Edgar Galvan
Design Systems Laboratory,
Department of Mechanical Engineering,
Texas A&M University,
College Station, TX 77843
e-mail: e_galvan@tamu.edu

Richard J. Malak[1]
Associate Professor
Design Systems Laboratory,
Department of Mechanical Engineering,
Texas A&M University,
College Station, TX 77843
e-mail: rmalak@tamu.edu

Sean Gibbons
Computational Materials Science Lab,
Department of Materials Science
and Engineering,
Texas A&M University,
College Station, TX 77843
e-mail: sean.l.gibbons@gmail.com

Raymundo Arroyave
Associate Professor
Computational Materials Science Lab,
Department of Materials Science
and Engineering,
Texas A&M University,
College Station, TX 77843
e-mail: rarroyave@tamu.edu

A Constraint Satisfaction Algorithm for the Generalized Inverse Phase Stability Problem

Researchers have used the (calculation of phase diagram) CALPHAD method to solve the forward phase stability problem of mapping from specific thermodynamic conditions (material composition, temperature, pressure, etc.) to the associated phase constitution. Recently, optimization has been used to solve the inverse problem: mapping specific phase constitutions to the thermodynamic conditions that give rise to them. These point-wise results, however, are of limited value since they do not provide information about the forces driving the point to equilibrium. In this paper, we investigate the problem of mapping a desirable region in the phase constitution space to corresponding regions in the space of thermodynamic conditions. We term this problem the generalized inverse phase stability problem (GIPSP) and model the problem as a continuous constraint satisfaction problem (CCSP). In this paper, we propose a new CCSP algorithm tailored for the GIPSP. We investigate the performance of the algorithm on Fe–Ti binary alloy system using ThermoCalc with the TCFE7 database against a related algorithm. The algorithm is able to generate solutions for this problem with high performance.
[DOI: 10.1115/1.4034581]

1 Introduction

The designers of novel materials require an understanding of phase stability in order to assess the feasibility of a material and how it changes during processing. The calculation of phase diagram (CALPHAD) method has enabled researchers to develop databases that contain pertinent thermodynamic information on specific alloys and associated phases [1]. In this method, the thermodynamics of phases are described through mathematical models fitted to experimental data.

Many CALPHAD software packages developed to date are well-suited for solving the forward phase stability problem: calculating the phase stability state of a multicomponent, multiphase system given a set of thermodynamic conditions (processing conditions such as composition, temperature, and pressure). However, because design is inherently an inverse process, we are interested instead in the inverse problem: determining the thermodynamic conditions that give rise to a desired phase stability state.

Previous investigations have used optimization algorithms in the context of inverse phase stability problems to search for extremal points in multidimensional phase stability maps [2–8]. These extremal points are locations along a direction corresponding to a single thermodynamic degree-of-freedom, e.g., the lowest liquidus temperature. This class of problems typically has a single solution and can be formulated straightforwardly as

$$\underset{x\in\Omega}{\text{Minimize}} \quad f(x) \tag{1}$$

where the set $\Omega \subset R^N$ is the feasible region in the decision space (thermodynamic conditions), and f is a function (e.g., a CALPHAD model) from R^N to R that maps thermodynamic conditions to the thermodynamic degree-of-freedom of interest. The optimization problem in Eq. (1) can be solved using conventional techniques. In Ref. [4], mesh-adaptive direct search (MADS) [9] algorithms are used to find extrema in phase stability. More recently, the Arroyave group has used genetic algorithms (GAs) to solve similar problems [6].

Although the location of extremal points, or any isolated point(s) in the phase constitution space, is sometimes of interest to material scientists [10,11], more generalized knowledge is desirable for the materials discovery and design process. Specifically, we are interested in identifying the region or set in the design space that gives rise to a desirable range of thermodynamic conditions, we refer to this problem as the generalized inverse phase stability problem (GIPSP).

For example, a designer may want to know what thermodynamic conditions (a set of designs) result in the presence of σ phase in steels, since any amount of σ phase is undesirable due to its embrittling effects (as little as 3 wt.% reduces impact toughness by more than half [12]) and its drastic deterioration of the stability against corrosion. A compact representation of this set would be useful to material designers as a constraint on the search space that is easy to evaluate. In cases when uncertainty in the appropriate CALPHAD model is high, materials designers may be interested in finding a set of potentially desirable solutions that they can refine and prune through experimentation. In general, materials designers are interested in identifying the set of all the thermodynamic conditions that could produce the set of arbitrary phase constitutions. In the language of engineering design, materials designers would like to apply set-based approach [13] to the materials discovery process.

[1]Corresponding author.
Contributed by the Design Automation Committee of ASME for publication in the JOURNAL OF MECHANICAL DESIGN. Manuscript received March 21, 2016; final manuscript received July 25, 2016; published online October 5, 2016. Assoc. Editor: Carolyn Seepersad.

In the context of set-based design, Shahan and Seepersad suggested the use of Bayesian network classifiers to model the satisfactory region [14]. In Ref. [15], the technique is extended to multilevel design problems (multilevel models are those that are hierarchically nested) and applied to materials design. An extension is developed by Rosen [16] that enables the incorporation of process knowledge. However, the focus of these works is on the design methodology concerning multiscale design rather than an efficient algorithm approach for approximating the satisfactory set, which is the aim of this paper.

We model the GIPSP as a continuous constraint satisfaction problem (CCSP) [17], a type of constraint satisfaction problem (CSP) where all the variable domains are continuous real intervals. The GIPSP is to map specific regions in multidimensional phase constitution spaces to ranges of values of thermodynamic conditions space. The solution set to this problem is ranges of values (enclosures) rather than discrete points. The GIPSP is particularly challenging because the search space is highly nonlinear and discontinuous, the CALPHAD model is a black-box since its details are not available, and the problems of interest may be multidimensional (>10).

Typical algorithmic approaches for CCSPs (Fig. 1), such as those based on interval arithmetic [18], require accessible analytical problem formulations. However, the GIPSP involves a nonanalytical CALPHAD model. Methods such as design of experiments (DOE) [19] or inductive design exploration (IDEM) [20,21] are not tractable for high-dimensional problems since they sample the search space. Another approach is to use adaptive sampling schemes to replace the expensive to evaluate constraint with a surrogate model [22–25]. However, these approaches have difficulty representing discontinuities in the search space and suffer from the curse of dimensionality [26]. To address this limitation, Basudhar and Missoum developed the explicit design space decomposition (EDSD) method. The EDSD method relies on a support vector machine (SVM) classifier to model the design space constraint boundary, rather than approximate the constraint function. The EDSD method has been shown to accurately model the constraint boundary on several test problems within only a fewer iterations.

However as we will show that the SVM technique used in EDSD has difficulty representing the solution to a typical GIPSP. In a GIPSP, it is undesirable to densely sample the unsatisfactory region since it comprises the vast majority of the space. In the case where the samples are imbalanced (many more from one class than another), the SVM will be unreliable in the undersampled region. In cases with imbalanced sampling, the support vector domain description (SVDD) technique has been shown to be more appropriate [27]. In this paper, we present an algorithm that instead relies on the support vector domain description (SVDD) [27] classifier, which leads to improvements in both computation time and solution quality of the GIPSP.

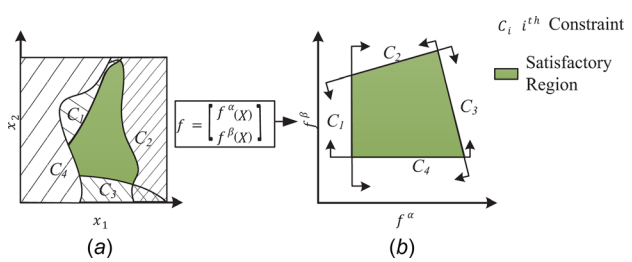

Fig. 1 Illustration of a CCSP. The function $f(\cdot)$ represents the nonlinear CALPHAD model that maps the thermodynamic condition space (*a*) to the phase constitution space (*b*). The query $\mathcal{C} = (C_1, C_2, \ldots, C_4)$ is defined by the user in the phase constitution space, e.g., $C_4 \equiv (f^\beta \geq c)$, where *c* is a constant. Due to the nonlinear mapping between the thermodynamic conditions space and the phase constitution space, the solution in the thermodynamic conditions space is nonconvex.

2 Generalized Inverse Phase Stability Problem as a Continuous Constraint Satisfaction Problem

We define the GIPSP using standard notation in the CSP literature [28] as a triple

$$\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C}) \tag{2}$$

where $\mathcal{X} = (x_1, x_2, \ldots, x_n)$ is a n-tuple of variables defined in the thermodynamic conditions space, and $\mathcal{D} = I_1, \times, I_2, \times \cdots \times, I_n$ is the Cartesian product of the corresponding domains, where I_i is a real interval for $i = 1, 2, \ldots, n$. The set of constraints that must be satisfied is denoted as $\mathcal{C} = (C_1, C_2, \ldots, C_t)$, which is a t-tuple of constraints. Each constraint C_j is a pair R_j, S_j, where R_j is a relation on the variables in $S_j = \text{scope}(C_j)$. The relation R_j defines the consistent value combinations. Specifically, in the context of the GIPSP, R_j is an inequality or equality in terms of the function mapping between thermodynamic conditions and phase constitutions (i.e., constraints defined on the CALPHAD model). The set $S_j \subset \mathcal{X}$ is an unordered k-tuple of distinct variables, where k is the arity of R_j. In other words, S_j is simply the tuple of variables that participate in the constraint R_j.

A solution is a n-tuple $\mathcal{A} = (a_1, a_2, \ldots, a_n)$, where $\mathcal{A} \in \mathcal{D}$ and each C_j is satisfied, in which R_j holds on the projection of $\mathcal{A}$ onto the scope, i.e., S_j. The problem is to find the set of all the solutions to the problem, denoted $\text{sol}(\mathcal{P})$. In the GIPSP, the user defines each constraint C_j in the phase constitution space. For example, one may be interested in the thermodynamic conditions $\mathcal{X} = (x_1, x_2)$ that produce materials consisting of between 40 wt.% and 60 wt.% of a specific phase. This constraint is expressed as $C \equiv 0.4 \leq f(\mathcal{X}) \leq 0.6$, where $f(\cdot)$ maps thermodynamic conditions to the phase composition of interest. The CCSP is expressed as

$$\begin{aligned} \mathcal{X} &= (x_1, x_2) \\ \mathcal{D} &= (x_1^{\text{lb}}, x_1^{\text{ub}}) \times (x_2^{\text{lb}}, x_2^{\text{ub}}) \\ \mathcal{C} &\equiv 0.4 \leq f(\mathcal{X}) \leq 0.6 \end{aligned} \tag{3}$$

where the superscripts lb and ub denote the variable lower and upper bounds, respectively.

Since mapping from the thermodynamic conditions space to the phase constitution space is highly complex, discontinuous, and nonanalytical, satisfying the user-defined constraints C is nontrivial. In the motivating problem, we are interested in the set of all the solutions, $\text{sol}(\mathcal{P})$, to the CCSP where all the variable domains are continuous real intervals and all the numeric relations are equalities and/or inequalities. The constraints are, in general, highly nonlinear.

3 Related Work

3.1 Existing CSP Algorithms. Classical methods for solving CSPs such as backtracking [29], iterative broadening search [30], and limited discrepancy search [31] are intended for problems with discrete domains and are not efficient for solving CCSPs. To apply these to a CCSP, one must discretize the variable space to an enumerable set [32]. Such an approach would be intractable for most GIPSPs because of the high-dimensional (often $n > 10$) [33] of many materials design scenarios. Most techniques developed specifically for solving CCSPs are based on interval arithmetic, branch and bound, or the root inclusion test. In one of the first examples of set-based design, Ward proposed the use of interval arithmetic to search for the feasible set of designs efficiently [18]. Subsequently, this work was extended to more general set-based representations [34]. Hu et al. proposed a method that uses generalized interval to solve for the feasible set [35,36]. The NUMERICA [37] modeling language in particular guarantees correctness, completeness, and certainty. Devanathan and Ramani presented a polytope-based method, where the constraints are transformed

into ternary constraints [38]. Then, the so-called consistency method is used to prune the search space. These techniques require an analytical or closed-form expression to determine whether a subsection of the search space contains feasible solution [28,39,40]. Since the CALPHAD model is a "black-box" in the sense that the details are not accessible, methods that rely on interval arithmetic or constraint decomposition cannot be used for the GIPSP.

3.2 Design of Experiments. Identifying the feasible set (solving a CSP) is a key challenge in many set-based design methods. Design of experiments (DOE) can be used to test the constraint model at a set of finite points in the design domain. A limitation is that the points will not lie on the boundary of the constraints. The inductive design exploration method (IDEM) addresses this limitation [20,21]. The IDEM consists of discrete point evaluations (discrete sampling) of the design process, performing inductive, top-down feasible space exploration based on metamodels. The aim is to obtain a robust solution with respect to model uncertainty. To approximate the constraint boundary, the design space is sampled using a typical DOE. The true constraint boundary will lie between the satisfactory and unsatisfactory discrete points. A root-finding technique is used to find the location of the boundary between these points. The constraint is then represented as a set of boundary points and points inside the feasible space. Although this method generates points on the boundary of the constraint, it does little to reduce the computational burden of the initial DOE, which suffers from the curse of dimensionality.

3.3 Constraint Modeling. Constraint satisfaction has also been addressed in the reliability-based design optimization (RBDO), where the aim is to avoid the performance failure region. Consider a product with random system parameters $\mathcal{X}$ and a performance function $g(x)$, where the product fails if $g(x) < 0$. The reliability of the product is defined as

$$R = P(g(X) \leq 0) = \int_{g(x)>0} \cdots \int f_{\mathcal{X}}(x)dx \tag{4}$$

where $f_{\mathcal{X}}(x)$ is the joint probability density function of all the random parameters. This integral may be approximated using Monte Carlo simulation (MCS), but this approach can be computationally expensive when a large number of samples are required or the constraints are expensive to evaluate. A principle focus in RBDO is the efficient approximation of the solutions to the constraint $g(x)$. An approach common in the literature is to replace the expensive to evaluate constraint $g(x)$, with an inexpensive surrogate model generated from sampled data. The most straightforward methods sample the space globally, which becomes prohibitively expensive at high dimensionality. An approach for reducing the number of samples required to construct the surrogate model is adaptive sampling. Adaptive sampling approaches incrementally improve the surrogate model by sampling points that most effectively improve the model [22–24]. These approaches attempt to efficiently "train" a surrogate model over the entire search space. However, to approximate Eq. (2), it is only necessary to know where the constraint is satisfied. This insight led to the efficient global reliability analysis (EGRA) method. In EGRA, samples are generated by maximizing the expected feasibility function, which provides an indication of how well the true value of the response is expected to satisfy the constraint [25]. Several other criteria can be found in the literature [41–43].

As argued by Basudhar and Missoum, the principle limitations surrogate-based approaches are response discontinuities and the curse of dimensionality [26]. The discontinuities in the search space are problematic for gradient-based techniques. To address this limitation, they proposed a method referred to as explicit design space decomposition (EDSD), which does not approximate the response of the limit state function, instead constructs an explicit constraint boundary around the design variables [44]. Thus, the EDSD method is unaffected by discontinuities and other irregularities in the search space. The EDSD method relies on the use of support vector machines to construct an explicit boundary of the constraint in the design space. Support vector machines (SVMs) are a machine-learning technique for supervised learning associated with *two-class* classification. That is, given a set of training examples, each example is marked as belonging to one of the two classes. The SVM uses the training examples to build a model that assigns new (unobserved) examples to one of the categories. The SVM is initially trained on a sample of points. Then, an adaptive strategy is used to generate points that likely lie on the constraint boundary.

The EDSD method has been shown to approximate the limit state function successfully for a number of test problems [45]. However, because the EDSD method relies on the SVM method, its performance deteriorates when an SVM is not an appropriate representation of the boundary. The SVM treats evaluated samples as a *two-class* data set: feasible and infeasible observations. For a good performance, the SVM technique requires both classes to be well sampled [46,47].

This limitation is significant in the GIPSP, where, intuitively, few process parameter combinations will yield a desirable material. Thus, the solution to a GIPSP is typically small relative to the search space. In this scenario, it is undesirable to evenly sample both satisfactory and unsatisfactory classes, especially for high-dimensional problems where a balances sampling of the entire search space may be computationally expensive. If to reduce computational expense, we undersample the unsatisfactory space, the resulting SVM will be unreliable in the undersampled unsatisfactory space and will tend to have high rates of false positives. In cases with imbalanced sampling, the support vector domain description (SVDD) technique has been shown to be more appropriate [27]. Whereas the SVM scheme is a two-class classifier, the SVDD scheme is a one-class classifier intended to address the case where the training data are mainly from a single category.

4 Support Vector Domain Description

The support vector domain description (SVDD) [27] is used to approximate the solution based on observed data. The SVDD method is a nonprobabilistic machine-learning technique for predicting the boundary of a data set in an Euclidean space. The SVDD method bears close resemblance to support vector machines (SVMs) [48,49]. The principal difference being that SVDD is a one-class classifier while SVMs are two-class classifiers. For a detailed description of the SVDD, see Refs. [27] and [50]. Under the SVDD method, one finds the minimum radius hypersphere that contains a set of training data. Let $\mathbf{x}_i$ denote a vector in the design variable space $\mathcal{X}$ —the input space. Given n data points, $X = \{\mathbf{x}_i | i = 1, 2, \ldots, n\}$, the minimum radius hypersphere containing every data point with centroid $\mathbf{a}$ and radius r

$$\begin{aligned} &\underset{r,\mathbf{a}}{\text{Minimize}} \quad r^2 + c\sum_i \xi_i \\ &\text{subject to} \quad \|\mathbf{x}_i - \mathbf{a}\|^2 \leq r^2 + \xi_i,\ \ \xi_i \geq 0\,\forall i \end{aligned} \tag{5}$$

where ξ_i are the slack variables that allow for the possibility of outliers in the training set. The parameter c defines how to trade-off between Hampshire volume and errors. Any point at a distance equal to or less than r from the hypersphere center is inside of the domain description. However, because a hypersphere is typically a poor representation of the domain, a kernel function is used to nonlinearly remap the training data into a higher-dimensional feature space where a hypersphere is a good model. Through the so-called kernel trick, the data points are mapped to the feature space, without computing the mapping explicitly. The result is an implicit mapping to a feature space of unknown, possible infinite,

dimensionality. There are several valid kernel functions common in the literature [51]. The proposed algorithm uses the Gaussian kernel function

$$K_G(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) = e^{-q\|\mathbf{x}_i - \mathbf{x}_j\|^2} \tag{6}$$

where $\Phi(\cdot)$ is the nonlinear mapping from the data space to the feature space. The q parameter determines how "tightly" or "loosely" the domain description is fit around the training data. The constraint in Eq. (5) then becomes

$$\|\Phi(\mathbf{x}_i) - \mathbf{b}\|^2 \leq r^2 + \xi_i \ \forall i \tag{7}$$

where $\mathbf{b}$ is the centroid of the feature space hypersphere. Rewriting in terms of the kernel function, the Wolfe dual problem can be developed from Eq. (7) as

$$\begin{aligned} \underset{\beta_i}{\text{Maximize}} \quad & \sum_i \beta_i K(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i,j} \beta_i \beta_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} \quad & 0 \leq \beta_i \leq c \ \forall i \\ & \sum_i \beta_i = 1 \end{aligned} \tag{8}$$

For a detailed description of the method for formulating the Wolfe dual problem see Ref. [52]. For each data point, $\mathbf{x}_i$ for $i = 1, 2, \ldots, n$, there are three possible classifications:

(1) It is inside the hypersphere, which is indicated by $\beta_i = 0$.
(2) It is on the boundary of the hypersphere, which is indicated by $0 < \beta_i < c$.
(3) It is an outlier outside of the hypersphere, which is indicated by $\beta_i = c$.

Data on the boundary of the hypersphere are called support vectors (SVs) and are essential to the domain description representation. The outliers are not part of the domain description. Choosing $c \geq 1$ yields no outliers since $\sum_i \beta_1 = 1$ and $0 < \beta_i < c \ \exists i$, and therefore, $\beta_i \neq c \ \forall i$. The squared distance of the feature space image of a point, $\mathbf{z}$, to the centroid of the hypersphere is

$$r^2(\mathbf{z}) = K(\mathbf{z}, \mathbf{z}) - 2\sum_i \beta_i K(\mathbf{x}_i, \mathbf{z}) - \sum_{i,j} \beta_i \beta_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{9}$$

A new test point, $\mathbf{z}$, is inside the domain description if the distance from the feature space image of test point to the hypersphere centroid is less than the radius of the hypersphere. The expression for classification, Eq. (9), is a simple algebraic expression that is fast to evaluate. In fact for the Gaussian kernel function, the first term is equal to 1, and the last term can be precomputed since it is independent of $\mathbf{z}$.

A final consideration is the SVDD method that is able to tighten the description by using negative examples—labeled outliers. The aim in this case is to find the minimum radius hypersphere that includes the positive examples (target-data) while excluding the negative examples (outlier-data). Consider that target-data are enumerated by indices i, j and the outlier-data by l, m. Further, the target-data are labeled $y_i = 1$ and outlier-data are $y_l = -1$. The search problem becomes

$$\begin{aligned} \underset{r,\mathbf{a}}{\text{Minimize}} \quad & r^2 + c_1 \sum_i \xi_i + c_2 \sum_l \xi_l \\ \text{subject to} \quad & \|\mathbf{x}_i - \mathbf{b}\|^2 \leq r^2 + \xi_i \\ & \|\mathbf{x}_l - \mathbf{b}\|^2 \geq r^2 + \xi_l \\ & \xi_i, \xi_l \geq 0 \, \forall i, l \end{aligned} \tag{10}$$

Again, by the method of Lagrange multipliers, we can obtain

$$\begin{aligned} L = & \sum_i \beta'_i K(\mathbf{x}_i, \mathbf{x}_j) - \sum_l \beta'_l K(\mathbf{x}_l, \mathbf{x}_l) - \sum_{i,j} \beta'_i \beta'_j K(\mathbf{x}_i, \mathbf{x}_j) \\ & + 2\sum_{l,j} \beta'_l \beta'_j K(\mathbf{x}_l, \mathbf{x}_j) - \sum_{l,m} \beta'_l \beta'_m K(\mathbf{x}_l, \mathbf{x}_m) \end{aligned} \tag{11}$$

where $\beta'_i = y_i \beta_i$ (the index i again enumerates both target and outlier-data). See Ref. [46] for a detailed exposition of the SVDD method with negative examples.

To prevent weighting variables with large magnitudes more than those with lower ones in this comparison, the training data are centralizing (scale all data to a -1 to 1 range), which improves the SVDD model. An important benefit of the SVDD method is that it can be constructed incrementally and decrementally [53]. This allows for a relatively inexpensive update procedure to be used when new members are added or removed from the SVDD. Figure 2 is an illustration of the SVDD method on a two-dimensional data set in the thermodynamic conditions space.

To underscore the difference in performance between the SVM and SVDD classifiers, we develop two test problems where one class (the satisfactory region) is (a) large and (b) small relative to the domain, see Fig. 3. In each case, we created a training data set with 50 satisfactory and 50 unsatisfactory random examples. In example (a) where the regions are proportional, the samples are relatively balanced, while in (b), they are imbalanced (the unsatisfactory region is undersampled).

We use both examples to train an SVM and SVDD classification model. We used the SVM algorithm in MATLAB's Statistics and Machine Learning Toolbox [54], and SVDD model in DD_Tools [55] developed in the Pattern Recognition Laboratory at Delft University of Technology, Delft, Netherlands. The results are illustrated in Fig. 3. In scenario (a), the SVM model outperforms the SVDD model. This is expected since the SVDD tends to generate conservative model; recall that the SVDD finds the minimum radius hypersphere around the target-data. In contrast, in scenario (b) where the data are not balanced, the SVM classifier results in a significant overestimation. In the latter case, the more conservative SVDD scheme has better performance. This basic insight motivates an adaptive sampling scheme based on SVDD rather than SVM for the GIPSP.

5 Proposed Algorithm

Our aim is to develop an adaptive sampling scheme based on the SVDD technique for approximating the constraint boundary. The basic idea of the proposed algorithm is that the true boundary of the satisfactory region is approximately parallel to our current best guess. In the proposed algorithm, we search along the direction perpendicular to the SVDD boundary (our current best guess) for a point on the *true* boundary of the satisfactory region using a root-finding method. Figure 4 is an illustration of this idea. An initial point is selected along the SVDD boundary along with an initial step size. The initial step size is selected using information from the SVDD. If the end point is outside of the satisfactory region, a root-finding method is used to find a point on the boundary. The proposed method is described in detail in Algorithm 1. We assume that the designer has available small number of designs that satisfy the specified phase state properties, i.e., the constraints. The initial samples may be obtained from prior equilibrium experimental data or found using conventional optimization techniques. In the case studies presented in this paper, we use random sampling to generate n data points $X = \{\mathbf{x}_i | i = 1, 2, \ldots, n\}$ in the design variable space. The randomly generated samples are assigned labels $Y = \{y_i | i = 1, 2, \ldots, n\}$, such that $y = 1$ or $y = -1$ according to whether or not they satisfy the constraints, respectively. The set of indices I_b is initialized to the empty set.

The samples and their corresponding labels are used to generate an approximation of the solution set using the SVDD technique. We use $\mathcal{M}$ to denote this classification model. The model

parameters q and c are selected such that the so-called F_1 measure is minimized, see Ref. [56] for details. Ultimately, our goal is to search along the direction perpendicular to the boundary of $\mathcal{M}$. First, we must select a suitable starting point for this search. An intuitive initial point is some support vector (SV), since the SVs are those samples that lie on the boundary of the domain description. However, in practice, there tends to be few SVs relative to the size of training data. In only a few iterations, all the SVs of the SVDD model may lie on the true boundary of the satisfactory region but our approximation may still be poor. Instead, the algorithm selects the point on the boundary of $\mathcal{M}$ that maximizes the distance to any point on the true boundary. Let the indices corresponding to the samples that lie on the boundary of the true solutions be I_b. The initial sample $\mathbf{x}_a$ is found such that

$$\begin{aligned} \underset{t,\mathbf{x}_a}{\text{Maximize}} \quad & t \\ \text{subject to} \quad & t \leq ||\mathbf{x}_j - \mathbf{x}_a|| \ \forall j \in I_b \\ & \sum_i \beta_i' K_G(\mathbf{x}_i, \mathbf{x}_a) = r \end{aligned} \tag{12}$$

where t is a dummy variable, $X = \{\mathbf{x}_i, i = 1, \ldots, n\}$ are the training data, and $r = \sum_i \beta_i' K_G(\mathbf{x}_i, \mathbf{x}_k)$ for any $k \in SV \subset \{1, \ldots, n\}$, the set of support vectors. To prevent the case where the algorithm attempts to "reuse" a previous initial point, I_b should also include the indices corresponding to previous initial sample points. Because the Gaussian Kernel is an inexpensive function, Eq. (12) is fast to evaluate. It is important to note that the initial point $\mathbf{x}_a$ is not guaranteed to be satisfactory. This can occur when the SVDD modeling parameter q (see Sec. 4) is set too "loose." As a result, it is necessary to evaluate the initial point $\mathbf{x}_a$ against the constraints $\mathcal{C}$ to determine its label, denoted as y_a.

Next, the algorithm takes an initial step in the direction perpendicular to the boundary of $\mathcal{M}$ at $\mathbf{x}_a$, which is the direction that maximizes the distance to the feature space centroid

$$d = \frac{\partial r^2(\mathbf{z})}{\partial \mathbf{z}} = -2 \sum_i \beta_i' \frac{\partial K(\mathbf{x}_i, \mathbf{z})}{\partial \mathbf{z}_i} \tag{13}$$

which is the gradient of Eq. (9). In the case of the Gaussian kernel

$$\frac{\partial K_G(\mathbf{x}_i, \mathbf{z})}{\partial \mathbf{z}} = 2(\mathbf{x}_i - \mathbf{z}) e^{q||\mathbf{z}-\mathbf{x}_i||^2} \tag{14}$$

It is desirable to choose a step size γ large enough to cross the boundary of the satisfactory region. If the label $y_a = 1$, we should step outside of $\mathcal{M}$ to find additional satisfactory points, "growing" the model. On the other hand, $y_a = -1$ indicates that $\mathcal{M}$ is optimistic, and we should step inside of $\mathcal{M}$ to find additional unsatisfactory points, "tightening" the model. Choosing an appropriate initial step size has a significant impact on algorithmic performance. One consideration is that the initial step should not take us too far from the current SVDD, $\mathcal{M}$. To address this, we limit the step size to $\gamma \leq \min_\gamma\{r^2(\mathbf{x}_a - \gamma * d)\}/2$, which limits the step size according to the size and shape of $\mathcal{M}$ along the direction of d. Another consideration is that during search, we should expect disjointed "clusters" in $\mathcal{M}$. In this situation, we should take a step size that is at the feature space midpoint of the clusters. If disjointed clusters exist along the direction d, their feature space midpoint is at $\gamma = \max_\gamma\{r^2(\mathbf{x}_0 + \gamma * d)\}$. We use these concepts to determine the initial step size according to Algorithm 2.

The next step is to search for a boundary point using line search (spec. bisection search), see Algorithm 3 for a description. If the initial step $\mathbf{x}_b$ did not cross the boundary of the true solution, that is, $y_a = y_b$, the bisection search algorithm terminates and returns the training set X and Y containing only the samples $\mathbf{x}_a$ and $\mathbf{x}_b$. Else, the algorithm uses bisection search to reduce the size of the interval between $\mathbf{x}_a$ and $\mathbf{x}_b$ until it is less than some user-defined error tolerance, ε.

Finally, the training set X and Y are updated to contain the samples points that were evaluated against the constraints, i.e., points for which a label y was generated. The set of indices I_b is also updated to contain indices corresponding to the true boundary points found (within tolerance ε). This process is repeated user-defined N times, a termination rather than a convergence criteria. It would be possible to develop a convergence criteria based on the change in $\mathcal{M}$ at each generation; this is left as future work. Without a convergence criteria, analysis of computation complexity is less meaningful but still worth considering. In the case of the GIPSP, evaluating a design against constraints (**ClassLabel** in Algorithms 1–3) is the elementary operation, all others are lower order. Let ε_0 and ε be the maximum interval length and error tolerance for the bisection search, respectively. The time complexity is $\mathcal{O}(N \log_2(\varepsilon_0/\varepsilon))$, where N is the number of iterations to be performed.

Algorithm 1 SVDD-Based Sampling Algorithm

procedure SVDDsample($\mathcal{X}, \mathcal{D}, \mathcal{C}$)
  $X, n \leftarrow$ **RandomSample**($\mathcal{X}, \mathcal{D}$)
  $Y \leftarrow$ **ClassLabel**($X, \mathcal{C}$)
  $I_b \leftarrow \varnothing$
  for $i \leftarrow 1$ to N **do**
    $\mathcal{M} \leftarrow$ **train** SVDD model with X, Y ▷ Eq. (8)
    $\mathbf{x}_a, y_a \leftarrow$ **Select** initial point ▷ Eq. (12)
    $\mathbf{x}_b, y_b \leftarrow$ **TakeInitStep**($\mathcal{M}, \mathcal{C}, \mathbf{x}_a, y_a$)
    $X_i, Y_i, I_i \leftarrow$ **BisectionSearch**($\mathbf{x}_a, y_a, \mathbf{x}_b, y_b, \mathcal{C}$)
    $X, Y \leftarrow (X, X_i), (Y, Y_i)$ ▷ Concatenate
    $I_b \leftarrow (I_b, I_i + n)$
    $n \leftarrow n + I_i$
  return $\mathcal{M}$

Algorithm 2 Take Initial Step

procedure TakeInitStep($\mathcal{M}, \mathcal{C}, \mathbf{x}_a, y_a$)
  $d \leftarrow$ **gradient** of $r(\mathbf{x}_a)$ ▷ Eq. (13)
  $\gamma^{\max} \leftarrow \max_\gamma\{r^2(\mathbf{x}_a + \gamma\ d)\}$ ▷ r^2 from Eq. (9)
  $\gamma^{\min} \leftarrow \min_\gamma\{r^2(\mathbf{x}_a - \gamma\ d)\}$
  $\gamma \leftarrow \min\{\gamma^{\max}, \gamma^{\min}\}$
  if $y_a = -1$ **then** ▷ If $\mathbf{x}_a$ is not satisfactory
    $\gamma \leftarrow -\gamma$
  $\mathbf{x}_b \leftarrow \mathbf{x}_a + \gamma\ d$
  $y_b \leftarrow$ **ClassLabel**($\mathbf{x}_b, \mathcal{C}$)
  return $\mathbf{x}_b$, y_b

Algorithm 3 Bisection Search

procedure BisectionSearch($\mathbf{x}_a, y_a, \mathbf{x}_b, y_b, \mathcal{C}$)
  $X, Y \leftarrow (\mathbf{x}_b, \mathbf{x}_a), (y_b, y_a)$ ▷ Concatenate
  $I \leftarrow 2$ ▷ Counter
  if $y_a \neq y_b$ **then**
    while $||\mathbf{x}_a - \mathbf{x}_b|| \leq \varepsilon$ **do**
      $\mathbf{x}_c \leftarrow \mathbf{x}_b + \frac{1}{2}(\mathbf{x}_a - \mathbf{x}_b)$ ▷ Midpoint
      $y_c \leftarrow$ **ClassLabel**($\mathbf{x}_c, \mathcal{C}$)
      $X, Y \leftarrow (X, \mathbf{x}_c), (Y, y_c)$
      if $y_a = y_c$ **then** ▷ Update interval
        $\mathbf{x}_a, y_a \leftarrow \mathbf{x}_c, y_c$
      else
        $\mathbf{x}_b, y_b \leftarrow \mathbf{x}_c, y_c$
      $I \leftarrow I + 1$ ▷ Update counter
  return X, Y, I

6 Case Study

6.1 Test Problems. We evaluate the performance of the proposed algorithm on Fe–Ti binary alloy system, see phase diagram in Fig. 5. Given the thermodynamic conditions, the phase compositions are computed using ThermoCalc with the TCFE7 database.

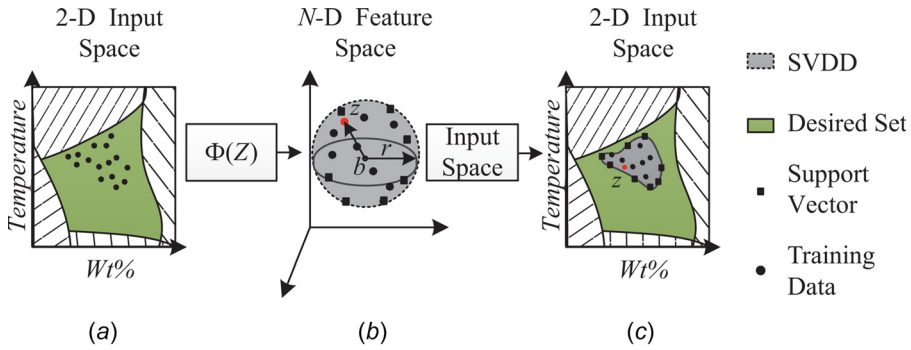

Fig. 2 Illustration of the SVDD method for two-dimensional training data. The training data depicted in (*a*) are implicitly mapped to (*b*), an N-D feature space where a hypersphere is a good representation of the population members. The hypersphere is defined by a centroid *b* and radius *r*. Equation (9) is a test to determine whether a new point *z* is inside the domain description. This test defines the boundary illustrated in (*c*).

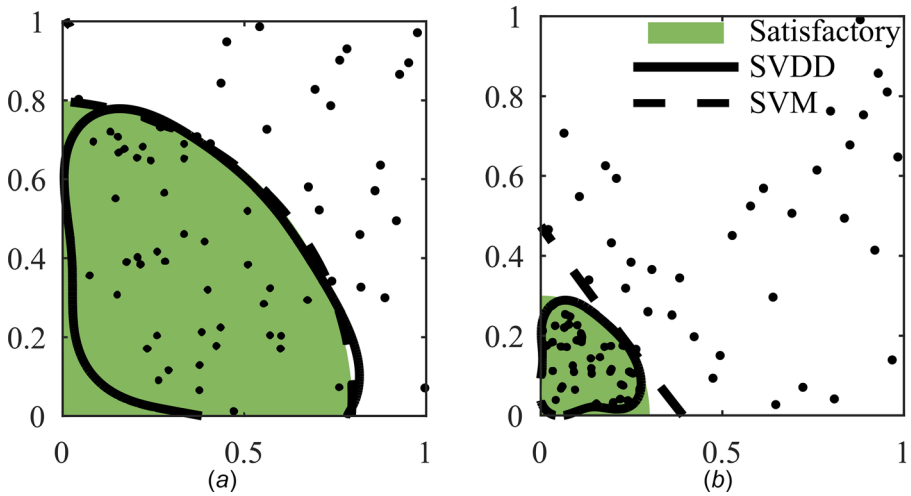

Fig. 3 Comparison of SVM and SVDD on a simple classification problem, where the sampling is balanced (*a*) and imbalanced (*b*). The shading indicates the true classification, and the points are the training data.

Recall that the GIPSP is the triple $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$. The thermodynamic conditions (design variables) are $\mathcal{X} = (x_1, x_2, x_3)$, where x_1 is the mass percent of Ti, x_2 is the mass percent of Fe, and x_3 is the temperature (Kelvin). The search domain $\mathcal{D}$ is defined as

$$\begin{aligned} &x_1 + x_2 = 100 \\ &0 \le x_i \le 100 \ \text{ for } \ i = 1, 2 \\ &600 \le x_3 \le 2000 \end{aligned} \tag{15}$$

The CALPHAD model, denoted $f = (f_1(\mathcal{X}), f_2(\mathcal{X}), \ldots, f_5(\mathcal{X}))$, maps the thermodynamic conditions to the volume fraction of BCC, FCC, HCP, Laves, and liquid phase, respectively. We consider two different scenarios (test cases). In the first test case, the materials designer wishes to find all the thermodynamic conditions that result in volume fraction of Laves and liquid phase between 0.25 and 0.75. For this test case, the constraints $\mathcal{C}$ are

$$0.25 \le f_i(\mathcal{X}) \le 0.75 \ \text{ for } \ j = 4, 5 \tag{16}$$

In the second test case, the materials designer wishes to find all thermodynamic conditions that result in volume fraction FCC greater than 0.05. For this test case, the constrain C is

$$f_2(\mathcal{X}) \ge 0.05 \tag{17}$$

The solutions to these search problems are illustrated in Fig. 5, the shaded regions. The test cases were selected to investigate the performance of the algorithm when the solution is nonconvex (test case 1), and small relative to the search space (test case 2). Notice that for test case 1, interactions at the eutectic cause irregularities in the solution set. Although these irregularities will not agree

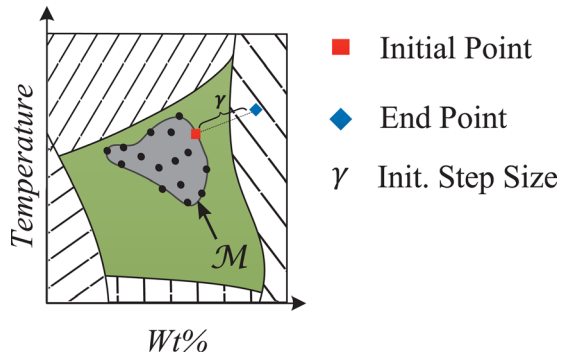

Fig. 4 Illustrative example depicting a possible step size and resulting endpoint from a given initial point along the boundary of the SVDD $\mathcal{M}$. An aim in selecting a step size is to cross the boundary of the satisfactory region.

with experimental data, they complicate the search space and can give better insight into the performance.

6.2 Results and Analysis. The proposed algorithm is motivated by the GIPSP, which is typically multidimensional and features response discontinuities that are problematic for gradient-based search techniques. To evaluate the performance of the proposed algorithm on this class of problem, we compare to EDSD, which is intended for constraint satisfaction problems with similar characteristics. The principal difference is that the proposed algorithm is intended to address the case where the satisfactory region is undersampled (as we expect is the case with most GIPSPs).

In both algorithms, performance is dependent on the initial training data. To account for this, each case was tested for 30 trials with random initial training data. For each trial (in either test case), we randomly sampled the domain $\mathcal{D}$ to find ten feasible and ten infeasible sites. The same data are used to initialize each algorithm. These initializing functions are not counted toward the overall function count of either algorithm.

Since both EDSD and the GIPSP algorithm use binary classifiers, we evaluate solution quality using the *precision* and *recall* metrics commonly used in pattern recognition [57]. We generate a set X of 10^6 random samples in the design space domain $\mathcal{D}$ defined by Eq. (15). We then find $X' \subset X$, the subset that satisfies the user-specified conditions, $\mathcal{C}$, in Eq. (16) for test case 1 and Eq. (17) for test case 2. Next, we find $X'' \subset X$ the subset that is classified as belonging to the satisfactory set, according to the classifier (SVM for EDSD and SVDD for the GIPSP algorithm). We compute the *true positives*, *true negatives*, *false positives*, and *false negatives* as

	Positive	Negative
True	$N_{\text{tp}} = \lvert X' \cup X'' \rvert$	$N_{\text{tn}} = \lvert X \backslash X' \cup X \backslash X'' \rvert$
False	$N_{\text{fp}} = \lvert X' \cup X \backslash X'' \rvert$	$N_{\text{fn}} = \lvert X \backslash X' \cup X'' \rvert$

where $\lvert \cdot \rvert$ denotes cardinality. The terms *positive* and *negative* refer to the classifier's prediction and *true* and *false* refer to how that prediction corresponds to the *actual* classification. Taking these terms into account, we can compute the precision and recall measures as

$$\text{Precision} = \frac{N_{\text{tp}}}{N_{\text{tp}} + N_{\text{fp}}} \qquad \text{Recall} = \frac{N_{\text{tp}}}{N_{\text{tp}} + N_{\text{fn}}} \tag{18}$$

If we consider the positive classifier predictions to be the "solution set," precision can be interpreted as the probability that a randomly selected alternative in the solution set will be truly satisfactory. Recall is the probability that a randomly selected true solution will be represented in the solution set. Thus, higher values of precision and recall are preferred. We also consider the misclassification rate of the classifier as a performance metric

$$\text{Misclassification rate} = \frac{N_{\text{fp}} + N_{\text{fn}}}{N_{\text{fp}} + N_{\text{fn}} + N_{\text{tp}} + N_{\text{tn}}} \tag{19}$$

Lower values of misclassification rate are preferred, however, one must be cautions when interpreting this measure. For example, in a case where the satisfactory region is very small, classifying the entire region as "unsatisfactory" will have a low misclassification rate.

We calculate each performance metric at several intervals for each algorithm. We report the mean values and 95% confidence interval of 30 trials for both test cases in Figs. 6 and 7, respectively. In test case 1, the GIPSP algorithm produces a higher precision approximation for any given number of function evaluations. This is not unexpected since the SVDD is more conservative than the SVM technique. Both algorithms converge to a similar measure of recall and a low level of misclassification error. In test case 2, the GIPSP algorithm generates high-precision solutions and converges to a solution with high recall and low misclassification rate. However, the precision of the EDSD solution does not improve significantly after 100 iterations, and the recall measure becomes worse. Further, the solution quality is highly variable across each iteration, resulting in large confidence intervals.

For illustrative purposes, we include Figs. 8 and 9, which depict the progression of each algorithm at the (*a*) 100, (*b*) 250, and (c) 400, function evaluations for each test case. The shaded region represents the true solution to each CCSP found through an exhaustive search. An attempt was made to illustrate trials that are representative of the mean values in 6 and 7. However, we should not that in both test cases (but especially for test case 2), the performance of the EDSD algorithm varied significantly. Therefore, no illustration of a single result can be truly illustrative of the typical results. Taking these limitations into account, the illustrations still provide some valuable insight into the performance of each algorithm.

As can be seen in Fig. 8, the GIPSP algorithm maintains high precision during its progression, including few false positives. The EDSD, on the other hand, produces more optimistic estimations of the satisfactory region, initially. The higher precision of the GIPSP solution is reflected in the "tighter" classifier boundary. The precision at (*c*) 400 function evaluations for the EDSD algorithm is considerably higher than the med.

Note that Fig. 9 is focused on the solution space (which is quite small), the search space for this test problem is defined by Eq. (15) and depicted in Fig. 5. Test case 2 is intended to highlight the limitations of EDSD method on problems where the satisfactory region is small relative to the search space. In such cases, the SVM technique used in EDSD tends to overestimate the satisfactory region. In Fig. 9, the overestimation occurs near the true solution (shaded portions), however, this is not always the case. Figure 10 is an illustration of the results from another trial of the EDSD algorithm in test case 2.

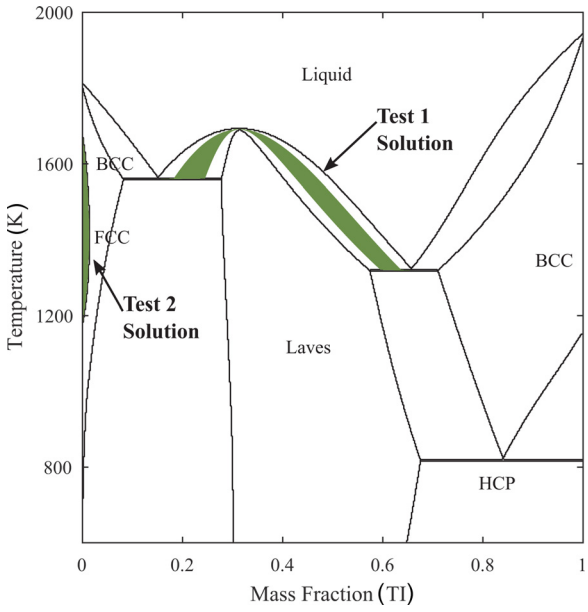

Fig. 5 Phase diagram for Fe–Ti binary alloy system. The shaded regions correspond to the solution sets for test cases 1 and 2.

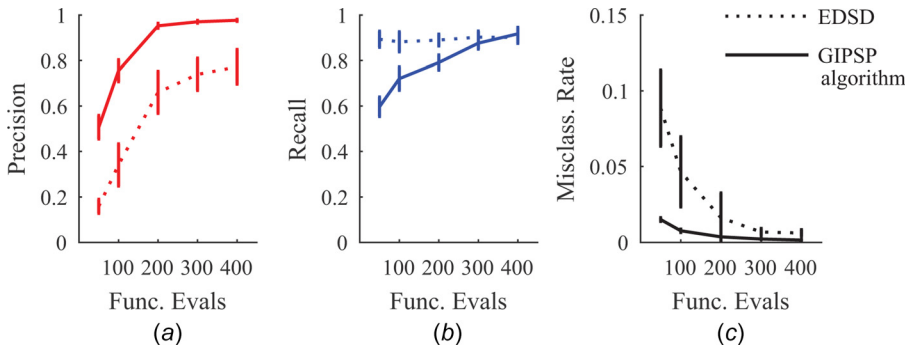

Fig. 6 Algorithmic performance on test case 1 as measured by (*a*) precision, (*b*) recall, and (*c*) misclassification. Error bars indicate the mean values and 95% confidence interval of 30 trials.

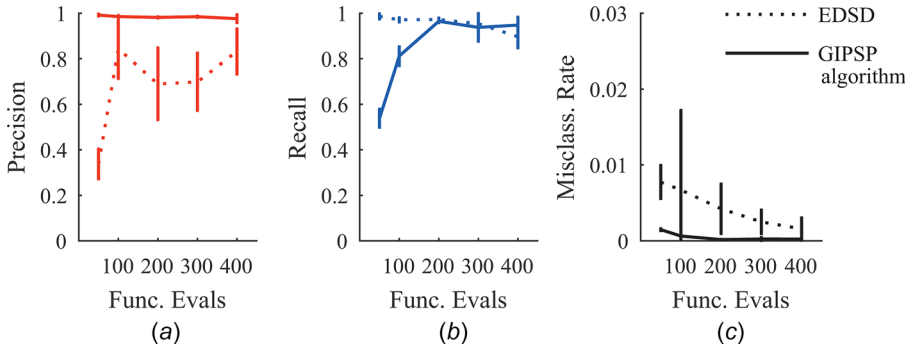

Fig. 7 Algorithmic performance on test case 2 as measured by (*a*) precision, (*b*) recall, and (*c*) misclassification. Error bars indicate the mean values and 95% confidence interval of 30 trials.

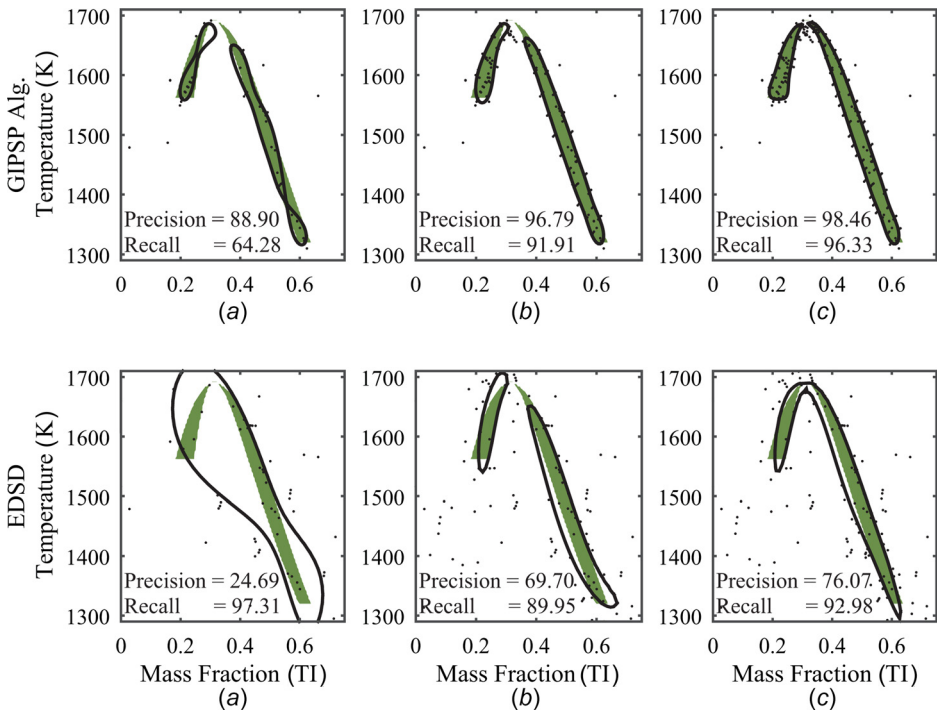

Fig. 8 Illustration of the progression of each algorithm at the function evaluations for test case 1. The top row corresponds to the progression of the GIPSP algorithm, and the bottom row corresponds to the EDSD algorithm. The satisfactory region is shaded, the samples are the points, and the classification boundaries are the outlines.

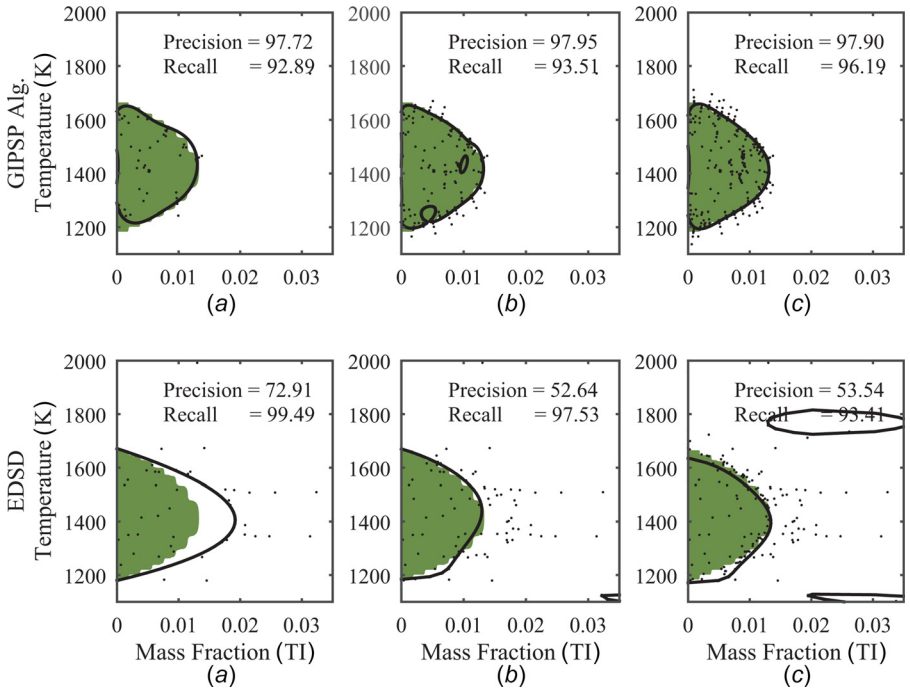

Fig. 9 Illustration of the progression of each algorithm at the (*a*) 100, (*b*) 250, and (*c*) 400, function evaluations for test case 2. The top row corresponds to the progression of the GIPSP algorithm, and the bottom row corresponds to the EDSD algorithm. The satisfactory region is shaded, the samples are the points, and the classification boundaries are the outlines.

7 Discussion and Summary

In this paper, we have presented a novel algorithm for approximating all the solutions to a CCSP with nonisolated solution where the satisfactory region is small relative to the search space. The algorithm uses the SVDD technique combined with a sampling strategy to gradually develop the solution. The motivation for the algorithm is the general inverse phase stability problem of mapping user-specified regions in multidimensional phase constitution space to ranges in values of thermodynamic conditions, which we term the GIPSP. In the GIPSP, one class (the satisfactory region) is small relative to the other (unsatisfactory region). For scalability, it is desirable to undersample the unsatisfactory region, since it comprises the vast majority of the space. This motivates the use of the SVDD method in the algorithm since it is able to more accurately (in terms of precision and recall) model scenarios with imbalanced training data.

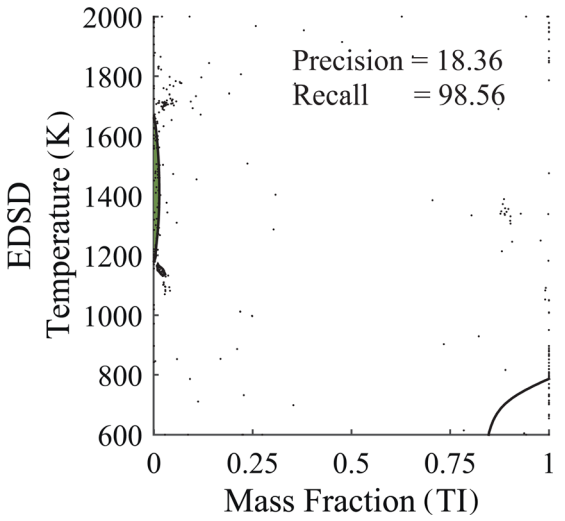

Fig. 10 Illustration of a single trial of the EDSD algorithm at the 400 function evaluations for test case 2. The satisfactory region is shaded, the samples are the points, and the classification boundaries are the outlines. The results in this trial illustrate a possible overestimation of the EDSD algorithm in regions of the search space far from the true solution.

We investigated the performance of the algorithm on Fe–Ti binary alloy system using ThermoCalc with the TCFE7 database. Using this system, we formulated two test cases. In the first test case, the solution set is nonconvex; in the second, the solution set is small relative to the search space. We compare the performance of the GIPSP algorithm to the EDSD algorithm, which uses a related classification scheme, namely, SVM. The performance of each algorithm on the test problems was measured as the precision, recall, and misclassification rate. In both test problems, the GIPSP algorithm is able to converge to a solution with high precision and recall. The EDSD algorithm, however, had significant difficulty in approximating the solution to test case 2. This is likely the result of the limitations of the SVM technique used in EDSD. The SVM technique is known to underperform in cases with imbalanced training data sets. In test case 2, the satisfactory region is small relative to the search space, resulting in an imbalanced training data set.

Future work should also investigate the performance of the algorithm on problems of higher dimensionality that are more representative of real-world materials design problems.

Acknowledgment

This work was supported by the National Science Foundation and the Air Force under Grant No. EFRI-1240483. The authors would like to thank Paul Mason from ThermoCalc for providing critical thermodynamic data, which made this research possible.

Nomenclature

a = centroid of hypersphere
b = centroid of feature space hypersphere

c = SVDD parameter
C_j = constraint
d = direction perpendicular to SVDD boundary
f = thermodynamic conditions → phase constitution
$f_{\mathcal{X}}$ = random parameter joint pdf
g = performance function
I_b = set of indices corresponding to boundary points
I_i = real interval
K = kernel function
n = number of data points
N = dimensionality of thermodynamic conditions space
N_{fn} = number of false negatives
N_{fp} = number of false positives
N_{tn} = number of true negatives
N_{tp} = number of true positives
q = Gaussian kernel parameter
r = hypersphere radius
R = reliability
R_j = relation on the variables involved in constraint C_j
S_j = scope of the constraint C_j
SV = set of support vectors
t = dummy variable
X = training data set
x_i = design variable
$\mathbf{x}_i$ = a vector in the design variable space $\mathcal{X}$
Y = set of training data labels
y_i = training data label
$\mathbf{z}$ = test point
β_i = Lagrangian multiplier
γ = step size
ε = error tolerance
ε_0 = maximum interval length
ξ_i = slack variable
Φ = data space → feature space
$\mathcal{A}$ = n-tuple solution to $\mathcal{P}$
$\mathcal{C}$ = t-tuple of constraints
$\mathcal{D}$ = search space
$\mathcal{M}$ = classification model
$\mathcal{P}$ = constraint satisfaction problem
$\mathcal{X}$ = n-tuple of variables

References

[1] Saunders, N., and Miodownik, A. P., 1998, *CALPHAD (Calculation of Phase Diagrams): A Comprehensive Guide*, Vol. 1, Elsevier, Amsterdam, The Netherlands.
[2] Bale, C., Chartrand, P., Degterov, S., Eriksson, G., Hack, K., Ben Mahfoud, R., Melançon, J., Pelton, A., and Petersen, S., 2002, "Factsage Thermochemical Software and Databases," Calphad, **26**(2), pp. 189–228.
[3] Gheribi, A. E., Robelin, C., Digabel, S. L., Audet, C., and Pelton, A. D., 2011, "Calculating All Local Minima on Liquidus Surfaces Using the Factsage Software and Databases and the Mesh Adaptive Direct Search Algorithm," J. Chem. Thermodyn., **43**(9), pp. 1323–1330.
[4] Gheribi, A. E., Audet, C., Le Digabel, S., Bélisle, E., Bale, C., and Pelton, A., 2012, "Calculating Optimal Conditions for Alloy and Process Design Using Thermodynamic and Property Databases, the Factsage Software and the Mesh Adaptive Direct Search Algorithm," Calphad, **36**, pp. 135–143.
[5] Zhu, R., Li, S., Karaman, I., Arroyave, R., Niendorf, T., and Maier, H. J., 2012, "Multi-Phase Microstructure Design of a Low-Alloy Trip-Assisted Steel Through a Combined Computational and Experimental Methodology," Acta Mater., **60**(6–7), pp. 3022–3033.
[6] Zhu, R., Li, S., Karaman, I., Arroyave, R., Niendorf, T., and Maier, H. J., 2012, "Multi-Phase Microstructure Design of a Low-Alloy Trip-Assisted Steel Through a Combined Computational and Experimental Methodology," Acta Mater. **60**(6–7), pp. 3022–3033.
[7] Li, S., Zhu, R., Karaman, I., and Arroyave, R., 2012, "Thermodynamic Analysis of Two-Stage Heat Treatment in Trip Steels," Acta Mater., **60**(17), pp. 6120–6139.
[8] Li, S., Zhu, R., Karaman, I., and Arroyave, R., 2012, "Thermodynamic Analysis of Two-Stage Heat Treatment in TRIP Steels," Acta Mater. **60**(17), pp. 6120–6130.
[9] Audet, C., and Dennis, J., Jr., 2006, "Mesh Adaptive Direct Search Algorithms for Constrained Optimization," SIAM J. Optim., **17**(1), pp. 188–217.
[10] Gomez-Acebo, T., Sarasola, M., and Castro, F., 2003, "Systematic Search of Low Melting Point Alloys in the Fe-Cr-Mn-Mo-C System," Calphad, **27**(3), pp. 325–334.
[11] Du, H., and Morral, J., 1997, "Prediction of the Lowest Melting Point Eutectic in the Fe-Cr-Mo-V-C System," J. Alloys Compd., **247**(1), pp. 122–127.
[12] Hou, J. S., Guo, J. T., Zhou, L. Z., and Ye, H. Q., 2006, "Sigma Phase Formation and Its Effect on Mechanical Properties in the Corrosion-Resistant Superalloy K44," Z. Metallkd./Mater. Res. Adv. Tech., **97**(2), pp. 174–181.
[13] Ward, A. C., Liker, J. K., Cristiano, J. J., and Sobek, D. K., II, 1995, "The Second Toyota Paradox: How Delaying Decisions Can Make Better Cars Faster," MIT Sloan Manage. Rev., **Spring**, pp. 43–61.
[14] Shahan, D. W., and Seepersad, C. C., 2012, "Bayesian Network Classifiers for Set-Based Collaborative Design," ASME J. Mech. Des., **134**(7), p. 071001.
[15] Matthews, J., Klatt, T., Morris, C., Seepersad, C. C., Haberman, M., and Shahan, D., 2016, "Hierarchical Design of Negative Stiffness Metamaterials Using a Bayesian Network Classifier," ASME J. Mech. Des., **138**(4), p. 041404.
[16] Rosen, D. W., 2015, "A Set-Based Design Method for Material-Geometry Structures by Design Space Mapping," ASME Paper No. DETC2015-46760.
[17] Cruz, J., 2005, "Constraint Reasoning for Differential Models," 2005 Conference on Constraint Reasoning for Differential Models, IOS Press, pp. 1–216.
[18] Ward, A. C., 1989, "A Theory of Quantitative Inference Applied to a Mechanical Design Compiler," Ph.D., MIT, Cambridge, MA.
[19] Montgomery, D. C., 2008, *Design and Analysis of Experiments*, Wiley, New York.
[20] Choi, H.-J., Mcdowell, D. L., Allen, J. K., and Mistree, F., 2008, "An Inductive Design Exploration Method for Hierarchical Systems Design Under Uncertainty," Eng. Optim., **40**(4), pp. 287–307.
[21] Choi, H., McDowell, D. L., Allen, J. K., Rosen, D., and Mistree, F., 2008, "An Inductive Design Exploration Method for Robust Multiscale Materials Design," ASME J. Mech. Des., **130**(3), p. 031402.
[22] Wang, C., Duan, Q., Gong, W., Ye, A., Di, Z., and Miao, C., 2014, "An Evaluation of Adaptive Surrogate Modeling Based Optimization With Two Benchmark Problems," Environ. Modell. Software, **60**, pp. 167–179.
[23] Huang, D., Allen, T., Notz, W., and Miller, R., 2006, "Sequential Kriging Optimization Using Multiple-Fidelity Evaluations," Struct. Multidiscip. Optim., **32**(5), pp. 369–382.
[24] Wang, G. G., Wang, L., and Shan, S., 2005, "Reliability Assessment Using Discriminative Sampling and Metamodeling," SAE Technical Paper No. 2005-01-0349.
[25] Bichon, B. J., Eldred, M. S., Swiler, L. P., Mahadevan, S., and McFarland, J. M., 2008, "Efficient Global Reliability Analysis for Nonlinear Implicit Performance Functions," AIAA J., **46**(10), pp. 2459–2468.
[26] Basudhar, A., and Missoum, S., 2008, "Adaptive Explicit Decision Functions for Probabilistic Design and Optimization Using Support Vector Machines," Comput. Struct., **86**(19), pp. 1904–1917.
[27] Tax, D. M., and Duin, R. P., 1999, "Support Vector Domain Description," Pattern Recognit. Lett., **20**(11–13), pp. 1191–1199.
[28] Tsang, E., 1993, *Foundations of Constraint Satisfaction*, Vol. 289, Academic Press, London.
[29] Golomb, S. W., and Baumert, L. D., 1965, "Backtrack Programming," JACM, **12**(4), pp. 516–524.
[30] Ginsberg, M. L., and Harvey, W. D., 1992, "Iterative Broadening," Artif. Intell., **55**(2), pp. 367–383.
[31] Harvey, W. D., and Ginsberg, M. L., 1995, "Limited Discrepancy Search," IJCAI, **1**, pp. 607–615.
[32] Sam-Haroud, D., and Faltings, B., 1996, "Consistency Techniques for Continuous Constraints," Constraints, **1**(1), pp. 85–118.
[33] Cruz, J., and Barahona, P., 2003, "Constraint Satisfaction Differential Problems," *Principles and Practice of Constraint Programming–CP 2003*, Springer, Berlin, pp. 259–273.
[34] Finch, W. W., and Ward, A. C., 1997, "A Set-Based System for Eliminating Infeasible Designs in Engineering Problems Dominated by Uncertainty," Proceedings of the ASME Design Engineering Technical Conferences, Sacramento, CA, Paper No. DETC97/DTM-3886.
[35] Hu, J., Aminzadeh, M., and Wang, Y., 2014, "Searching Feasible Design Space by Solving Quantified Constraint Satisfaction Problems," ASME J. Mech. Des., **136**(3), p. 031002.
[36] Hu, J., Wang, Y., Cheng, A., and Zhong, Z., 2015, "Sensitivity Analysis in Quantified Interval Constraint Satisfaction Problems," ASME J. Mech. Des., **137**(4), p. 041701.
[37] Van Hentenryck, P., Michel, L., and Deville, Y., 1997, *Numerica: A Modeling Language for Global Optimization*, MIT Press, Cambridge, MA.
[38] Devanathan, S., and Ramani, K., 2010, "Creating Polytope Representations of Design Spaces for Visual Exploration Using Consistency Techniques," ASME J. Mech. Des., **132**(8), p. 081011.
[39] Neumaier, A., 2004, "Complete Search in Continuous Global Optimization and Constraint Satisfaction," Acta Numer., **13**(1), pp. 271–369.
[40] Kearfott, R. B., 1987, "Abstract Generalized Bisection and a Cost Bound," Math. Comput., **49**(179), pp. 187–202.
[41] Sasena, M. J., 2002, "Flexibility and Efficiency Enhancements for Constrained Global Design Optimization With Kriging Approximations," Ph.D. thesis, University of Michigan, Ann Arbor, MI.
[42] Schonlau, M., 1998, *Computer Experiments and Global Optimization*, University of Waterloo, Waterloo, ON, Canada.
[43] Au, S., and Beck, J. L., 1999, "A New Adaptive Importance Sampling Scheme for Reliability Calculations," Struct. Saf., **21**(2), pp. 135–158.

[44] Basudhar, A., and Missoum, S., 2010, "An Improved Adaptive Sampling Scheme for the Construction of Explicit Boundaries," Struct. Multidiscip. Optim., **42**(4), pp. 517–529.
[45] Basudhar, A., 2011, *Computational Optimal Design and Uncertainty Quantification of Complex Systems Using Explicit Decision Boundaries*, The University of Arizona, Tucson, AZ.
[46] Tax, D. M., and Duin, R. P., 2004, "Support Vector Data Description," Mach. Learn., **54**(1), pp. 45–66.
[47] Le, T., Tran, D., Ma, W., and Sharma, D., 2012, "A Unified Model for Support Vector Machine and Support Vector Data Description," 2012 International Joint Conference on Neural Networks (IJCNN), IEEE, pp. 1–8.
[48] Vapnik, V., 1995, *The Nature of Statistical Learning Theory*, Springer, New York.
[49] Scholkopf, B., and Smola, J. A., 2002, *Learning With Kernels*, MIT Press, Cambridge, MA.
[50] Malak, J. R. J., and Paredis, C. J. J., 2010, "Using Support Vector Machines to Formalize the Valid Input Domain of Predictive Models in Systems Design Problems," ASME J. Mech. Des., **132**(10), p. 101001.
[51] Scholkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., and Platt, J., 1999, "Support Vector Method for Novelty Detection," *Advances in Neural Information Processing Systems*, MIT Press, pp. 582–588.
[52] Wolfe, P., 1961, "A Duality Theorem for Nonlinear Programming," Q. Appl. Math., **19**(3), pp. 239–244.
[53] Cauwenberghs, G., and Poggio, T., 2001, "Incremental and Decremental Support Vector Machine Learning," Neural Information Processing Systems, **13**, pp. 409–415.
[54] MathWorks, 1998, "Mathworks User's Guide," Vol. 5, MathWorks Inc., Natick, MA, p. 333.
[55] Tax, D., 2005, "Ddtools: The Data Description Toolbox for MATLAB," Delft University of Technology, Delft, Netherlands.
[56] Tax, D. M., 2015, "Ddtools: The Data Description Toolbox for MATLAB, Version 2.1.2," Delft University of Technology, Delft, Netherlands.
[57] Powers, D. M., 2011, "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation," J. Mach. Learn. Technol., **2**(1), pp. 37–63.