The current issue and full text archive of this journal is available at
www.emeraldinsight.com/0264-4401.htm

Received 31 August 2011
Revised 16 March 2012
Accepted 22 March 2012

Parallel construction of explicit boundaries using support vector machines

Ke Lin
The State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan, China, and

Anirban Basudhar and Samy Missoum
Aerospace and Mechanical Engineering Department, The University of Arizona, Tucson, Arizona, USA

Abstract

Purpose – The purpose of this paper is to present a study of the parallelization of the construction of explicit constraints or limit-state functions using support vector machines. These explicit boundaries have proven to be beneficial for design optimization and reliability assessment, especially for problems with large computational times, discontinuities, or binary outputs. In addition to the study of the parallelization, the objective of this article is also to provide an approach to select the number of processors.

Design/methodology/approach – This article investigates the parallelization in two ways. First, the efficiency of the parallelization is assessed by comparing, over several runs, the number of iterations needed to create an accurate boundary to the number of iterations associated with a theoretical "linear" speedup. Second, by studying these differences, an "appropriate" range of parallel processors can be inferred.

Findings – The parallelization of the construction of explicit boundaries can lead to a markedly reduced computational burden. The study provides an approach to select the number of processors for an optimal use of computational resources.

Originality/value – The construction of explicit boundaries for design optimization and reliability assessment is designed to alleviate many hurdles in these areas. The parallelization of the construction of the boundaries is a much needed study to reinforce the efficacy and efficiency of this approach.

Keywords Explicit design space decomposition, Support vector machines, Parallel processing, Optimum design, Reliability management

Paper type Research paper

1. Introduction

Simulation-based design optimization or reliability assessment is often hampered by the large computational time associated with complex models. In addition, these models often exhibit nonlinear behaviors thus further hampering the computational design process. In particular, the discontinuity of the system responses is a major hurdle. Similarly, binary responses (pass or fail), represent another challenge. Approximation techniques based on surrogates (Mack *et al.*, 2007; Simpson *et al.*, 2001, 2004; Wang and

Engineering Computations: International Journal for Computer-Aided Engineering and Software
Vol. 30 No. 1, 2013
pp. 132-148

0264-4401
DOI 10.1108/02644401311286099

This research was supported in part by the National Science Foundation (award CMMI-1029257). The authors are also thankful to Mr Peng Jiang for his constructive remarks.

Shan, 2007), such as response surfaces (Myers *et al.*, 2009; Roux *et al.*, 1998) or Kriging (Wang and Shan, 2007; Jin *et al.*, 2001) are often used in computational design to reduce the computational cost. However, these techniques might fail in the cases of discontinuous and binary responses.

For these reasons, a method referred to as explicit design space decomposition (EDSD) (Missoum *et al.*, 2007) was developed whereby the boundaries of the failure domain are constructed explicitly. The main feature of EDSD stems from the fact that the explicit boundaries are constructed based on the classification of responses but not their approximation. Therefore, this procedure removes the hurdles due to discontinuities and binary system outputs in approximation-based approaches. It was found that the most flexible tool to construct the explicit boundaries was support vector machines (SVMs) (Vapnik, 1998; Basudhar *et al.*, 2008). In addition, an adaptive sampling strategy was developed to create accurate explicit boundaries (Basudhar and Missoum, 2010). It was further realized that besides the handling of discontinuous and binary problems, the EDSD approach could tackle a large number of constraints or failure modes (Arenbeck *et al.*, 2010). It also provides a framework for the calculation of probabilities of failure and reliability-based design optimization (Arenbeck *et al.*, 2010; Dribusch *et al.*, 2009; Layman *et al.*, 2010).

However, the exisiting EDSD studies based on adaptive sampling are serial in nature (Arenbeck *et al.*, 2010; Basudhar and Missoum, 2008; Dribusch *et al.*, 2009; Layman *et al.*, 2010). In other words, the approximated explicit boundary is updated after the addition of a new sample. The main objective of this article is to study the efficiency of a parallel implementation of the adaptive sampling scheme.

The study is performed by monitoring the number of iterations needed to reach a given accuracy of the explicit boundary as a function of the number of processors. The number of iterations is compared to the ideal case whereby a twofold increase in the number of processors would lead to a twofold decrease in the number of iterations. Based on this ideal case and a given efficiency of the parallel implementation, this article provides an approach to select an "appropriate" range for the number of parallel processors needed without necessarily using all the available processors. It is essential for the reader to note that in an actual design problem with expensive simulations (e.g. a finite element model), the number of iterations is the most important efficiency metric.

The paper is organized as follows. Section 2 provides a background on EDSD, SVMs, and the adaptive sampling strategy. Section 3 presents the proposed parallel implementation and a derivation for the number of processors needed. To illustrate the method, numerical experiments on analytical problems with 3D to 7D are provided in Section 4. An example involving the buckling of a rectangular thin plate is also presented. Finally, Section 5 provides a few conclusions on this study.

2. EDSD and adaptive sampling strategy

2.1 Explicit design space decomposition

EDSD was originally proposed to deal with the optimization and reliability assessment of systems exhibiting highly nonlinear behaviors (Missoum *et al.*, 2007). More specifically, the approach was introduced to deal with well-known hurdles such as nonlinear limit-state functions or constraints, discontinuous responses, and disjoint failure regions. The basic premise of EDSD is to construct an explicit approximation of the failure domain using a classification technique as opposed to the traditional

approximation approach. The explicit boundary is constructed by using SVMs (Vapnik, 1998; Basudhar and Missoum, 2010). The basic steps of EDSD are:

(1) Performing a design of experiments (DOE) to sample the design space. The initial number of samples is usually small. In order to avoid lack or redundancy of information in certain regions of the design space, the choice of DOE requires the samples to be distributed as uniformly as possible. For this purpose, the centroidal Voronoi tesselation (CVT) proposed by Burkardt *et al.* (2002) is of particular interest.

(2) Response evaluation of the system for each DOE sample. For example, the responses may be evaluated using finite element analysis.

(3) Classification of the responses as safe or failure (two classes). Note that for discontinuous responses, a clustering technique needs to be used (Missoum *et al.*, 2007; Basudhar *et al.*, 2008).

(4) Based on the classification information, an explicit boundary delimiting the failure region is constructed using SVMs.

2.2 Support vector machines

2.2.1 Linear decision function. Consider a d-dimensional space and a training set of N samples $\mathbf{x}_i$ with class $y_i = \pm 1$. In the case of linear decision functions, the decision hyperplane is defined by $\mathbf{w}\cdot\mathbf{x} + b = 0$, where $\mathbf{w}$ is the normal vector of the hyperplane and b is the bias. We also define a pair of "support" hyperplanes, whose equations are $\mathbf{w}\cdot\mathbf{x} + b = 1$ and $\mathbf{w}\cdot\mathbf{x} + b = -1$. The distance between the "support" hyperplanes is $2/\|\mathbf{w}\|$.

The optimal separating hyperplane is found by solving an optimization problem which maximizes the geometric margin while ensuring that no point lies between the two "support" hyperplanes. The optimization problem is:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 0 \quad 1 \leq i \leq N \end{aligned} \tag{1}$$

Equation (1) is a convex quadratic programming problem which can be efficiently solved with available optimization packages. As a result, the optimal $\mathbf{w}$, b, and the Lagrange multipliers λ_i at the optimum are obtained. Then, the SVMs decision function can be expressed as:

$$s(\mathbf{x}) = b + \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i \cdot \mathbf{x} \tag{2}$$

2.2.2 Nonlinear decision function. In order to deal with the case where the data cannot be separated by a linear boundary, SVMs are extended to the nonlinear case by mapping the input space into a higher dimensional space, referred to as the feature space where the data are linearly separable.

In the feature space, the new components of a sample $\mathbf{x}$ are given by $(\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_n(\mathbf{x}))$, where ϕ_i is the feature. The nonlinear decision function is obtained by reformulating the linear classification problem which can be rewritten as:

$$s(\mathbf{x}) = b + \sum_{i=1}^{N} \lambda_i y_i K(\mathbf{x}_i, \mathbf{x}) \tag{3}$$

where $K(\mathbf{x}_i, \mathbf{x}) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle$ is the kernel and $\langle \cdot \rangle$ is the inner product.

The most commonly used kernel functions are the polynomial kernel and the Gaussian kernel (Basudhar and Missoum, 2008). The polynomial kernel is defined as $K(\mathbf{x}_i, \mathbf{x}) = (\langle \mathbf{x}_i, \mathbf{x} \rangle + 1)^p$, where p is the degree of the kernel. The Gaussian kernel is defined as:

$$K(\mathbf{x}_i, \mathbf{x}) = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}\|^2}{2\sigma^2}\right),$$

where σ is the width parameter. In this article, the polynomial kernel was adopted.

2.3 Adaptive sampling strategy

Because of the inaccuracy of the initial SVM boundary, an adaptive sampling strategy is used to refine the boundary with a reasonable number of samples. In its basic form, the scheme proposed by Basudhar and Missoum (2010) selects samples located in sparse regions with the highest probability of misclassification by the SVM boundary. Practically, this is implemented by locating the samples on the SVM boundary at maximum minimum distance to existing samples. The corresponding optimization problem is:

$$\begin{aligned} \max_{\mathbf{x}} \quad & \|\mathbf{x} - \mathbf{x}_{nearest}\| \\ \text{s.t.} \quad & s(\mathbf{x}) = 0 \end{aligned} \tag{4}$$

where $\mathbf{x}_{nearest}$ is the nearest (i.e minimum distance) training sample. It is a global nonlinear optimization problem which can be efficiently solved by the global optimization algorithm, such as genetic algorithm (GA) (Basudhar and Missoum, 2008).

2.4 Error criterion

In order to evaluate the error between the approximated and the actual boundary, two distinct error metrics are introduced as described in the following paragraphs. Note that the actual boundary is not known in practical applications. However, these metrics will allow us to perform the study of the parallel implementation and draw the corresponding conclusions.

2.4.1 Error based on a test data set. Based on a set of N_{test} points $\mathbf{x}_{\text{test}}$ with actual (known) labels y_{test}, the error can be calculated as a misclassification fraction:

$$Error = \frac{\text{num}\,(s(\mathbf{x}_{\text{test}}) y_{\text{test}} \leq 0)}{N_{\text{test}}} \tag{5}$$

This error metric is intuitive, easy to calculate, and general for all the problems (Basudhar *et al.*, 2008; Basudhar and Missoum, 2008; Hamel and Ebrary, 2009).

2.4.2 Error based on the polynomial coefficients. Basudhar and Missoum (2010) proposed an error measure based on the polynomial kernel. This error measure, used in

this article, is based on the fact that two polynomials are equal if and only if their coefficients are equal.

For a d-dimensional problem and a polynomial kernel of degree p, the number of coefficients is $\binom{d+p}{p}$. In order to find the coefficients, a set of $\binom{d+p}{p}$. CVT samples are generated. The coefficients are found using the following equation:

$$\beta = \mathbf{Q}^{-1}\mathbf{s} \tag{6}$$

where $\mathbf{s}$ is the array of SVM values for the CVT samples. The ith row of the coefficient matrix $\mathbf{Q}$ is given as:

$$\mathbf{R}_i = \begin{pmatrix} 1 & x_1 & x_2 & \dots & x_d & \dots & x_1^p & (x_1^{p-1}x_2) & \dots & x_d^p \end{pmatrix}\Big|_{\mathrm{x}_i} \tag{7}$$

The polynomial error ε_k is given as:

$$\varepsilon_k = \frac{\|\beta^{(act)} - \beta^{(k)}\|}{\|\beta^{(act)}\|} \tag{8}$$

where, $\beta^{(act)}$ is the vector of polynomial coefficients for the actual boundary. $\beta^{(k)}$ is the vector of the polynomial coefficients at iteration k. Convergence of the approximated boundary is obtained when $\varepsilon_k \leq \varepsilon_0$.

3. Parallel implementation

In this section, we propose a parallel implementation of EDSD and an approach to select the number of parallel processors.

3.1 Parallel implementation of EDSD

The proposed parallel EDSD scheme is based on the premise that the SVM boundary is updated after a given number of adaptive samples, referred to as "max-min" samples, have been selected. In order to obtain the actual class of these samples (safe or not), the corresponding responses (e.g. from a finite element code) are evaluated in parallel.

Note that, because of the need to calculate distances between samples, the positioning of the adaptive samples cannot be performed strictly in parallel. In other words, one sample requires the knowledge of the position of all the samples. However, the calculation of the positions of the samples is computationally light and can be considered as "overhead" compared to the function evaluation required for each sample to determine its class.

3.2 Efficiency metric

The efficiency of the parallel implementation of EDSD is calculated based on the number of iterations required to converge to an accurate explicit boundary. The number of iterations is compared to an ideal speedup (Dhillon and Modha, 2000). The variations between the actual experiments and the ideal cases will then be used to predict the number of processors needed for the parallel implementation. The following notations will be used:

- P is the number of processors.
- ε_k is the polynomial error associated with the SVM boundary.

- N_1 is the number of iterations for convergence using serial EDSD.
- N_{iter} is the general term of the number of iterations to convergence using the parallel EDSD scheme.
- N_P^{theory} and N_P are the ideal and the experimental number of iterations for convergence on P processors, respectively.
- $NT_{initial}$ is the initial training set size.

3.2.1 Iteration-based speedup. The speedup, S_P, is traditionally defined as the ratio of the execution time for the problem using one processor (serial) to the execution time for the same problem based on P processors (Dhillon and Modha, 2000; Xavier and Lyengar, 1998). However, for an actual design problem with expensive simulations (e.g. a finite element model), the number of iterations, is a more practical metric to compare algorithms in the field of design optimization or reliability assessment.

The speedup is now reformulated in terms of the number of iterations. It is defined as the ratio of the number of iterations to convergence for the serial implementation to the number of iterations to convergence for the parallel implementation using P processors:

$$S_P = \frac{N_1}{N_{iter}} \tag{9}$$

where, N_1 and N_{iter} are the numbers of iterations to convergence for the serial implementation (i.e. one processor) and for the parallel implementation, respectively.

Theoretically, when $S_P = P$, the number of iterations will decrease P times if the number of processors is multiplied by P times. Therefore, the theoretical speedup is linear. Figure 1(a) shows the linear speedup, where $P = 1,2,3,4,\ldots, 20$.

Note that in our article, the efficiency is quantified through the speedup. It is therefore different from the conventional "parallel efficiency" defined as the ratio of the speedup to the number of processors. This quantity provides the speedup per processor and is therefore equal to unity in the ideal case. However, in our study, the speedup and its inverse are used to provide a mean to compare the parallel and the

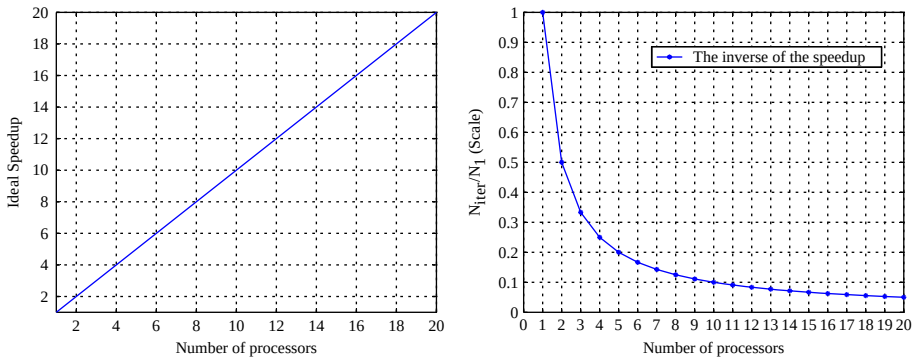


Figure 1.
Linear speedup and the inverse of the speedup

Notes: (a) Ideal linear speedup; (b) the inverse of the speedup

serial implementations as well as a way to select the number of processors. The latter aspect is described in the following section.

3.2.2 The "optimal" number of processors needed. We now turn to the definition of the "optimal" number of processors needed for the parallel implementation. Of course, the word "optimal" is not entirely appropriate as the maximum number of available processors will always provide the largest reduction in computational time. Rather, we are looking for the number of processors beyond which there is only a minor change in efficiency of the parallel implementation. For this purpose, we examine the inverse of the speedup as shown in Figure 1(b), where $P = 1, 2, 3, 4, \ldots, 20$. In our approach, the number of processors needed is based on the "slope" of the curve (i.e. the sensitivity), which is evaluated by the following equation:

$$\frac{d}{dP}\left(\frac{N_{iter}}{N_1}\right) \approx \frac{\left(\left(N_{P+1}^{theory}\right)/N_1\right) - \left(\left(N_P^{theory}\right)/N_1\right)}{(P+1) - P} = \frac{((N_1/(P+1))/N_1) - ((N_1/P)/N_1)}{1} = -\frac{1}{P(P+1)} \tag{10}$$

The number of processors needed is predicted by calculating the ratio of the sensitivity of the number of iterations for P processors to the sensitivity from one to two processors. This ratio is referred to as the relative iteration reduction factor (RIRF) and is defined as:

$$\text{RIRF} = \frac{(d/dP)(N_{iter}/N_1)}{d/dP((N_{iter}/N_1)|_{P=1})} = \frac{2}{P(P+1)} \tag{11}$$

The "optimal" number of processors needed is obtained based on an arbitrary, user selected, value η, so that:

$$\frac{2}{P(P+1)} \leq \eta \tag{12}$$

η represents how the inverse of the speedup "flattens", as shown in Figure 1(b). The "optimal" number of processors needed is then:

$$P^* = \lceil \frac{-1 + \sqrt{(8+\eta)/\eta}}{2} \rceil \tag{13}$$

where $\lceil$ and $\rceil$ signify "round up" to the nearest integer.

3.2.3 Correction factor α. As mentioned above, P^* is obtained based on the assumption of an ideal speedup. However, in general, the number of iterations needed for convergence will be larger. Therefore, the number of required processors corresponding to a RIRF of η will be larger than P^*. In addition, the "max-min" samples are calculated by GA which is a stochastic search method. Therefore, there will be variability in the number of actual iterations.

In order to account for both deviations from the theoretical case, a correction factor α is introduced to define the number of processors needed beyond P^*. The predicted number of processors needed P_{pred} is given as:

$$P_{pred} = (1 + \alpha)P* \tag{14}$$

where α is found experimentally as described in the result Section 4.2.

4. Results

4.1 Description of the test examples

Analytical test examples are used to study the parallel implementation of EDSD. The functions are derived from the same general equation written as a function of the dimensionality d (Basudhar and Missoum, 2010):

$$f(\mathbf{x}) = \sum_{i=1}^{d} (x_i + 2\xi)^2 - 3\sum_{j=1}^{d-2}\prod_{l=j}^{j+2} x_l + 1 \quad \xi = \begin{cases} -1 & mod(i,3) = 1 \\ 0 & mod(i,3) = 2 \\ 1 & mod(i,3) = 0 \end{cases} \tag{15}$$

For instance, for $d = 3$, the problem of equation (15) is to reproduce the following zero iso-value of f:

$$f(\mathbf{x}) = (x_1 - 2)^2 + x_2^2 + (x_3 + 2)^2 - 3x_1x_2x_3 + 1 = 0 \tag{16}$$

Equation (15) represents a non-convex set of disjoint regions. During the sampling scheme, the samples corresponding to $f(\mathbf{x}) > 0$ and $f(\mathbf{x}) < 0$ are labeled $+1$ and -1, respectively. A polynomial kernel is used to construct the SVM boundary in each of the examples. For all examples, starting from small CVT DOEs, the parallelization of EDSD is implemented until $\varepsilon_k \leq 0.01$. The initial numbers of training samples to construct the SVM boundary for each dimensional case are given in Table I.

4.2 Experiments and discussion

As mentioned previously, the basic metric used to assess the efficiency of the parallel implementation of EDSD, which is the focus of this work, is the number of iterations required for convergence of the adaptive sampling scheme. From the number of iterations, a speedup is studied as a function of the number of iterations vs P. Beyond the study of the speedup, this section proposes an approach to find the predicted number of processors needed. For this purpose, we introduce α as described in Section 3, which enables one to quantify how much the actual number of iterations departs from the ideal number of iterations based on linear speedup. In addition, α includes the variability due to the GA used to solve equation (4).

4.2.1 Speedup of the parallel implementation of EDSD. The number of iterations is studied as a function of the number of processors P as well as a function of the dimensionality d. For each experiment (i.e. (P, d) pair), the actual ("experimental") number of iterations N_P is plotted along with the ideal ("theoretical") number of iterations N_P^{theory}, which equals to N_1/P based on equation (9).

Table I. Initial numbers of training samples for 3D though 7D cases

Problem dimension	$d = 3$	$d = 4$	$d = 5$	$d = 6$	$d = 7$
$NT_{initial}$	40	80	160	320	640

The actual and theoretical number of iterations as a function of the number of processors P is depicted for 3D to 7D cases in Figure 2. It is interesting to note that for some instances the parallel implementation might require slightly less iterations than the theoretical case. The 5D case implemented with two processors in parallel provides a good example of this phenomenon. Two "max-min" samples are obtained and evaluated followed by the update of the SVM boundary. In this case, the actual number of iterations is less than the theoretical one.

4.2.2 The quantification of α. Since the experimental iteration numbers deviate from the theoretical one, α is used as a measure of the number of processors needed beyond P^* described in Section 3. Another incentive for using α stems from the fact that the number of iterations for convergence varies from time to time due to the stochastic nature of the GA. In order to assess α, the number of iterations is studied for various dimensions and various P^* calculated from a list of selected ηs as reported in Table II.

In order to quantify the difference between the actual and theoretical number of iterations, we introduce the quantity D_r:

$$D_r = \frac{N_{P^*} - N_{P^*}^{theory}}{N_{P^*}^{theory}} \tag{17}$$

The experiments were carried out as follows: we repeated the 3D example for each P^* five times. For higher dimensions, D_r was calculated once for each P^*. The runs were then repeated five times for P^* with the largest D_r. Performing several runs is important due to the stochastic nature of the GA used to generate the "max-min" samples.

In the course of the parallel implementation on P^* processors, we observed the large variability in the number of iterations for convergence, which leads to the large variability of D_r. The reason for this phenomenon can be attributed to the fact that the given ε_0 is very small. The phenomenon can be understood graphically as shown in Figure 3. The results of the 3D through 7D cases are shown in Figure 4.

The maximum D_r and minimum D_r are plotted in Figure 4(a) through 4(e). The statistics of D_r are provided in Table III.

From the experiments, the range for α is:

$$0 < \alpha \leq 0.5 \tag{18}$$

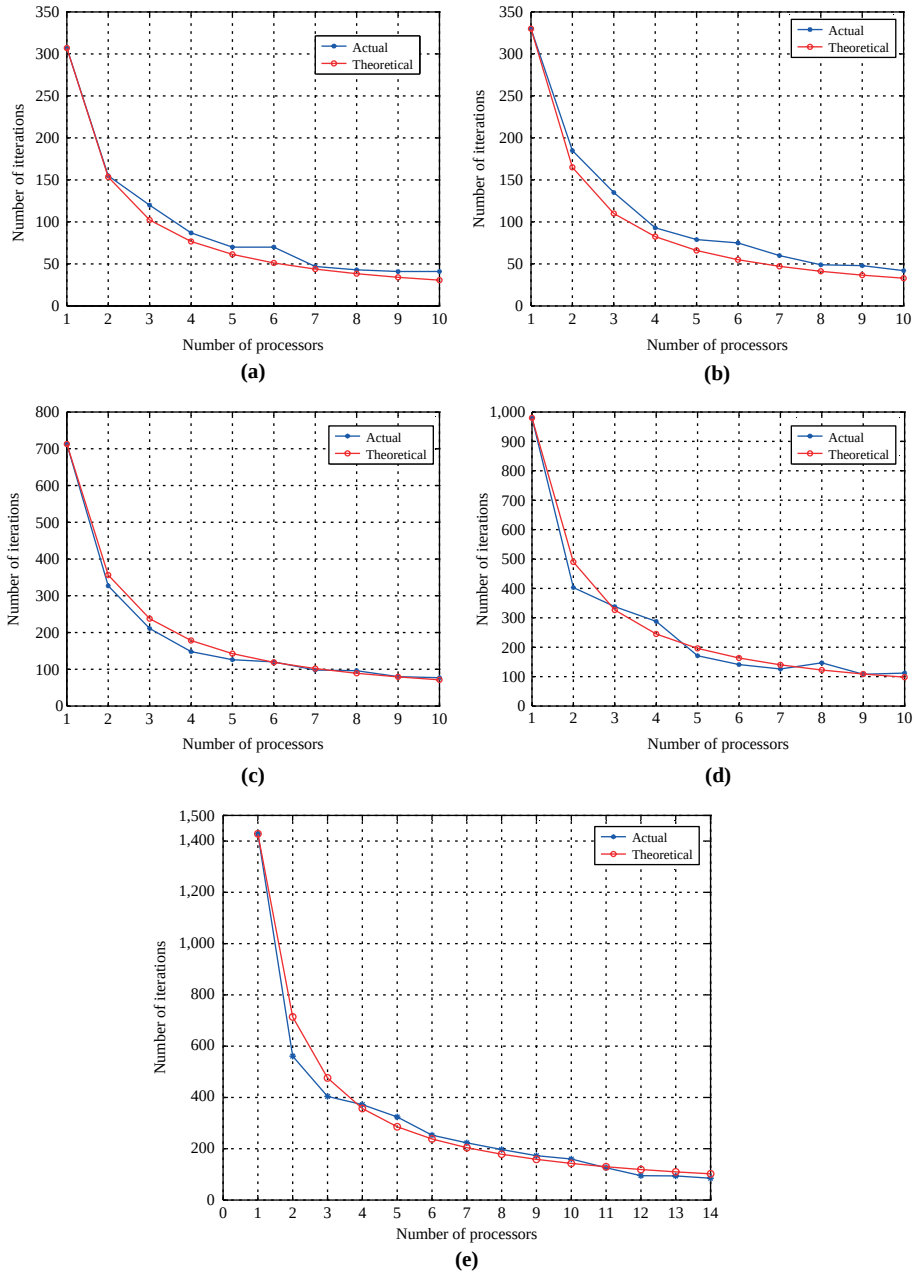
Then, the predicted number of processors needed is bounded as follows:

$$P^* < P_{pred} = (1 + \alpha)P^* \leq 1.5P^* \tag{19}$$

4.3 Validation examples

In this section, analytical example and an application are used to demonstrate the proposed approach.

4.3.1 Analytical example. This function of the example is defined by an analytical function of five variables x_1, x_2, x_3, x_4 and x_5. This analytical function is derived from the fourth test function which has been presented in the article of Jin *et al.* (2001):



Notes: (a) 3D case; (b) 4D case; (c) 5D case; (d) 6D case; (e) 7D case

Figure 2.
The actual number of iterations for convergence (N_P) vs the number of processors (P) as well as the theoretical number of iterations requires to convergence $\left(N_P^{theory}\right)$ vs the number of processors (P) for 3D to 7D cases

$$f(\mathbf{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 45 \quad (20)$$

Consider all the variables are uniformly distributed within the range of [0, 10]. The samples corresponding to $f(\mathbf{x}) > 0$ are labeled + 1, otherwise, labeled as − 1. In total, 160 training samples generated using CVT DOEs are used to construct the initial SVM boundary with a 25.24 percent initial polynomial error. The 500 population size of GA is used.

Table IV provides the results for parallel runs for $\eta = 0.05$ and $\eta = 0.1$. The table shows that the predicted range for the number of processors needed leads to satisfactory number of iterations to convergence.

4.3.2 Linear buckling of the rectangular thin plate. This section provides an example involving the linear buckling of a rectangular thin plate. When a thin plate is subjected to a compressive in-plane load, the plate will buckle at a critical load. The goal is to construct the boundary that separates buckled and non-buckled configurations. This is a binary problem well suited for the EDSD approach.

The rectangular thin plate with length a, width b, and thickness t is simply supported on all edges and under a uniaxial compressive in-plane load N_x in the X direction, as sketched in Figure 5. N_x is a constant load and equals to 5,483 N/mm. The material properties are Young's modulus $E = 210{,}000$ N/mm^2 and Poisson's ratio $\nu = 0.3$. Length a, width b and thickness t are the design variables and given in Table V.

Table II.
The "optimal" numbers of processors needed based on selected ηs

Selected η	0.18	0.1	0.08	0.05	0.04	0.03	0.025	0.02
P^*	3	4	5	6	7	8	9	10

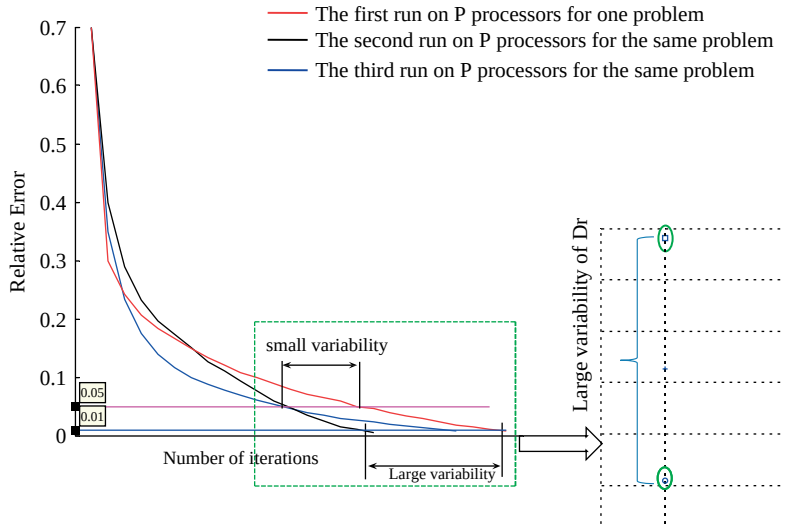
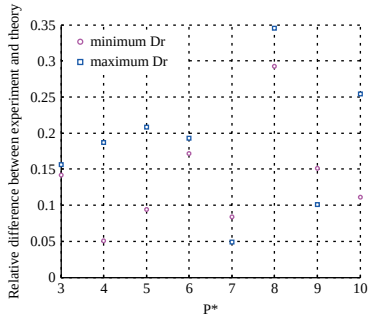
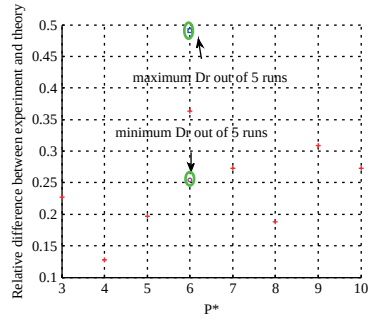


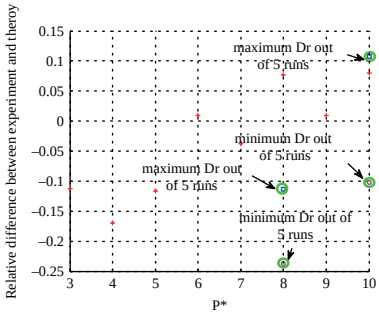
Figure 3.
Schematics of the relationship between convergence criterion and the variability in the number of iterations



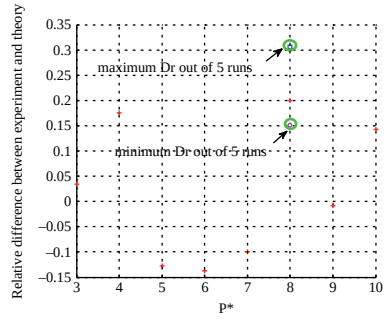
(a) 3 dimension problem, each P^* was repeated five times



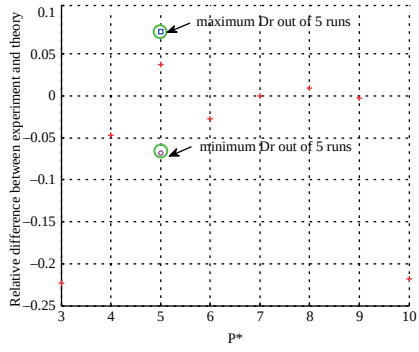
(b) 4 dimension problem. Because D_r of 6 processors was the largest D_r among all the D_rs, we rerun 5 times on 6 processors



(c) 5 dimension problem. Both D_r of 8 processors and D_r of 10 processors were the maximum D_r. We rerun two cases 5 times.



(d) 6 dimension problem. Because D_r of 8 processors was the largest D_r, we rerun the experiment 5 times on 8 processors.



(e) 7dimension problem. Because D_r of 5 processors was the largest D_r among all the D_rs, we rerun 5 times on 5 processors.

Notes: For 3D example, each P^* was repeated five times; for higher dimensions, D_r was calculated once for each P^* (labeled as symbol +); the runs were repeated five times for P^* with the largest D_r

Figure 4.
Relative Difference D_rs for all cases

Based on the theory of plate buckling, the critical buckling load can be written as Vinson (1974):

$$N_{xcr} = -\frac{D\pi^2 a^2}{m^2}\left[\frac{m^2}{a^2}+\frac{n^2}{b^2}\right]^2, \quad D = \frac{Et^3}{12(1-\nu^3)} \tag{21}$$

Table III.
The statistics of D_r associated with the 3D to 7D cases

Problem dimension	$d=3$		$d=4$	$d=5$	$d=6$	$d=7$
Mean	0.2516	0.1002	0.2079	−0.0342	0.0003	−0.0574
Std (Standard deviation)	0.0932	0.0718	0.1060	0.0844	0.1370	0.1351
Mean +2*Std	0.4362	0.2437	0.4199	0.1346	0.2743	0.2128
Mean −2*Std	0.0671	−0.0434	−0.0041	−0.203	−0.2737	−0.3276
Maximum D_r	0.4332		0.4909	0.108	0.3061	0.0762

Table IV.
Results of analytical example

N_1	η	$N_{P^*}^{theory}$	P_{pred}	N_P
286	0.1	$N_1/P^* = 286/4 = 71.5$	$P^* < P_{pred} \leq 1.5P^*$	$N_4 = 91$
			$4 < P_{pred} \leq 6$	$N_5 = 71, N_6 = 65$
	0.05	$N_1/P^* = 286/6 = 47.67$	$P^* < P_{pred} \leq 1.5P^*$	$N_6 = 65$
			$6 < P_{pred} \leq 9$	$N_7 = 51, N_9 = 23$

Notes: N_1, η, the theoretical number of iterations $N_{P^*}^{theory}$, the predicted number of processors needed P_{pred}, and the experimental number of iterations N_P

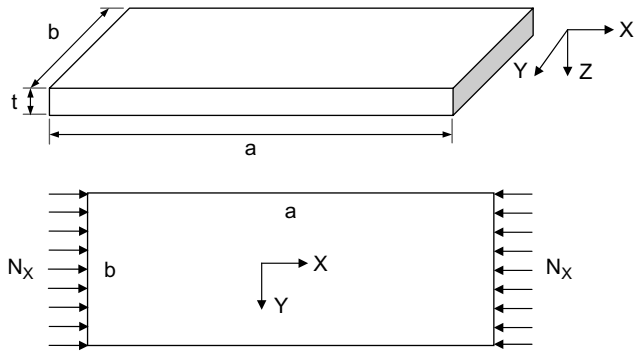


Figure 5.
Rectangular thin plate under in-plate uniaxial compression

Table V.
Design variables for the rectangular thin plate problem

	Thickness (t) mm	Length (a) mm	Width (b) mm
Min. value	3	90	90
Max. value	10	300	300

where D is the plate rigidity, a measure of the flexural stiffness of the plate; m is the number of buckle half-waves in the X direction; and n is the number of buckle half-waves in the Y direction. With $m = n = 1$, the critical buckling load can be rewritten as Vinson (1974):

$$N_{xcr} = -D\pi^2 a^2 \left[\frac{1}{a^2} + \frac{1}{b^2}\right] \tag{22}$$

A SVM boundary is constructed in the (a, b, t) space which separates buckled and non-buckled configurations. The initial SVM boundary, before adaptive sampling, is constructed with ten CVT samples. For this problem, the configurations that have a critical load N_{xcr} lower than 5,483 N/mm will be considered as buckled. For comparison and calculation of the error, an accurate SVM is constructed with 8,000 samples. The classification error is based on a test set of 10^6 points. The initial error is 23.49 percent. Figure 6 shows the buckled and non-buckled configurations with respect to the length, width and thickness. And Figure 7 shows the comparison of the accurate SVM boundary with the final updated SVM boundary.

The results of the parallel runs using 4, 5, 6, 7 and 9 processors are gathered in Table VI. The table includes cases with $\eta = 0.1$ and $\eta = 0.05$. The table demonstrates that the range for the number of processors needed is adequate when compared to the "theoretical" case (second column of the table).

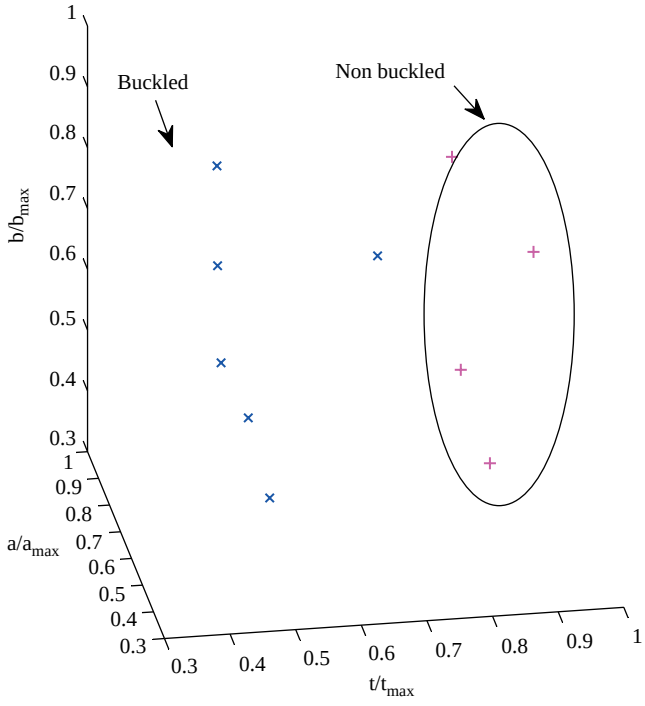


Figure 6.
Buckled and non-buckled configurations with respect to the length, width and thickness

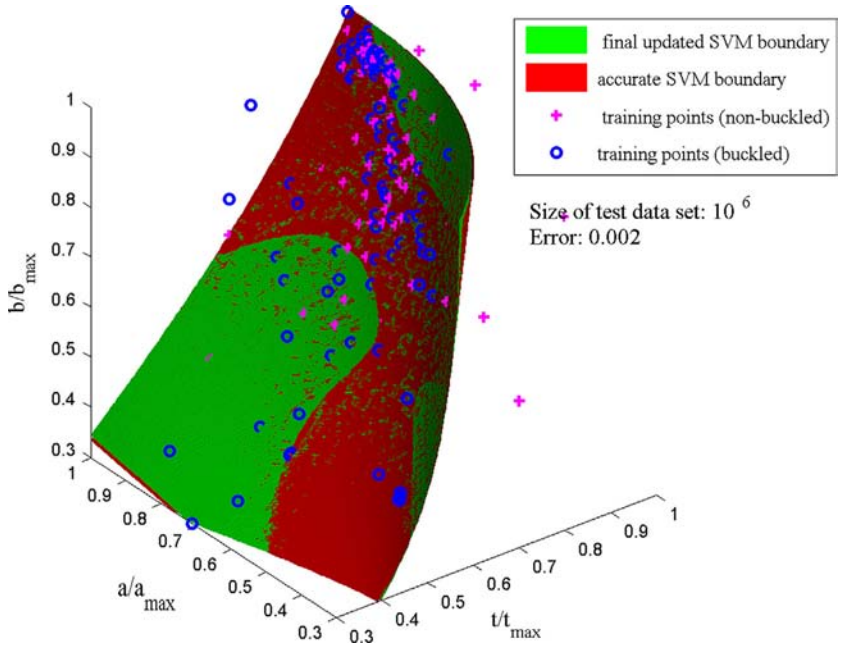


Figure 7.
Comparison of the accurate SVM boundary with the final updated SVM boundary of the rectangular thin plate problem

N_1	η	$N_{P^*}^{theory}$	P_{pred}	N_P
179	0.1	$N_1/P^* = 179/4 = 44.75$	$P^* < P_{pred} \leq 1.5P^*$	$N_4 = 58$
			$4 < P_{pred} \leq 6$	$N_5 = 46, N_6 = 38$
	0.05	$N_1/P^* = 179/6 = 29.83$	$P^* < P_{pred} \leq 1.5P^*$	$N_6 = 38$
			$6 < P_{pred} \leq 9$	$N_7 = 32, N_9 = 24$

Table VI.
Results of the linear buckling of the rectangular thin plate

Notes: N_1, η, the theoretical number of iterations $N_{P^*}^{theory}$, the predicted number of processors needed P_{pred}, and the experimental number of iterations N_P

5. Conclusion

This article presents a study of the parallelization of EDSD approach. This work demonstrates the efficiency of the parallelization. This is implemented by comparing the number of iterations to reach a converged explicit boundary to the theoretical number of iterations required following a linear speedup. By quantifying the differences in number of iterations between the "experimental" and the ideal cases, a range of needed parallel processors is obtained. This range also includes the variations inherent to the GA used in the adaptive sampling scheme. The definition of this range is important in order not to "waste" all the processors for the parallel implementation as those might have a marginal effect on the convergence.

In the proposed work, the load per processor is assumed comparable. However, this might not always be true, and an optimal strategy for load balancing between the processors might be required. This will be the object of a future study.

References

Arenbeck, H., Missoum, S., Basudhar, A. and Nikravesh, P. (2010), "Reliability-based optimal design and tolerancing for multibody systems using explicit design space decomposition", *Journal of Mechanical Design*, Vol. 132 No. 2, pp. 1-11.

Basudhar, A. and Missoum, S. (2008), "Adaptive explicit decision functions for probabilistic design and optimization using support vector machines", *Computers and Structures*, Vol. 86 Nos 19/20, pp. 1904-17.

Basudhar, A. and Missoum, S. (2010), "An improved adaptive sampling scheme for the construction of explicit boundaries", *Structural and Multidisciplinary Optimization*, Vol. 42 No. 4, pp. 1-13.

Basudhar, A., Missoum, S. and Harrison Sanchez, A. (2008), "Limit state function identification using support vector machines for discontinuous responses and disjoint failure domains", *Probabilistic Engineering Mechanics*, Vol. 23 No. 1, pp. 1-11.

Burkardt, J., Gunzburger, M., Peterson, J. and Brannon, R. (2002), *User Manual and Supporting Information for Library of Codes for Centroidal Voronoi Point Placement and Associated Zeroth, First, and Second Moment Determination*, SAND Report SAND2002-0099, Sandia National Laboratories, Albuquerque, NM.

Dhillon, I. and Modha, D. (2000), "A data-clustering algorithm on distributed memory multiprocessors", *Large-Scale Parallel Data Mining*, Vol. 1756, pp. 245-60.

Dribusch, C., Missoum, S. and Beran, P. (2009), "A multidelity approach for the construction of explicit decision boundaries: application to aeroelasticity", *Structural and Multidisciplinary Optimization*, Vol. 42 No. 3, pp. 1-13.

Hamel, L. and Ebrary, I. (2009), *Knowledge Discovery with Support Vector Machines*, Wiley, New York, NY.

Jin, R., Chen, W. and Simpson, T. (2001), "Comparative studies of metamodelling techniques under multiple modelling criteria", *Structural and Multidisciplinary Optimization*, Vol. 23 No. 1, pp. 1-13.

Layman, R., Missoum, S. and Geest, J. (2010), "Simulation and probabilistic failure prediction of grafts for aortic aneurysm", *Engineering Computations*, Vol. 27 No. 1, pp. 84-105.

Mack, Y., Goel, T., Shyy, W. and Haftka, R. (2007), "Surrogate model-based optimization framework: a case study in aerospace design", *Evolutionary Computation in Dynamic and Uncertain Environments Studies in Computational Intelligence*, Vol. 51, pp. 323-42.

Missoum, S., Ramu, P. and Haftka, R. (2007), "A convex hull approach for the reliability-based design optimization of nonlinear transient dynamic problems", *Computer Methods in Applied Mechanics and Engineering*, Vol. 196 Nos 29/30, pp. 2895-906.

Myers, R., Montgomery, D. and Anderson-Cook, C. (2009), *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, Wiley, New York, NY.

Roux, W., Stander, N. and Haftka, R. (1998), "Response surface approximations for structural optimization", *International Journal for Numerical Methods in Engineering*, Vol. 42 No. 3, pp. 517-34.

Simpson, T., Poplinski, J., Koch, P. and Allen, J. (2001), "Metamodels for computer-based engineering design: survey and recommendations", *Engineering with Computers*, Vol. 17 No. 2, pp. 129-50.

Simpson, T., Booker, A., Ghosh, D., Giunta, A., Koch, P. and Yang, R. (2004), "Approximation methods in multidisciplinary analysis and optimization: a panel discussion", *Structural and Multidisciplinary Optimization*, Vol. 27 No. 5, pp. 302-13.

Vapnik, V.N. (1998), *Statistical Learning Theory*, Wiley, New York, NY.

Vinson, J. (1974), *Structural Mechanics: The Behavior of Plates and Shells*, Wiley, New York, NY.

Wang, G. and Shan, S. (2007), "Review of metamodeling techniques in support of engineering design optimization", *Journal of Mechanical Design*, Vol. 129 No. 4, pp. 129-370.

Xavier, C. and Lyengar, S. (1998), *Introduction to Parallel Algorithms*, Wiley, New York, NY.

Further reading

Booker, A., Dennis, J., Frank, P., Serani, D., Torczon, V. and Trosset, M. (1999), "A rigorous framework for optimization of expensive functions by surrogates", *Structural and Multidisciplinary Optimization*, Vol. 17 No. 1, pp. 1-13.

About the authors

Ke Lin now is a PhD candidate of Huazhong University of Science and Technology, School of Mechanical Science and Engineering. His research is in the area of optimization, reliability, and reliability-based design optimization.

Dr Anirban Basudhar was a graduate student in the Aerospace and Mechanical Engineering Department at the University of Arizona and is currently a Software Developer at the Livermore Software Technology Corporation.

Dr Samy Missoum is an Associate Professor in the Aerospace and Mechanical Engineering Department at the University of Arizona. Samy Missoum is the corresponding author and can be contacted at: smissoum@email.arizona.edu