is imposed and a user can specify the number of training iterations. Our novelty is in the application of SOM in adaptive sampling scheme. This enables us to sample from distributions without the need to parametrize them. Furthermore, SOM is scalable to high-dimensional input space.

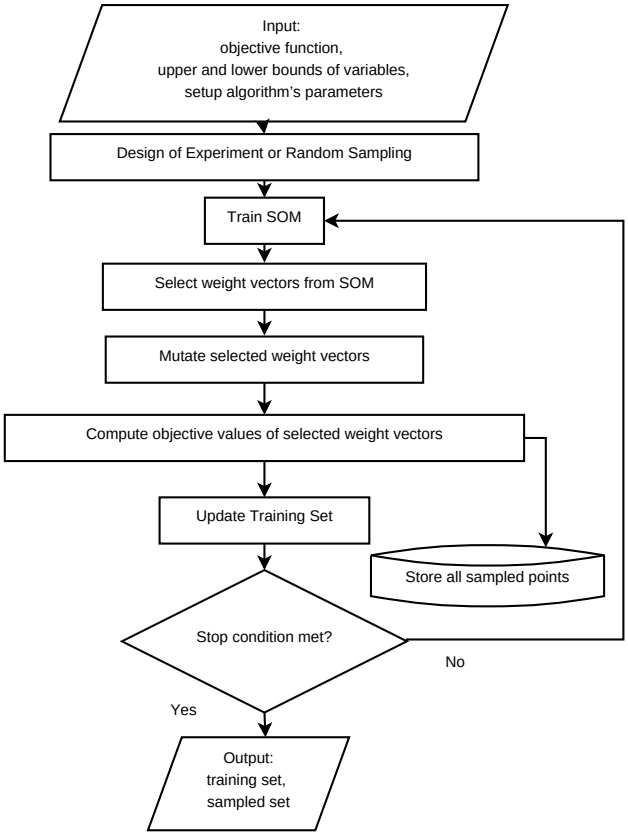


Figure 1: High level flowchart of SOMBAS.

Emmerich et al. [19] and Ulrich and Thiele [20] have proposed algorithms with identical objectives as SOMBAS. They propose diversity measures in their evolutionary algorithms and explicitly optimize for this measures. However, their methods involve multidimensional integrations or matrix inversions that would make the algorithms difficult to apply in high-dimensional problems. In SOMBAS, diversity is kept by a simple merit function that takes the distance to the nearest-neighbor into account and a mutation algorithm.

2. Self-Organizing Map Based Adaptive Sampling (SOMBAS)

We use Self-Organizing Maps'(SOM) weight vectors as a representation of a sample distribution. Typically, SOM is represented as a two-dimensional array of cells

(be it hexagonal or square shaped). Each of these cells has a weight vector associated with it. In this work, the weight vector is a set of continuous design variables that represents an instance of a possible new solution. We initially assign random numbers to the vector elements. Then, the weight vectors are learned from a given set of training samples. The trained weight vectors can be considered to be a finite sample representation of the training sample distribution. The weight vectors $\mathbf{w}_j$ are updated using the following equation for a given training sample $\mathbf{t}$.

$$\mathbf{w}_j(k+1) = \mathbf{w}_j(k) + h_{cj}(k)\Big[\mathbf{t}(k) - \mathbf{w}_j(k)\Big], \qquad (1)$$

where j is a spatial index that identifies the cells in SOM, k is the training iteration index, h_{cj} is a neighborhood function that depends on the distance between $\mathbf{w}_c$ and $\mathbf{w}_j$ on the map where $\mathbf{w}_c$ is the closest weight vector to the training sample $\mathbf{t}(k)$ in the Euclidean sense. The neighborhood function decreases as the distance between the cells becomes far apart on the map. Thus, given a training sample and the closest matching weight vector, the farther cells on the map receive less influence of the weight update. The shape and magnitude of $h_{cj}(k)$ are changed as k increases in such a way that the second term (the weight update term) on the right-hand side of equation (1) reduces the radius and magnitude of influence.

Algorithm 1 shows a high-level description of the Self-Organizing Map Based Adaptive Sampling. In each iteration, the trained Self-Organizing Map (SOM) produces new solution candidates and their corresponding objective values. Weight vector selection is based on these estimated values. Perturbations are applied to these selected vectors, and their objective values are computed, replacing the estimated values. Updating of the training set is performed and a subset of these selected samples are included in the training set to train the SOM of next iteration.

The probability of a weight vector being selected depends on how close its objective value estimate is to the known smallest value. Note that the objective values in the weight vectors of SOM are estimates. The selection condition is

$$r < \exp\left(\frac{y_{\min} - \hat{y}}{T}\right), \qquad (2)$$

where $0 \le r < 1$ is a random number drawn from a uniform distribution, $y_{\min}$ is the smallest output in the training sample, and $\hat{y}$ is the estimated objective value from the weight vector. The temperature $0.01 \le T \le 10$ defines how selective the condition is and a smaller value of T results in fewer new samples added to the training data set. The pseudocode of this selection condition is given in Algorithm 2.

Algorithm 1 SOM BASED ADAPTIVE SAMPLING

1: Generate N samples to create initial training set
2: **while** Termination condition not met **do**
3: Train SOM using the normalized training set
4: **for all** cells satisfying SELECTION CONDITION **do**
5: Perturb the weight vectors of the selected cells according to MUTATION
6: **end for**
7: Un-normalize the perturbed samples
8: Evaluate true output of the perturbed and unperturbed samples
9: UPDATE TRAINING SET
10: **end while**

Algorithm 2 SELECTION CONDITION

1: Let $y_{\min}$ be the smallest output in the training set X, $\hat{y}$ be output from a cell weight vector, and T be the selectivity parameter (or Temperature)
2: Generate a uniform random number $0 \le r < 1$ and check the following:
3: **if** $r < \exp\left(\frac{y_{\min} - \hat{y}}{T}\right)$ **then**
4: Corresponding weight vector is selected
5: **end if**

We consider a case in which we seek to minimize an objective value y below certain threshold L. Below this threshold, diversity of solutions is sought. In this paper, we will call such a search as *feasible region identification* or *feasible region search*. One idea is to use a merit function similar to those described in Torczon and Trosset [21]. One could give a better chance of being selected to points (i.e. cell weight vector) that are distant from existing training samples regardless of y value. To achieve this, we propose the following formula for the merit function.

$$F = \max(L, y) - \rho \min(\|\mathbf{s} - \mathbf{t}_i\|_2),\ i = 1, 2, \cdots, N \qquad (3)$$

where $\mathbf{s}$ is the input vector for which F needs to be minimized, $\mathbf{t}_i$ is a set of target samples from which minimum distance to the input vector $\mathbf{s}$ is calculated, N is the number of such target vectors, and ρ is a weight constant. Unlike Torczon's merit function, our merit function incorporates a "truncation" value L below which only the separation from other target vectors $\mathbf{t}_i$ matters. To minimize this merit function, one needs $y < L$ and maximize

the distance to the nearest target vector $\min(\|\mathbf{s}-\mathbf{t}_i\|_2)$. In our case, target vectors are the training set and the input vector $\mathbf{s}$ is the selected weight vector from SOM. The algorithm to replace the output with this merit function is described in Algorithm 3. If $\hat{y}$ is greater than the threshold L, both $\hat{y}$ and the new weight vector's distance from the training set are taken into account. If $\hat{y}$ is less than L, then the distance to the nearest training vector is the only term affecting the objective value and smaller F is obtained when the weight vector's distance to the nearest neighbor is larger. The ρ in equation 3 is a positive weighing constant

Algorithm 3 MERIT FUNCTION

1: Let L denote the value below which objective or output y is considered to be "good enough", $\mathbf{s}$ denote a weight vector $(\mathbf{x}^T, \hat{y})$ from SOM, and $\mathbf{t}_{i=1,2,\ldots,N}$ denote the training samples
2: **if** Trunc is specified **then**
3: Normalize L ($\mathbf{s}$, and $\mathbf{t}_i$ are already normalized)
4: $\hat{y} \leftarrow \max(L, \hat{y}) - \rho \min(\|\mathbf{s} - \mathbf{t}_i\|_2)$
5: Normalize $\hat{y}$
6: **end if**
7: Use this $\hat{y}$ in SELECTION CONDITION

Mutation, as described in Algorithm 4, is applied to the selected weight vectors. We use the weight vectors as the centers of multivariate Gaussian distributions. The covariance matrix is obtained from the selected weight vectors. We use an idea similar to CMA-ES [22] to update the covariance matrices. The covariance matrix in the current iteration is combined with the covariance matrix computed in the previous iteration: $0.2C + 0.8C_{old}$. This is to avoid adapting too quickly to a local minimum. On top of that, we multiply a factor which is different whether the previous iteration produced a new minimum or not. If the previous iteration achieved a new minimum, we apply an expansion factor F_e, to which we assign a real value larger than 1. On the other hand, if the previous iteration did not produce a new minimum, we multiply a contraction factor F_c, to which we assign a real value between 0 and 1. The covariance matrix is the same for all the selected weight vectors. Each weight vector is perturbed by sampling from the multivariate Gaussian distribution. Mutation is very important to avoid premature convergence in SOMBAS.

After the perturbation of new samples, the training set is updated. Algorithm 5 and Algorithm 6 are two such methods. Algorithm 5 has a faster convergence but is more prone to lose diversity in the training set prematurely compared to Algorithm 6. In the latter method, if $\max(L, y)$ of the new perturbed sample and that of the randomly selected training sample are the same, the replacement of the selected training sample takes place only if the new perturbed sample has a larger distance to its nearest neighbor than the distance of the training sample to its nearest neighbor. Otherwise, the new perturbed sample replaces the training sample when the new sample has a smaller objective value. The nearest neighbors are searched among all the sampled points. In the next section, we will use Algorithm 6.

Algorithm 4 MUTATION

1: Let F_c be contraction factor and F_e be expansion factor
2: Let P_m be mutation probability
3: Given training samples, compute covariance matrix C, and let the covariance matrix from previous iteration be C_{old}
4: **if** current $y_{\min}$ < previous $y_{\min}$ **then**
5: $C = F_e\,(0.2C + 0.8C_{\text{old}})$
6: **else**
7: $C = F_c\,(0.2C + 0.8C_{\text{old}})$
8: **end if**
9: For each selected sample, perturb it by sampling from multivariate normal distribution with center at the selected sample with covariance C.
10: Replace a parameter in the selected vectors with the mutated one at probability of P_m

Algorithm 5 UPDATE TRAINING SET 1

1: Add the perturbed samples to the training set
2: **if** Training set sample size larger than maximum sample size **then**
3: Sort the training set with respect to output value
4: Remove the worst samples to make the training sample size equal to maximum sample size
5: **end if**

3. Experiments

3.1. Feasible Region Identification

In this subsection, we consider a constraint satisfaction problem in which there is a constraint on the objective (to be below a certain threshold value) but one would like to know what inputs would satisfy this condition. Ideally, one would like to have as much variety as possible in the collection of inputs that we obtain as solutions.

Algorithm 6 UPDATE TRAINING SET 2

1: **for all** perturbed weight vectors' response y_p **do**
2: Randomly pick one of the training sample, and obtain its response y_t
3: Obtain d_p, the nearest neighbor distance of perturbed sample to sampled points thus far and d_t the nearest neighbor distance of the training sample to sampled points thus far
4: **if** $\max(L, y_p) = \max(L, y_t)$ and $d_p > d_t$ **then**
5: Replace the training sample with the perturbed weight vector
6: **else if** $y_p < y_t$ **then**
7: Replace the training sample with the perturbed weight vector
8: **end if**
9: **end for**

The analysis and evaluation of the results are not straightforward because we do not have established performance measures. Solow and Polasky [23] and Emmerich et al. [19] show some possibilities, but they do not scale well to high-dimensional problems or can cause numerical problems in matrix inversion. We first resort to visual cues, and then propose some performance measures.

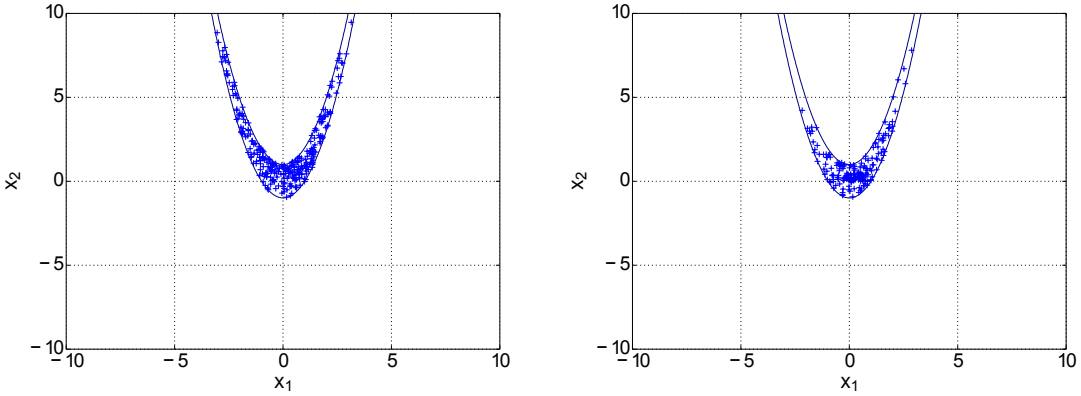
We use the two dimensional Rosenbrock function for non-convex region identification, the two dimensional Rastrigin function for discrete region identification, and the Hollow Beam [24, p. 35] problem with two design variables for feasible region with sharp corners. Then we look into 30 and 100 dimensional Rosenbrock and Rastrigin functions to see the scalability of SOMBAS to high-dimensional problems.

SOMBAS is compared with Differential Evolution (DE). DE learns from the distribution of its population to choose the next sampling points. It does not rely on any parameterized model of the distribution, but the difference vectors adapt to the objective function landscape. Price et al. [25, pp. 44 - 47] called this property *contour matching* and described it as one of the advantages of DE. Furthermore, it is not restricted to low dimensional problems as in Emmerich et al. [19]. Therefore, DE is suited for the identification of the input regions of the functions described above. The purpose of the comparison is to elucidate differences between the two algorithms that have similar characteristics, but serve different purposes, namely optimization and feasible region identification.

In Fig. 2 through Fig. 4, we visualized different types of feasible domains and the sampling (shown in cross) inside them. The contour plots show the boundaries of the domains. SOMBAS was able to produce a fairly uniform infill of samples in the 2D input domain for the functions tested. Since DE also has distribution learning characteristics, it did very well in the feasible domain infill task.

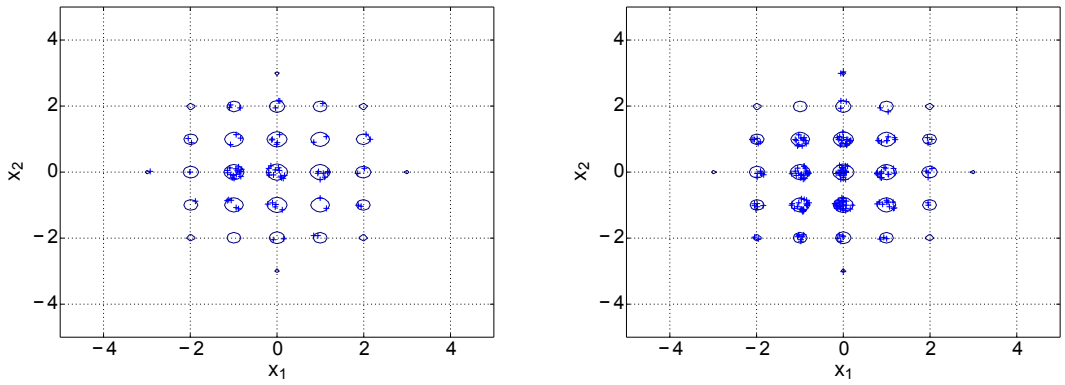
To base feasible region identification on a more statistical ground, we repeated the sampling task for each of the three equations 20 times. We terminated the sampling when all the training sample achieved the objective value $f \leq L$, where $L = 100$ for Rosenbrock, $L = 10$ for Rastrigin, and $L = 0$ for Hollow Beam. Table 1 shows the results. Here, $\tilde{\tilde{d}}$ is the average distance of nearest neighbors. The tilde on top of the symbol signifies averaging and the nearest neighbor has two tildes corresponding to the average in the feasible domain and an average of 20 runs. The $\tilde{N}_f$ is the average number of function evaluations, $\tilde{N}_s$ is the average number of samples in the feasible domain, $\tilde{\sigma}_d$ is the average standard deviation of distances to the nearest neighbors. We also define the feasible rate $\tilde{N}_s/\tilde{N}_f$, coverage length $\tilde{l} = \tilde{\tilde{d}} \times \tilde{N}_s$ and coverage rate $\tilde{l}/\tilde{N}_f$. The feasible rate gives the ratio of the number of feasible samples meeting the truncation value L to the total number of function evaluations. Higher the better. The coverage length gives the efficiency of infill. Higher the better. However, since $\tilde{\tilde{d}}$ is scale dependent, the relative importance of having small N_s but large $\tilde{\tilde{d}}$ or large N_s but small $\tilde{\tilde{d}}$ will be different from problem to problem. The coverage length is meaningful only when we compare different methods on the same feasible domain identification problem. The coverage rate is defined as the coverage length per function evaluation. The larger the value the better, and it is also scale-dependent. For these two dimensional examples, DE and SOMBAS did not show marked difference in performance. SOMBAS showed some advantage in feasible rate for Hollow Beam and DE showed some advantage in Rastrigin in terms of coverage length.

In higher dimensional problems, it is often the case that there is no feasible solution in the beginning. The algorithm has to search and fill the feasible region. Fig. 5 shows the histories of feasible rates with respect to number of function evaluations N_f. The criteria of feasibility were set to $f \leq 20 \times D$ for Rastrigin function and $f \leq 5000 \times D$ for Rosenbrock function. Both DE and SOMBAS were run 20 times. SOMBAS shows very rapid gains in the feasible rate N_s/N_f compared to DE. Moreover, with the feasibility condition set above, the number of function evaluations to reach a given feasible rate (below 0.6) is about the same for both 30-



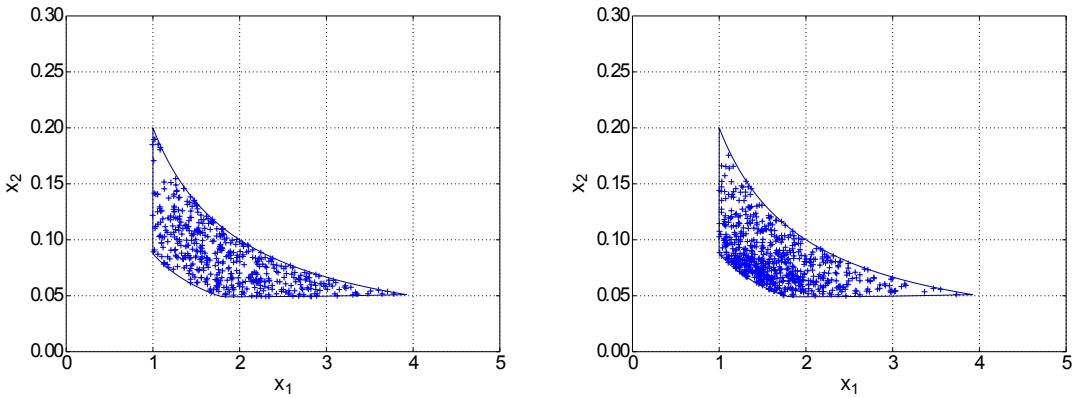
(a) SOMBAS (b) Differential Evolution

Figure 2: Rosenbrock function: sample distribution satisfying objective condition $f \leq 100$.



(a) SOMBAS (b) Differential Evolution

Figure 3: Rastrigin function: sample distribution satisfying objective condition $f \leq 10$.



(a) SOMBAS (b) Differential Evolution

Figure 4: Hollow Beam function: sample distribution satisfying constraints.

Table 1: Average Nearest Neighbor Distances and Space Filling Measures of Sampled Points by SOMBAS and DE.

Function	Algorithm	$\tilde{\tilde{d}}$	$\tilde{\sigma}_d$	$\tilde{N}_s$	$\tilde{N}_f$	$\frac{\tilde{N}_s}{\tilde{N}_f}$	$\tilde{l} = \tilde{\tilde{d}} \times \tilde{N}_s$	$\frac{\tilde{l}}{\tilde{N}_f}$
Rosenbrock	SOMBAS	1.14	0.84	207	786	**0.27**	**227**	0.30
	DE	1.37	1.11	158	689	0.23	213	**0.31**
Rastrigin	SOMBAS	0.38	0.28	146	1136	0.13	53.3	0.05
	DE	0.41	0.31	180	989	**0.18**	**70.4**	**0.07**
Hollow Beam	SOMBAS	0.17	0.11	138	342	**0.40**	**23.1**	**0.07**
	DE	0.20	0.13	112	395	0.28	21.8	0.06

dimensional functions and 100-dimensional functions. However, this is not the case with DE. For DE, the number of function evaluations necessary to reach a given feasible rate seems to be proportional to the number of dimensions. The wiggles on the evolution curves of SOMBAS show that some portion of the sampled points is infeasible and its proportion varies from iteration to iteration and eventually stagnates around certain values of feasible rate N_s/N_f. On the other hand, DE is an optimization algorithm and it keeps searching for lower objective values. Thus, the wiggles (not visible in the plot) disappear once the populations are well within the feasible domain. This, in turn, indicates that for DE the feasible rate N_s/N_f can eventually reach higher rate than SOMBAS as the number of function evaluations is increased.

3.2. Engineering Application

In this subsection, we present an application example of SOMBAS using a simulation code for stability analysis of planing crafts [26, 27, 28]. This example uses SOMBAS in a very simplified simulation based design task. A boat or a seaplane is in a planing condition when it is traveling on water and most of the lifting force of the water comes from hydrodynamic pressure exerted on its hull, buoyancy playing minor to negligible role in its steady state equilibrium. In such condition, the craft may be subject to a dangerous longitudinal oscillation mode called porpoising. Porpoising is a coupled oscillatory motion between heaving and pitching that can manifest when seaplanes and power boats are traveling on water at planing speed. This motion, once manifested, is often unstable and can pose a significant risk to the safe operation of these vehicles.

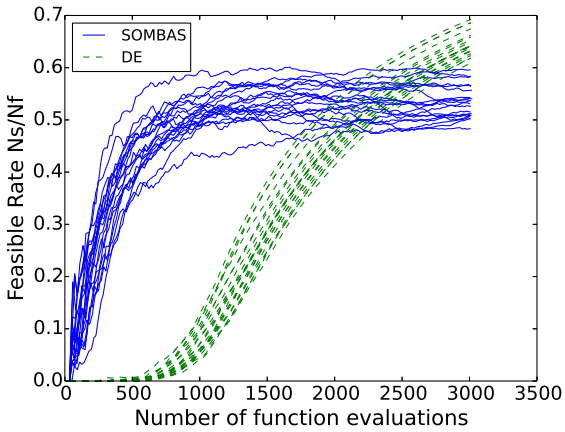
We numerically simulated a 1/3 scale towed tank model as reported in [26]. The model was a catamaran, so we employed the half body representation for simplicity. The model was parameterized as a two or seven variable design problem. The dynamic stability was computed using small perturbation analysis as presented in Faltinsen's book [27, chap. 9]. The two degrees of freedom dynamics - pitching and heaving - were thus represented as a couple of second order differential equations with respect to time. Then, we use a state-space formulation to represent this dynamical system as a system of first order differential equations,

$$\dot{x} = Kx, \tag{4}$$

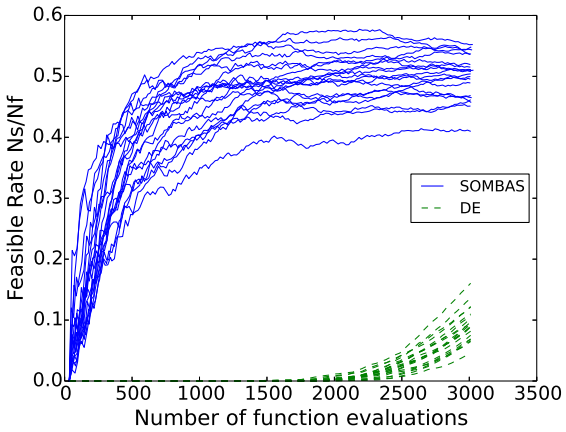
where $x = [\dot{\eta}_3, \dot{\eta}_5, \eta_3, \eta_5]^T$ and η_3 is a displacement upward and η_5 is a pitch-up rotation. The dot above denotes time derivative. The K is a 4×4 matrix and contains information about the inertia, damping, and force reactions. By computing the eigenvalues of this matrix we obtain the stability of this dynamical system. If one of the eigenvalues has positive real part, it means the system has an unstable oscillation mode.

In this example, we try to seek a stable design at a given planing speed by varying the design variables. For the two-design-variable case, we use the longitudinal distance of CG along the keel line l_{cg} measured from the step or transom, and vertical distance of CG from the keel line *vcg*. For the seven-design-variable case we use beam length B, deadrise angle β (in degrees), pitching moment of inertia I_{55},thrust line distance f from CG (positive when pitch-up moment results) and thrust line angle with respect to keel line (positive upwards) ϵ. Fig. 6 shows a diagram describing the design variables except the inertial variable I_{55}. For each design, a trim position needs to be calculated by an iterative root finding process and then Ks are determined via semi-empirical equations based on the trim position. Thus, the mapping from design variables to maximum eigenvalues is non-linear and non-analytical involving loops and branching in the algorithm of the function. This is a typical situation in engineering applications.

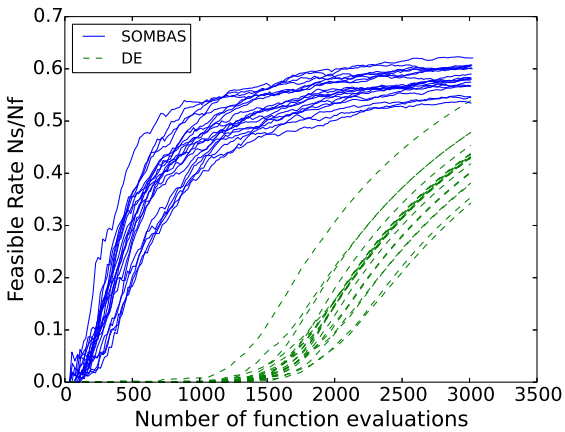
Fig. 7 shows a contour plot of the largest real part of non-dimensionalized eigenvalues with respect to l_{cg}



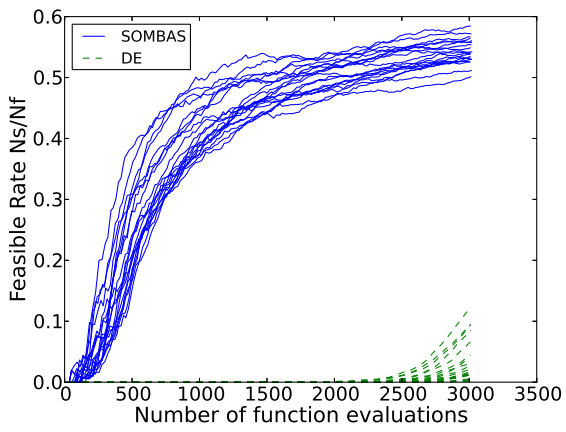
(a) Rastrigin 30 dimensions, feasible solution as $f \leq 600$



(b) Rastrigin 100 dimensions, feasible solution as $f \leq 2000$



(c) Rosenbrock 30 dimensions, feasible solution as $f \leq 150000$



(d) Rosenbrock 100 dimensions, feasible solution as $f \leq 500000$

Figure 5: Evolution of Feasible Rate N_s/N_f of SOMBAS and DE on test functions.

and *vcg* along with sampled points by SOMBAS in two-design-variable case. SOMBAS was set to search feasible designs by setting $L = 0$ in our merit function. That is, if the largest eigenvalue was less than 0, the design was considered stable and thus satisfactory. The sampled points that satisfy $L < 0$ are shown along with the final location of the training samples for SOM. One can see that there is an interval of l_{cg} that produces unstable designs. There is also a large domain that is stable. A small portion of the design space near the transom or very small value of l_{cg} generates stable designs, and most seaplanes have this configuration to facilitate the pitch up at the moment of take off.

Fig. 8a shows a scatter plot matrix of the seven design variable case. The lower triangular cells show the absolute values of correlation coefficients. Again, it clearly shows the unstable "band" for l_{cg} at the top row of the scatter plot matrix. Other parameters do not show clear unfeasible regions. Further restriction was applied by setting $L = -0.3$ and the results are shown in Fig. 8b. It shows some interesting trend. For example, *vcg* tends to lower value as the eigenvalue becomes more negative. On the other hand, the beam length B tends to larger values as the eigenvalue becomes more negative. In the current setup, the deadrise angle β shows a positive correlation with f. Likewise, l_{cg} and B show a positive correlation.

3.3. Machine Learning Application

We continue with the above example, but this time, would like to fit a classifier on top of the sampled points. If a classifier is constructed, a new point's feasibility

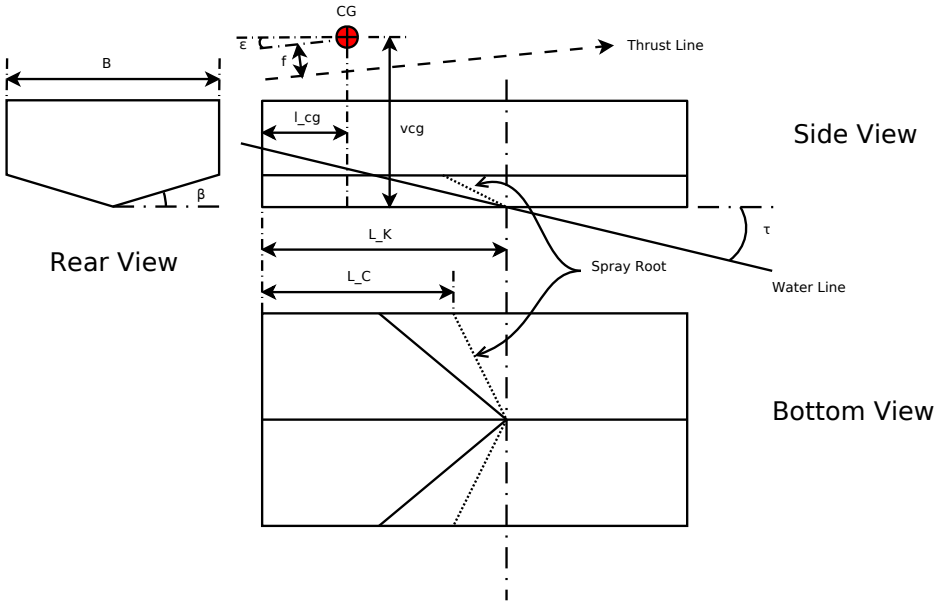


Figure 6: Diagram of planing hull cut-out.

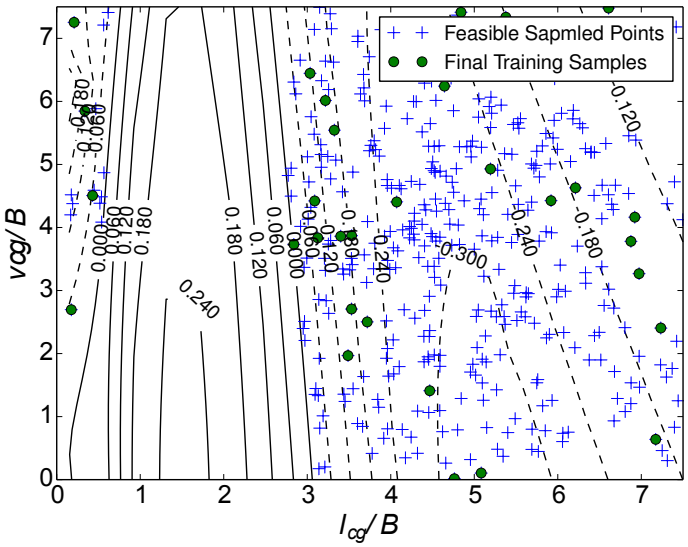
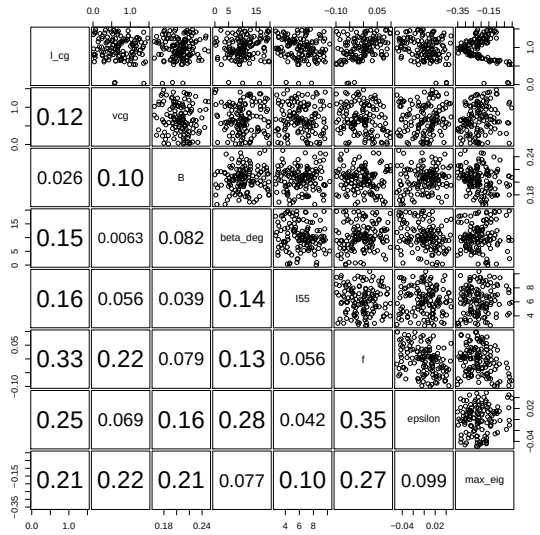
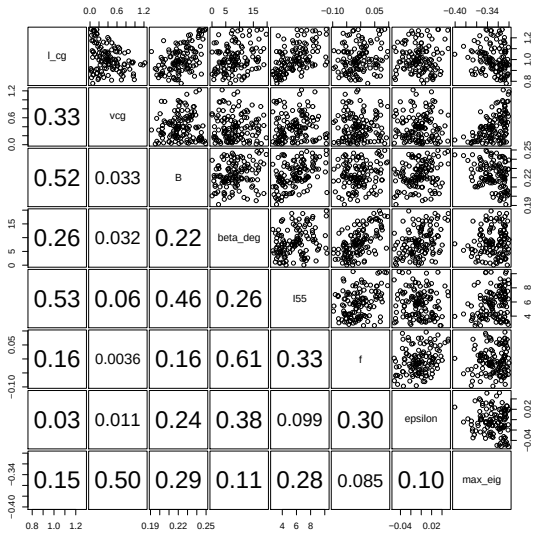


Figure 7: Contour and scatter plot of the real part of the largest eigenvalues of oscillation modes (negative values indicate stable modes) with respect to l_{cg} and *vcg*, both non-dimensionalized with respect to *B*.



(a) Maximum eigenvalues of oscillation modes less than 0, (N_f = 170, N_s = 132)



(b) Maximum eigenvalues of oscillation modes less than −0.3, (N_f = 442, N_s = 123)

Figure 8: Scatter Matrix showing distribution of feasible designs.

can be predicted without evaluating the original (possibly expensive) function. The planing stability example in the previous subsection can be considered as a binary classification problem once enough number of design points are sampled. We employed Support Vector Machine (SVM) [29, 30] to learn the classification problem from the points sampled by SOMBAS, DE, and random sampling.

We run the seven-design-variable case searching for solutions with maximum eigenvalues of oscillation modes less than −0.3 for a given number of simulation budget, namely 1000, 2000, and 4000. Let us call the designs satisfying this condition as stable designs. A test set with 2000 designs randomly sampled from the domain is prepared to evaluate the performance of the SVM classifiers. For performance measure, we use the F_1 score, which is defined as follows.

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R}, \tag{5}$$

where, in our case, the precision P is the proportion of stable designs (according to the simulation) among all designs predicted as stable (by a classifier) in the test set, and the recall R is the proportion of designs correctly predicted as stable (by a classifier) among all the stable designs in the test set (according to the simulation). In this measure, too liberal (e.g., $P \simeq 0$, $R \simeq 1$) or too conservative (e.g., $P \simeq 1$, $R \simeq 0$) classifiers get low score values. $F_1 = 1$ means perfect prediction.

In the top row of Fig. 9, we have shown the distribution of F_1 obtained by fitting Support Vector Machines (SVM) to the sampled points from SOMBAS, DE, and random sampling. The box plots of F_1 was computed from 20 independent runs of each of the three sampling methods. The figures 9a, 9b, and 9c show the results of different function evaluation budgets of the planing craft simulation, 1000, 2000, and 4000 respectively. In the bottom row of Fig. 9, we have the box plots of feasibility rates N_s/N_f of the three sampling methods at corresponding sampling budgets.

In Fig. 9, we observe two trends. The first trend is that the F_1 scores for SVM on SOMBAS and random samples increases as sampling budget increases from 1000 to 4000 while the improvement of SVM on DE is rather small if any and the first and the third quartile of the F_1 scores remain between 0.7 and 0.8. The second trend is that, contrary to the F_1 scores, the feasible rates N_s/N_f for DE increases from around 0.5 to around 0.7 while those for SOMBAS increases very little staying around 0.3 and those for random sampling stays practically constant at 0.07.

These two trends are due to the fact that DE is minimum seeking and SOMBAS and random sampling are space filling. Since SOMBAS searches and fills out the feasible region, the feasible rates reached higher values than those of random sampling. On the other hand, being space filling in the feasible domain, SOMBAS

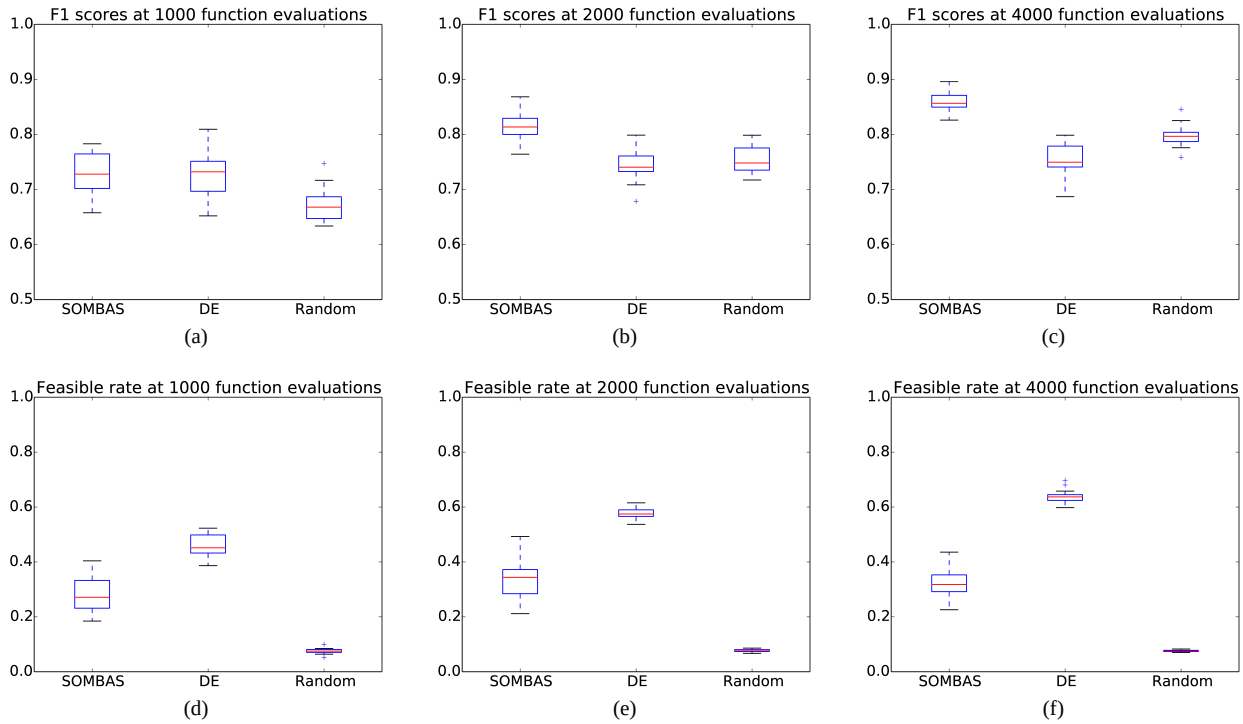


Figure 9: Planing stability prediction performance of Support Vector Machine using samples from SOMBAS and DE. Box plots show the distributions of 20 independent runs at budgets of 1000, 2000, and 4000 function evaluations.

inevitably keeps sampling also from the infeasible domain, because when the mutation happens one does not know if the perturbed points will lie inside the feasible domain. This causes the stagnation of the feasible rate N_s/N_f, but it is beneficial for a classification model as evidenced by superior F_1 scores in Fig. 9b and in Fig. 9c. In principle, the F_1 score of SVM using random sampling should eventually catch up with that of SOMBAS as the number of sampled points is increased. On the other hand, DE (or any optimization method) keeps searching for the smaller response and the sampling concentrates around the points with minimal responses. Since this trend does not help in defining the boundary between feasible and infeasible, the F_1 score stagnates after a certain number of function evaluations, but the feasible rate N_s/N_f will keep increasing if the minimum is in the interior of the feasible domain.

Table 2 to Table 4 summarize the results of the significance of differences in F_1 score and feasible rate N_s/N_f distributions among different sampling methods and sampling budgets in Fig. 9. Wilcoxon Rank Sum Test was used. In this test, the parametric distributions of two random variables are not assumed and sample size of the two variables can be different. It tests whether the distribution of the two random variables, say X and Y, are the same after a translation of size k. That is,

$$P(X < x) = P(Y < x + k) \tag{6}$$

for all x. The null hypothesis is $k = 0$ and the alternative hypothesis is $k \neq 0$.

The hypothesis tests in Table 2 support (at 0.05 significance level) that the shifts in distributions of F_1 scores exist between SVM obtained from SOMBAS and DE in Fig. 9b and in Fig. 9c. The improvement of F_1 scores of random sampling based SVM relative to F_1 scores of DE based SVM (from Fig. 9a to Fig. 9c) is also supported by the significance test.

Table 3 summarizes whether F_1 score distributions in Fig. 9 differ between sampling budgets 1000, 2000, and 4000 function evaluations. For SOMBAS and random sampling, the shifts in the distributions were detected. On the other hand, the null hypothesis was not rejected for DE when the budget was increased from 1000 to 2000 and from 2000 to 4000, although between 1000 and 4000 the alternative hypothesis of $k \neq 0$ was supported. This supports the observation that the increase in F_1 score of SVM using samples from DE is not as rapid as those of the remaining two sampling methods.

Table 4 shows whether the shift in the distribution

Table 2: Hypothesis test of shift in F_1 score distributions in Fig. 9 among different sampling methods (p-values shown in the bracket)

Sampling Methods	1000 eval.	2000 eval.	4000 eval.
SOMBAS vs. DE	Null(0.841)	Alt.($3.407e-07$)	Alt.($1.451e-11$)
SOMBAS vs. Random	Alt.($3.926e-05$)	Alt.($1.281e-06$)	Alt.($1.233e-07$)
DE vs. Random	Alt.($1.831e-05$)	Null(0.2852)	Alt.($1.917e-05$)

Table 3: Hypothesis test of shift in F_1 score distributions in Fig. 9 among different sampling budgets (p-values shown in the bracket)

No. Function Evaluations	SOMBAS	DE	Random
1000 vs. 2000	Alt.($1.407e-09$)	Null(0.1555)	Alt.($2.952e-07$)
2000 vs. 4000	Alt.($1.061e-07$)	Null(0.3104)	Alt.($1.407e-05$)
1000 vs. 4000	Alt.($1.451e-11$)	Alt.(0.03264)	Alt.($6.786e-08$)

of the feasible rate N_s/N_f exists between different sampling budgets. For SOMBAS null hypothesis was kept between 2000 and 4000 function evaluations. For random sampling, no shifts in the distributions were detected among the three sampling budgets. On the other hand, shifts were supported for all the three comparisons for DE. This supports along with Fig. 9 that DE's feasible rate N_s/N_f kept increasing when the sampling budget increased. On the other hand, the feasible rate for SOMBAS stagnated and that of the random sample showed no shift in distribution (which was expected from the law of large numbers).

4. Conclusions

SOMBAS is able to select samples in the design space below a given threshold value, and in addition, it is able to do so in a space-filling way. Our approach to feasible region identification is different from binary classification methods in Machine Learning. Classification methods require both positive and negative samples from the outset of the learning iteration. SOMBAS, on the other hand, will search for feasible regions, even if all of the initial training samples were unfeasible.

SOMBAS' efficient acquisition of feasible solutions in higher dimensions, namely for the 30 and 100-dimensional Rastrigin function and Rosenbrock function, was shown to be superior to DE. It can be argued that feasible region identification becomes identical to optimization when the feasible region becomes infinitesimally small. For example, we could set $f \leq 10^{-6}$ as the feasible region in the 30-dimensional Rastrigin function. In this case, DE would be a better choice. Further research is needed to understand the relationship between accurately finding an optimum point and efficiently identifying a feasible region.

In the engineering example, we identified input values that generate satisfactory designs. By looking at multiple solutions, it enabled the extraction of design knowledge regarding how the design parameters interact under certain stability criteria. This is a significant advantage with respect to standard optimization techniques.

In the application SOMBAS in Machine Learning example, in which Support Vector Machine was used to learn a binary classification model from the sampled data, the accuracy of prediction improved as the number of samples increased and the number of feasible samples for a given function evaluation budget was substantially higher than the random sampling. On the other hand, DE achieved a steady increase in the proportion of number feasible samples (feasible rate N_s/N_f) while the accuracy of prediction (F_1 score) stagnated as the number of data increased. It would be beneficial to investigate the merit of applying SOMBAS to different Machine Learning tasks and methods.

5. Acknowledgments

Keiichi Ito has been funded by the Institute for the Promotion of Innovation through Science and Technology (IWT) through the Baekeland Mandate program. Ivo Couckuyt is a post-doctoral research fellow of the Research Foundation Flanders (FWO). This research has also been funded by the Interuniversity Attraction Poles Programme BESTCOM initiated by the Belgian Science Policy Office.

Appendix A. Test Functions

In the following, we describe the test functions used in this paper. The θ^* denotes the globally optimum so-

Table 4: Hypothesis test of shift in the feasible rate N_s/N_f distributions in Fig. 9 among different sampling budgets (p-values shown in the bracket)

No. Function Evaluations	SOMBAS	DE	Random
1000 vs. 2000	Alt.(0.01674)	Alt.($6.767e-08$)	Null(0.6259)
2000 vs. 4000	Null(0.6017)	Alt.($1.427e-07$)	Null(0.1675)
1000 vs. 4000	Alt.(0.04018)	Alt.($6.767e-08$)	Null(0.8497)

lution vector, and $f(\theta^*)$ its response value. The D denotes the number of dimensions. The upper and lower bounds of θ s are by default $-10 \leq \theta_j \leq 10$ where $\theta = [\theta_0, \theta_1, \ldots, \theta_{D-1}]^T$.

Rosenbrock

$$f(\theta) = \sum_{j=0}^{D-2} \Big(100(\theta_{j+1} - \theta_j^2)^2 + (\theta_j - 1)^2\Big), \qquad (A.1)$$
$$j = 0, 1, \ldots, D-1, \quad D > 1,$$
$$f(\theta^*) = 0, \quad \theta_j^* = 1.$$

Rastrigin

$$f(\theta) = \sum_{j=0}^{D-1} \Big(\theta_j^2 - 10\cos(2\pi\theta_j) + 10\Big), \qquad (A.2)$$
$$j = 0, 1, \ldots, D-1,$$
$$f(\theta^*) = 0, \quad \theta_j^* = 0.$$

Hollow Beam

Let

$$\begin{aligned} w &= 88.9\theta_0\theta_1 - 17.8, \\ g1 &= 0.0885 - \theta_0\theta_1, \\ g2 &= 0.994 - \theta_0, \\ g3 &= 0.05 - \theta_1. \end{aligned}$$

If $g1 > 0$ or $g2 > 0$ or $g3 > 0$,

$$f(\theta) = \max(0, g1) + \max(0, g2) + \max(0, g3), \qquad (A.3)$$

else,

$$f(\theta) = w. \qquad (A.4)$$

For this problem, the objective is to find $\theta = [\theta_1, \theta_2]^T$ such that $f(\theta) < 0$. The lower and upper bounds of θs for this problem are $0 < \theta_0 \leq 5$, $0 < \theta_1 \leq 0.3$.

Appendix B. Parameter Setups

In the following tables, column name NP signifies number of population in DE and NT signifies number of training samples for Self-Organizing Maps in SOMBAS. F and CR are scale factor and cross-over probability as typically defined for the classical DE [25, pp. 38,39]. The number of iteration for the Self-Organizing Map was set between 10 and 40 with no appreciable effect on the results whether one set the number to 10 or 40. The parameter setups of DE and SOMBAS for the feasible region identification in Table 1 are given in Table B.5 and Table B.6 respectively. For SOMBAS, the parameters were set up such that the number of feasible solutions N_f would be more or less the same as those of DE. Fig. 5 was created with with the setups described in Table B.7 for DE and Table B.8 for SOMBAS. In the engineering example of planing craft stability, we setup SOMBAS as $L = 0$, or $-0.3, \rho = 0.1, NT = 36$, SOM size $= 6 \times 6, T = 1, P_{mutation} = 1, F_e = 2.0, F_c = 0.75$.

Table B.5: Parameters setups for DE for the three test functions in Table 1

Function	NP	CR	F
Rosenbrock	45	0.9	0.5
Rastrigin	35	0.2	0.8
Hollow Beam	30	0.65	0.75

The subsequent results of SVM classification problem were obtained using SVC function in "scikit-learn" module [29] with Radial Basis Function (RBF) kernel, penalty parameter $C = 10000$ and kernel coefficient $\gamma = 0.5$. The DE and SOMBAS parameters for the Machine Learning problem shown in Fig. 9 are given in Table B.9 and Table B.10 respectively. The random sampling was done using a uniform random number generator in Python.

References

[1] Y. Tenne, A computational intelligence algorithm for simulation-driven optimization problems, Advances in Engineering Software 47 (2012) 62 – 71.

[2] A. J. Keane, P. B. Nair, Computational Approach for Aerospace Design, John Wiley & Sons, 2005.

[3] I. Couckuyt, F. D. Turck, T. Dhaene, D. Gorissen, Automatic surrogate model type selection during the optimization of expensive black-box problems, in: Proceedings of the 2011 Winter Simulation Conference, 2011, pp. 4274 – 4284.

[4] B. Liu, Q. Zhang, G. G. E. Gielen, A gaussian process surrogate model assisted evolutionary algorithm for medium scale expen-

Table B.6: Parameters setups for SOMBAS for the three test functions in Table 1

Function	L	ρ	NT	SOM size	T	$P_{mutation}$	F_e	F_c
Rosenbrock	100	2.0	45	6×6	0.5	1.0	2.0	0.5
Rastrigin	10	2.0	35	6×6	0.5	1.0	2.0	0.5
Hollow Beam	0	2.0	30	6×6	0.5	1.0	2.0	0.5

Table B.7: Parameters setups for DE for Fig. 5

Function	NP	CR	F
Rosenbrock	35	0.9	0.5
Rastrigin	35	0.2	0.8

sive optimization problems, IEEE Transaction on Evolutionary Computation 18 (2014) 180 – 192.

[5] S. Koziel, L. Leifsson, Surrogate-based aerodynamic shape optimization by variable-resolution models, AIAA Journal 51 (2013) 94 – 106.

[6] N. Courrier, P.-A. Boucard, B. Soulier, The use of partially converged simulations in building surrogate models, Advances in Engineering Software 67 (2014) 186–197.

[7] J. Kangas, T. Kohonen, Developments and applications of the self-organizing map and related algorithms, Mathematics and Computers in Simulation 41 (1996) 3 – 12.

[8] T. Kohonen, Essentials of the self-organizing map, Neural Networks 37 (2013) 52–65.

[9] D. Rajnarayan, D. Wolpert, I. Kroo, Optimization under uncertainty using probability collectives, in: 11th AIAA/ISSMO Multidisciplinary Analysis and Optimisation Conference, Portsmouth, Virginia, 2006.

[10] D. P. Kroese, S. Porotsky, R. Y. Rubinstein, The Cross-Entropy method for continuous multi-extremal optimization, Methodology and Computing in Applied Probability 8 (2006) 383 – 407.

[11] A. A. Taflanidis, J. L. Beck, An efficient framework for optimal robust stochastic system design using stochastic simulation, Computer Methods in Applied Mechanics and Engineering 198 (2008) 88 – 101. Computational Methods in Optimization Considering Uncertainties.

[12] A. Taflanidis, J. Beck, Stochastic subset optimization for optimal reliability problems, Probabilistic Engineering Mechanics 23 (2008) 324 – 338. 5th International Conference on Computational Stochastic Mechanics.

[13] M. Liukkonen, E. Havia, H. Leinonen, Y. Hiltunen, Quality-oriented optimization of wave soldering process by using self-organizing maps, Applied Soft Computing 11 (2011) 214 – 220.

[14] K. Ito, T. Dhaene, N. E. Masri, R. d'Ippolito, J. V. de Peer, Self-organizing map based adaptive sampling, in: Proceedings of 5th International Conference on Experiments/Process/System Modeling/Simulation/Optimization (5th IC-EpsMsO), volume II, Athens, Greece, 2013, pp. 504 – 513. ISBN:978-618-80527-2-7 or 978-618-80527-0-3.

[15] E. Kita, S. Kan, Z. Fei, Investigation of self-organizing map for genetic algorithm, Advances in Engineering Software 41 (2010) 148 – 153.

[16] I. Couckuyt, J. Aernouts, D. Deschrijver, F. Turck, T. Dhaene, Identification of quasi-optimal regions in the design space using surrogate modeling, Engineering with Computers 29 (2013) 127–138.

[17] D. Gorissen, K. Crombecq, I. Couckuyt, T. Dhaene, P. Demeester, A surrogate modeling and adaptive sampling toolbox for computer based design, Journal of Machine Learning Research 11 (2010) 2051 – 2055.

[18] D. R. Jones, M. Schonlau, W. J. Welch, Efficient Global Optimization of Expensive Black-Box Functions, Journal of Global Optimization 13 (1998) 455–492.

[19] M. Emmerich, A. H. Deutz, J. Kruisselbrink, On quality indicators for black-box level set approximation, in: EVOLVE- A Bridge between Probability, Set Oriented Numerics and Evolutionary Computation, volume 447 of *Studies in Computational Intelligence*, Springer Berlin Heidelberg, 2013, pp. 157–185.

[20] T. Ulrich, L. Thiele, Maximizing population diversity in single-objective optimization, in: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO '11, ACM, New York, NY, USA, 2011, pp. 641–648. URL: `http://doi.acm.org/10.1145/2001576.2001665`. doi:10.1145/2001576.2001665.

[21] V. Torczon, M. W. Trosset, Using approximations to accelerate engineering design optimization, Technical Report, Institute for Computer Applications in Science and Engineering (ICASE), 1998.

[22] N. Hansen, A. Ostemeier, Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation, in: Proceedings of the 1996 IEEE Conference on Evolutionary Computation, 1996, pp. 312 – 317.

[23] A. R. Solow, S. Polasky, Measuring biological diversity, Environmental and Ecological Statistics 1 (1994) 95–103.

[24] P. Y. Papalambros, D. J. Wilde, Principle of Optimal Design, Cambridge University Press, 2000.

[25] K. Price, R. M. Storn, J. A. Lampinen, Differential Evolution: APractical Approach to Global Optimization, Springer, 2005.

[26] Y. Hirakawa, T. Takayama, A. Kosaki, H. Kikuchi, T. Hirayama, T. Sakurai, Model experiment of a suppression-system for wave impact and porpoising phenomena, Conference Proceedings of The Japan Society of Naval Architects and Ocean Engineers (in Japanese) 3 (2006) 239–242.

[27] O. M. Faltinsen, Hydrodynamics of High-Speed Marine Vehicles, Cambridge University Press, 2005.

[28] K. Ito, Y. Hirakawa, T. Hirayama, T. Sakurai, T. Dhaene, Longitudinal stability augmentation of seaplanes in planing, in: Proceedings of AIAA Modeling and Simulation Technologies Conference (Aviation 2015), AIAA, Dallas, Texas, 2015.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[30] V. N. Vapnik, The Nature of Statistical Learning Theory, 2nd ed., Springer-Verlag, 1995.

Table B.8: Parameters setups for SOMBAS for Fig. 5

Function	L	ρ	NT	SOM size	T	$P_{mutation}$	F_e	F_c
Rosenbrock	$5000 \times D$	1.0	35	6×6	1	1.0	1.2	0.15
Rastrigin	$20 \times D$	1.0	35	6×6	1	1.0	2.0	0.15

Table B.9: Parameters setups for DE for Fig. 9

Function	NP	CR	F
Planing Craft	40	1.0	0.75

Table B.10: Parameters setups for SOMBAS for Fig. 9

Function	L	ρ	NT	SOM size	T	$P_{mutation}$	F_e	F_c
Planing Craft	-0.3	0.2	40	5×5	4	1.0	1.2	1.0