

**DETC2008-49147**

## **A LEARNING AND INFERENCE MECHANISM FOR DESIGN OPTIMIZATION PROBLEM (RE)- FORMULATION USING SINGULAR VALUE DECOMPOSITION**

**Somwrita Sarkar, Andy Dong**  
University of Sydney, Australia

**John S. Gero**  
George Mason University, USA

### **ABSTRACT**

This paper presents a knowledge-lean learning and inference mechanism based on Singular Value Decomposition (SVD) for design optimization problem (re)-formulation at the problem modeling stage. The distinguishing feature of the mechanism is that it requires very few training cases to extract and generalize knowledge for large classes of problems sharing similar characteristics. The genesis of the mechanism is based on viewing problem (re)-formulation as a statistical pattern extraction problem. SVD is applied as a dimensionality reduction tool to extract semantic patterns from a syntactic formulation of the design problem. We explain and evaluate the mechanism on a model-based decomposition problem, a hydraulic cylinder design problem, and a medium-large scale Aircraft Concept Sizing problem. The results show that the method generalizes quickly and can be used to impute relations between variables, parameters, objective functions, and constraints when training data is provided in symbolic analytical form, and is likely to be extensible to forms when the representation is not in analytical functional form.

Keywords: optimization, problem (re)-formulation, machine learning, singular value decomposition, pattern extraction

### **INTRODUCTION**

Design problem formulation and re-formulation significantly affect the final results of any design optimization process. Currently, design optimization tools do not support the designer to re-formulate the problem due to the not unsubstantial knowledge engineering that would have to be embedded into such a tool. Design problem re-formulation is difficult to automate because human subjective decision making and a large amount of domain and mathematical expertise form the basis of problem (re)-formulation. Designers learn from years of experience (Moss et al., 2004a) and trial and error to reach a mathematically formulated design optimization model that satisfactorily captures all design requirements, and guarantees both the existence of a

solution and locating the optimal solution within a bounded computation time through the application of a solution algorithm, numeric or symbolic.

In problem (re)-formulation, the same design work may be modeled in multiple ways, depending upon personal, design-based or mathematical choices exercised by the designer. The problem definition evolves as the designer's understanding of the problem grows. Many reformulations are typically required to reach a satisfactory model, making problem formulation and solution an iterative, cyclical process.

For example, consider the hydraulic cylinder design problem (Papalambros and Wilde, 2000) that has been modeled and solved in different ways –

- Papalambros and Wilde (2000) present it as a single-objective problem with 5 design variables, 6 constraints and solved it using monotonicity analysis. A preliminary approach used numerical parameter values, followed by a more general parametric solution method that kept the parameters symbolic.
- Michelena and Agogino (1988) cast the problem as a multi-objective problem with 5 design variables, 6 constraints (slightly different in form to Papalambros and Wilde) and solved it using monotonicity analysis according to the rules of non-linear optimization.
- Cagan and Williams (1997) solved the problem using activity analysis.
- It could be modeled and solved numerically using a deterministic, non-gradient based approach such as Powell's method, where multiple attempts at starting the optimization from a good initial point could lead to an optimal solution.
- If the designer thought of the problem as a stochastic global optimization problem, it could be solved using a genetic algorithm. The constrained problem would be re-formulated as an unconstrained optimization problem,

with the constraints incorporated into the objective function as penalty and barrier functions.

As is evident, there are multiple ways in which the same problem may be conceived and solved. In general, the choices exercised by the designer manifest in many ways (Papalambros and Wilde, 2000) –

- Selecting design variables and parameters
- Selecting natural design constraints (modeling the physics of the system), pragmatic ones guided by engineering guidelines and “rules of thumb” (handbooks, manuals etc.), or externally imposed ones (manufacturing limitations, material availability, interdisciplinary constraints, available software, etc.)
- Selecting algorithms that are applied to solve the model and parameters for these algorithms
- Deciding on the forms of coupling the design model and the solution method (algorithm)

Any learning and inference mechanism forming the basis of a computational system that assists a designer in design optimization problem modeling and (re)-formulation must be sensitive to these characteristics and difficulties of the design optimization process. As the complexity of processes and products in engineering increases, design models in optimization become cognitively intractable in terms of the number of variables, parameters, and constraints. The dimensionality of the problem often goes beyond human short-term memory capacities. There is a need for computational systems to provide support for tasks that were previously considered purely a human domain. There are various cognitive studies (Moss et al., 2004a) that show that designers learn from past experiences and apply this knowledge onto future design experiences. Computational design support systems must develop similar robust learning capacities.

In this paper, we present a general learning and inference mechanism that inductively extracts and learns semantic design elements and relationships from the syntax of analytically formulated design models. The distinguishing feature of the mechanism is that it requires a small number of training cases to extract and generalize knowledge for large classes of problems sharing similar characteristics. The main hypothesis is that the mathematical-symbolic representation of a design work implicitly captures not only the semantic concepts and choices that the designer uses, but also the structural-behavioral-functional characteristics of the design object that are being modeled.

The computational implementation of the methodology is based on Singular Value Decomposition (SVD) as a generalized dimensionality reduction and pattern extraction mechanism. The underlying aim of the research is a cognitive-computational exploration into the theoretical question of learning from a statistical pattern-based perspective, and the methodological question of being able to model this cognitive process computationally such that a learning algorithm, to some extent, could produce results similar to human cognitive processes in design. We make no claim that the methodology, in any way, represents actual cognitive processes, that is, how the brain works.

## THEORETICAL INSIGHTS

Much research has focused on combining artificial intelligence and machine learning techniques to improve upon and automate aspects of design optimization. Cagan et al. (1997) present a review of artificial intelligence techniques for optimization in engineering design. Schwabacher et al. (1998) explore inductive machine learning techniques such as decision tree induction for automating various tasks in design optimization. Other notable research exploring similar issues include work by Ellman et al. (1998) and Gelsey et al. (1998). Campbell et al. (1999, 2003) present a design synthesis tool called A-Design, in which agents in a multi-agent system, based on genetic algorithms, asynchronous teams and functional reasoning, evolve conceptual design objects. Moss et al. (2004a) explore learning in such a system, where useful “chunks” of design knowledge are learnt and used in future design tasks.

Most of these methods either require a high level of knowledge engineering, or they require a large training database of solved examples of various problems for the techniques to exhibit useful learning and inference characteristics. In contrast to other methods based on high level knowledge-engineering or supervised learning, the theoretical basis of this methodology views the design (re)-formulation problem from a statistical pattern extraction perspective using unsupervised learning. The approach is inspired by ideas from the statistical natural language processing (NLP) and digital image processing domains.

In statistical NLP, Latent Semantic Analysis, based on SVD (Landauer and Dumais 1998), is used to reveal semantic patterns in textual data. The underlying idea in LSA is that many domains of knowledge contain a large number of weak correlations between semantic concepts, and that these distributed, “context” based, latent correlations are captured by the syntax of representation. The main claim is that SVD is able to reveal these patterns by employing the correct dimensionality in data representation. In digital image processing (Kalman 1996, Strang 2003), SVD is used as a mathematical tool to identify pattern redundancy in image data for compression of images.

Engineering design as a discipline is vastly different from both these domains. In design optimization, mathematics with its precise syntax is the main representational mechanism. We conjecture that an application of the SVD technique, combined with similarity measurements from statistical NLP to mathematical design optimization models, could nonetheless reveal an interesting parallel observation – that SVD is able to reveal hidden design patterns inherent in the syntactic, analytical representation of a design work.

Mathematically, it is an intriguing and interesting characteristic that SVD, as an algorithm, has the capacity to be used as a general pattern extraction mechanism in such vastly different domains. We were intuitively inspired to explore this issue, leading to a general observation – the concepts of a domain are interpretively contained in the syntax of representation, i.e., words, sentences. The empirical patterns of occurrence capture inherent “meaning” in natural language; redundancies of pixel occurrences in an image capture the inherent graphic patterns of “meaning” in images in terms of graphic objects. SVD, as a dimensionality reduction and

pattern extraction mechanism, manages to bring out these latent or explicit empirical syntactic patterns of occurrences of “elements” in their “contexts,” where the co-occurrence of the “elements” themselves defines the “context.”

Mathematics, as a formal language, has characteristics very different from natural language. It has almost no ambiguity and very precise syntactic-semantic relationships. It is the primary symbolic language employed by designers for constructing representations in design optimization. Initial modeling choices made by a designer may, however, actually hide some of the relations between variables, parameters, objective functions, and constraints.

We, therefore, propose the following hypothesis – the mathematical-symbolic representation of a design work implicitly captures not only the semantic concepts and choices employed by the designer but also the structural-behavioral-functional characteristics of the design work being modeled. *The problem representation captures design semantics in latent and explicit manners. The variables and parameters as design concepts, representing physical/structural elements, occur in “context” of each other, capturing functional and behavioral relationships through the syntactic structure of the formulation.* Changes in this representation as the optimization process passes through formulations and reformulations will reflect how the modeling of the engineered object changes, as well as how the designer changes his / her choices and decisions on the modeling.

There is some support for this hypothesis from research applying natural language processing to design. Dong and Agogino (1997) use computational linguistics based methods to explore construction of design representations from textual documentation. Dong (2005) applied the Latent Semantic Indexing method for document analysis to show that designers develop a “shared understanding” of a design object that is inherently captured in textual design documentation. Moss et al. (2004b), through empirical studies on differences in expert and novice behavior in designers, propose that internal and external knowledge representation mechanisms and changes to them capture the structure and content of a domain and internal cognitive processes. In all these studies, the implicit assumption is that the textual-syntactic representation of a design captures both the semantic choices exercised by designers as well as objective knowledge about the design object itself. These provide evidentiary support for our hypothesis, as we conceptually replace “words” for “variables and parameters” as elements in contexts of “sentences” for “functions, objectives, and constraints”.

As a machine learning problem, this hypothesis provides an interesting mathematical insight – SVD can be used to extract empirical patterns of syntactic co-occurrences of variables and parameters in objective and constraint functions. By observing and inferring patterns imputed from different dimensional reductions of the original data, one can reach a variety of interesting conclusions for optimization. This basic pattern extraction view reduces the need for domain specific knowledge-engineering for specific optimization tasks, or the need for providing a large database of solved examples to teach the system specific design knowledge and optimization problem formulation strategies. As we will demonstrate, the same general algorithmic mechanism can be used to infer and

learn a variety of design tasks – select variables, parameters and functions for problem formulation, model decomposition, identify dependent and independent variables and parameters, etc.

This approach seems particularly relevant for problem formulation decisions for medium to large scale problems, where the number of design variables, parameters and constraints are too large for choices to be determined by simple novice reasoning or direct human observation of design relationships. Moreover, such statistically based formulation decisions can scale to very large problems since previous “real world” formulation cases can incrementally become training data for the algorithm, thereby refining the learning with time. Because the algorithm is knowledge lean, it requires very few cases to learn and generalize knowledge for large classes of problems sharing similar characteristics.

## METHODOLOGY

### Data representation

This research assumes that design problem formulation is available as input to the algorithm in the following standard mathematical form for optimization models:

$$\begin{aligned} \text{Min } & \mathbf{f}(\mathbf{x}, \mathbf{P}) \\ \text{Sub to } & \mathbf{g}(\mathbf{x}, \mathbf{P}) \leq \mathbf{0} \\ & \mathbf{h}(\mathbf{x}, \mathbf{P}) = \mathbf{0} \\ & \mathbf{x}, \mathbf{P} \in \mathcal{X} \subseteq \mathbf{R}^n \end{aligned} \quad (1)$$

Here,  $\mathbf{x}$  is the vector of design variables and  $\mathbf{P}$  is the vector of design parameters that are kept fixed for one design model. These belong to a subset of the real space  $\mathbf{R}^n$ .  $\mathbf{f}$  is the vector of objective functions, for a single objective problem this will be  $f$ .  $\mathbf{g}$  and  $\mathbf{h}$  are vectors of inequality and equality constraints.

This general problem model is now converted into a matrix  $A$  for input into the SVD algorithm. Since variables and parameters are elements forming the objective and constraint functions, the occurrence matrix is produced as follows: the rows of  $A$  are the variables and parameters, the columns of  $A$  are the objectives and constraints, each entry  $A_{ij}$  is either a 1 or a 0 depending upon whether or not a particular variable or parameter  $i$  occurs in objective or constraint  $j$ . Figure 1 shows an example formulation for a problem (Michelana and Papalambros, 1997) and the respective occurrence matrix  $A$ .

$$\begin{aligned} \text{Min } & f = f_1 + f_2 \\ \text{h1: } & f_1 = x_1 + \exp(x_1 x_4) \\ \text{h2: } & f_2 = 2x_2 + 4x_5 \\ \text{h3: } & x_1 + 2x_2 + 5x_5 - 6 = 0 \\ \text{h4: } & x_1 + x_2 + x_3 - 3 = 0 \\ \text{h5: } & x_4 + x_6 - 2 = 0 \\ \text{h6: } & x_1 + x_4 - 1 = 0 \\ \text{h7: } & x_2 + x_5 - 2 = 0 \\ \text{h8: } & x_3 + x_6 - 2 = 0 \end{aligned}$$

	$f$	$h1$	$h2$	$h3$	$h4$	$h5$	$h6$	$h7$	$h8$
$x_1$	0	1	0	1	1	0	1	0	0
$x_2$	0	0	1	1	1	0	0	1	0
$x_3$	0	0	0	0	1	0	0	0	1
$x_4$	0	1	0	0	0	1	1	0	0
$x_5$	0	0	1	1	0	0	0	1	0
$x_6$	0	0	0	0	0	1	0	0	1
$f_1$	1	1	0	0	0	0	0	0	0
$f_2$	1	0	1	0	0	0	0	0	0

**Figure 1: Model based decomposition problem and the data matrix A**

All design models presented in this paper are modeled by analytical functional representations consisting of non-linear, continuous equations. As this method operates solely on measurements of co-occurrences of variables and constraints in context of each other, the only information required to

generate the occurrence matrix is whether a variable is related, in any way, to a constraint or another variable.

**Performing SVD on the data matrix**

SVD takes a general rectangular matrix  $A$  with  $m$  rows and  $n$  columns and decomposes it into a product of three matrices,  $A = USV^T$ , where  $U$  ( $m \times m$ ) and  $V$  ( $n \times n$ ) are the orthogonal matrices with columns that are left and the right singular vectors and  $S$  ( $m \times n$ ) is a rectangular matrix with singular values on the diagonal that can be arranged to be non-negative and in order of decreasing magnitude. The number of singular values is  $r$ , where  $r$  is the rank of  $A$ .

The mathematical idea (Strang 2003) is that the row space of  $A$  is  $r$ -dimensional and inside  $R^m$  and the column space of  $A$  is also  $r$ -dimensional and inside  $R^n$ . We now choose special orthonormal bases  $V = (v_1, v_2, \dots, v_r)$  for the row space, and  $U = (u_1, u_2, \dots, u_r)$  for the column space, such that  $Av_i$  is in the direction of  $u_i$ , with  $s_i$  providing the scaling factor, i.e.  $Av_i = s_i u_i$ . In matrix form, this becomes  $AV = US$  or  $A = USV^T$ .  $A$  is thus the linear transformation that carries orthonormal basis  $v_i$  from space  $R^n$  to orthonormal basis  $u_i$  in space  $R^m$ , where the singular values expand or contract – scale the normal vectors that are being carried from one space to the other. In this research, any general matrix  $A$  where the rows and columns stand for design concepts (variables, parameters) and functions (objectives, constraints), respectively, can be decomposed into independent principal components represented by a set of new, abstract variables (normal vectors forming the orthonormal bases) that represent combinations / correlations between the variables of the original data set. SVD analysis on a matrix produces a distributed re-representation of the original data that can be re-combined to generate the original data perfectly, with the singular values providing information on how these normal vectors are scaled to constitute the original data. The new abstract variables produced contain within themselves information with regard to all correlations between the original data. If only one entry in the matrix is changed, then this is enough to produce changes in all of the components. Figure 2 shows the results of SVD performed on the data matrix in Figure 1.

**Dimensionality reduction**

The main claim in this research is that performing SVD on a design optimization problem formulation occurrence data matrix (data matrix for short) followed by re-representing that data in a dimensionally reduced form can bring out latent or explicit relationships existing between design variables, parameters, objectives and constraints. Figure 2 shows SVD performed on the data matrix  $A$  in Figure 1. If we retain the first  $k$  singular values and compute a truncated SVD, it will be a least squares approximation of the original one. For example, preserving the first 2 dimensions for this small example problem produces the truncated SVD shown in Figure 3. Note how all the 1s and 0s have been transformed, scaled higher or lower in terms of “stretched” or “contracted” mutual correlations between the entries, implying that some entries where a 0 or a 1 signified no relationship or a perfect relationship between the two respective quantities is now changed to show a higher or lower (positive or negative) correlation. For larger problems, one may experiment with

more number of dimensions. The choice of the number of singular values to preserve is important. Deciding the correct number of dimensions to retain for different families of problems is a matter of experimentation. It should be noted here that the “error” in the least squares approximation is what is interesting in terms of the latent relationships revealed. This method makes no attempt to reduce any “error”. In fact, the claim is that different degrees of dimensionality reduction, or “errors,” will reveal different latent relationships between the design variables, constraints and objectives. A larger “error” might actually turn out to be more useful in certain cases.

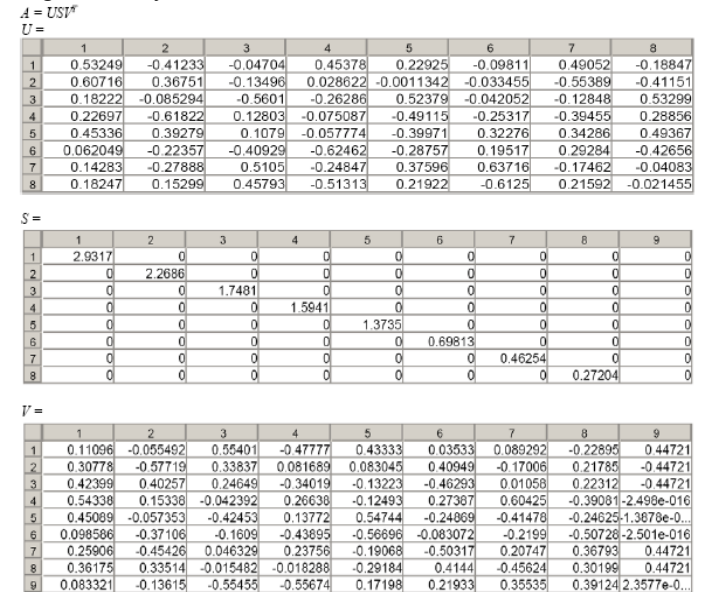


Figure 2: SVD performed on data matrix  $A$  in Figure 1

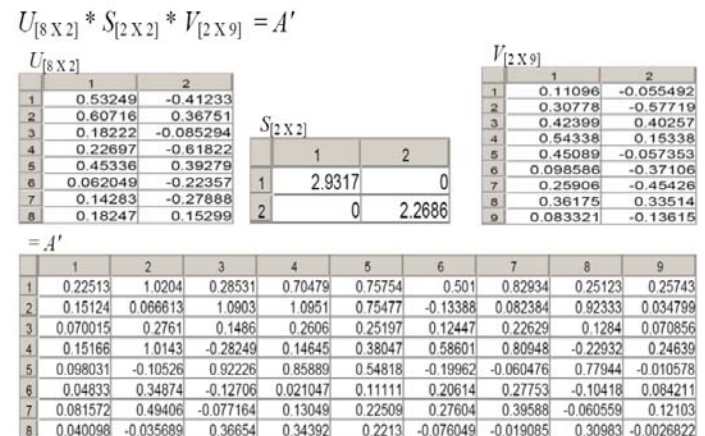


Figure 3: Truncated SVD with retained  $k = 2$  for  $A$

**Generating distributed graph representations: conceptual re-representation of data**

In the 2D and 3D cases, i.e. for  $k = 2$  and 3, it is helpful to visualize the dimensionality reduction in the form of graphs as they provide interesting insights into the “meaning” of the method. For example, when  $k = 2$ , if the first column of  $U$  is multiplied with the first singular value  $s_1$ , and the second column of  $U$  is multiplied with the second singular value  $s_2$ , then we get  $x$  and  $y$  coordinates for representing all the variables and parameters in a 2D plane. Similarly, if  $s_1$  and  $s_2$  are multiplied respectively with the first and second columns

of  $V$ , we will get  $x$  and  $y$  coordinates for all the objectives and constraint functions on the 2D plane. This 2D representation gives a visual idea of how the design elements and functions are related to each other in the reduced 2D space. In higher dimensional spaces, greater than 3, the relationships cannot be directly visualized, but the same computations will be relevant. Figure 4 shows the 2D distributed graph representation for the example shown in Figure 1.

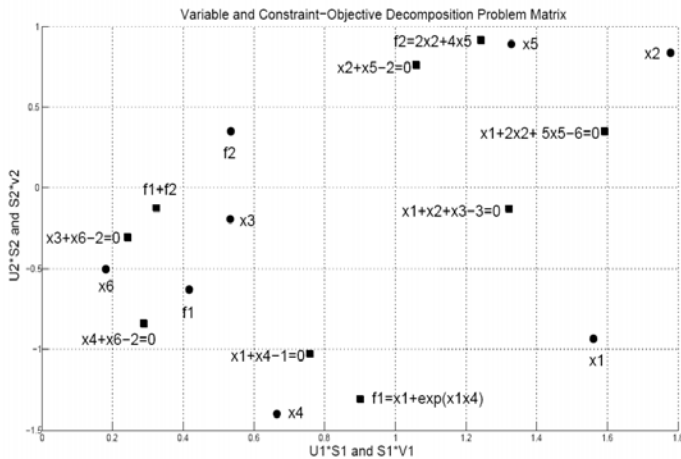


Figure 4: Distributed graph representation for design elements and functions in 2D space

### Inferring design (re)-formulation semantics

We now present mechanisms of querying the re-represented data in various ways for inferring various problem formulation semantics based on two basic mathematical tools – (i) cosine similarity measurements between variables, parameters and functions signifying inference of “semantic distance” between variables, parameters and functions, and (ii) k-means clustering on distributed graph representations to infer “semantically related groups” of variables, parameters and functions.

In general, the original data matrix for most problems will be sparse, because all variables and parameters do not appear in all objectives and constraints. The strength of the method lies in how SVD transforms the 0s and 1s into “stretched” or “contracted” relationships – computing cosine similarities between two elements or functions using the reduced dimensionality representation will bring out relationships between those variables, parameters and functions that are not explicitly related to each other in the syntactic formulation. For example, consider from Figure 1 that the variables  $x_1$ ,  $x_2$ ,  $x_4$  and  $x_5$  appear in functions  $f_1$  and  $f_2$ , but not directly in objective function  $f$ . It is “intuitive” from our knowledge of algebra that  $f$  could just be re-written in terms of  $x_1$ ,  $x_2$ ,  $x_4$  and  $x_5$ , but such an “intuition” is not something that can be taught to an automated system without an explicit rule about algebraic substitution. Thus, in the original matrix  $A$  (Figure 1), the data entry  $A_{11}$  (relationship between  $x_1$  and  $f$ ) is 0, and the data entry  $A_{12}$  (relationship between  $x_1$  and  $f_1$ ) is 1. These are explicit syntactic relationships. However, it is quite obvious that semantically  $x_1$  and  $f$  are correlated in a latent way, because  $f_1$  appears in  $f$ . Observing the same two entries in the 2D-truncated  $A'$  (Figure 3) shows that now  $A'_{11}$  is 0.22513 and  $A'_{12}$  is 1.0204, showing that the 0-1 relationship has transformed into a non-zero relationship between the three

quantities, one that may not be observed explicitly by a computational system. This observation is interesting because it shows that the SVD method does indeed reveal “hidden” relationships between the design elements and functions while maintaining the explicitly stated ones. On this basis, the designer can query the system for the following types of formulation decisions.

### Inferring selection of design variables and parameters

One of the most important design formulation and reformulation decisions that designers make based on their past experiences is which quantities should be chosen as design variables and which ones should be design parameters. Moreover, since design variables and parameters are often semantically related to each other (“the wall thickness of the hydraulic cylinder (variable  $t$ ) should not be less than a certain minimum wall thickness (parameter  $T$ )” or “the internal stress in the cylinder (variable  $s$ ) should always be less than a certain maximum stress (parameter  $S$ )”), designers have to take decisions on which set of other variables or parameters become important. These decisions must be considered in conjunction with a particular variable or parameter they have finalized as part of the model.

Figure 4 shows the case where only 2 dimensions have been preserved in order to re-represent data for a problem (Figure 1) that has 8 design variables and 9 functions. All these component variables and functions have now been projected onto a 2D space. Calculating the cosine between any two variables will reveal how “close” or “distant” two variables lie from each other. Consider the relationship between variables  $x_2$ ,  $x_4$  and  $x_5$  in the example problem. As can be directly observed from Figure 1,  $x_2$  and  $x_4$  do not co-occur together in any constraint, while  $x_2$  and  $x_5$  co-occur together in multiple constraints. A simple cosine calculation between the two is demonstrated to bring out how the 2D representation captures this relationship:  $x_2$  and  $x_4$  have a very low semantic correlation 0.0050, while  $x_2$  and  $x_5$  show a high semantic correlation.

Variable	x-coordinate	y-coordinate	cosine with $x_2$
$x_2$	1.78	0.83374	1.0
$x_4$	0.66542	-1.4025	0.0050
$x_5$	1.3291	0.8911	0.9884

Now, if the designer inputs a particular variable as a query, its cosine similarity with each of the other variables and parameters may be calculated. A cosine threshold may be decided upon, and all variables higher than a certain threshold level can be returned as semantically related to the query variable. The cosine threshold is a matter of experimentation, as having a higher threshold will return fewer cases and vice versa. In this example, setting the cosine threshold as 0.7, and setting  $x_2$  as the query variable returns the set  $\{x_3, x_5$  and  $f_2\}$ . This can be validated from Figure 1 – the variable  $x_2$  co-occurs with these three variables in functions  $\{h_2, h_3, h_4$  and  $h_7\}$ . As we will see in the experiments and results section, such queries on previous design formulation examples from a similar class of design problems returns cases of variables and parameters considered semantically related.

### Inferring selection of objectives and constraints

A similar query is possible for objectives and constraints – if the designer is considering fixing a particular variable or parameter, and wants to know which objectives and constraints should also be considered. For example, consider the query variable  $x_1$  with cosine threshold 0.7. The table below shows that the function set returned is  $\{f, h1, h3, h4, h5, h6, h8\}$ .

Cosine measures	$f$	$h1$	$h2$	$h3$	$h4$	$h5$	$h6$	$h7$	$h8$
$x_1$	0.9855	0.9100	0.3869	0.7283	0.9040	0.7647	0.9227	0.3977	0.9353

This is an interesting result because even though  $x_1$  does not occur directly in  $h5$  and  $h8$ , but because  $x_1$  has high correlations between both  $x4$  and  $x3$ , which do occur in  $h5$  and  $h8$  respectively, these indirect latent relationships are retrieved. Similar queries are also possible between function-function combinations.

Figure 5 shows the variables and functions that are returned with  $x_1$  as the query variable and 0.7 as the cosine threshold.

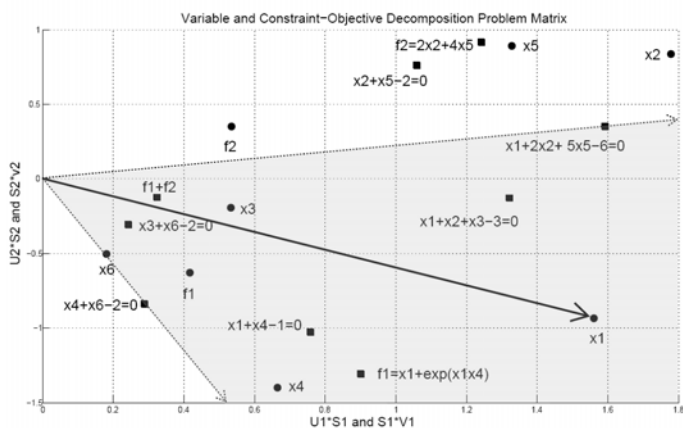


Figure 5: Variables and functions returned with query variable  $x_1$  and cosine threshold 0.7

### Inferring model-based decomposition using K-means algorithm

Decomposing large optimization problems into smaller ones is useful because large model sizes reduce the reliability and speed of numerical solution algorithms (Michelena and Papalambros, 1997). Many optimization problem models can be decomposed into independent sub-problems characterized by local variables and constraints. They become part of the main problem through linking variables and constraints. It is useful to identify these sub-problems because they can be solved independently, thereby simplifying the model. The example problem in Figure 1 is formulated and solved by Michelena and Papalambros (1997) as a model-based decomposition problem. They solve it using hypergraph partitioning. The problem is optimally partitioned into weakly connected sub-graphs.

Here, we show that performing a k-means clustering on the lower dimensional representation produced by the SVD algorithm can successfully identify these independent sub-problems, with local variables and constraints and the linking variables and constraints. The advantage of the method is that it is particularly simple to implement. For large scale problems, it may prove to be a satisfactory preliminary method for understanding decomposition strategies for the model. The k-means clustering algorithm is an iterative

method for putting  $N$  data points in an  $I$ -dimensional space into  $K$  clusters, where each cluster is parameterized by a vector  $\mathbf{m}^{(k)}$  called its mean (Mackay, 2003). The data points are vectors denoted by  $\mathbf{x}^{(n)}$ , where  $n = 1:N$ , each  $\mathbf{x}$  has  $I$  components  $x_i$ . A metric is defined for measuring distances between these data points, the simplest being Euclidean distance. For this work, we have used a cosine distance between data points. K-means is an iterative two-step algorithm, wherein an assignment step for each data point is assigned to the nearest mean (or centroid). In an update step, the means are adjusted to match the sample means of the data points that they are responsible for. This algorithm always converges to a fixed point, but initializing the means at different points can produce different clusters for the same data set.

We applied the k-means algorithm with cosine distance measurements onto the example problem shown in Figure 1. Figure 6 shows the results. It is interesting to note that the results almost exactly match with the results in the paper by Michelena and Papalambros – subproblem 1 with local variables  $\{x_4, x_6\}$  and functions  $\{h1, h5, h6, h8\}$ , subproblem 2 with local variables  $\{x_2, x_5\}$  and functions  $\{h2, h3, h4, h7\}$  and  $\{x_1, x_3\}$  as linking variables. In our results, the only difference is that  $h4$  is part of the cluster with the linking variables.

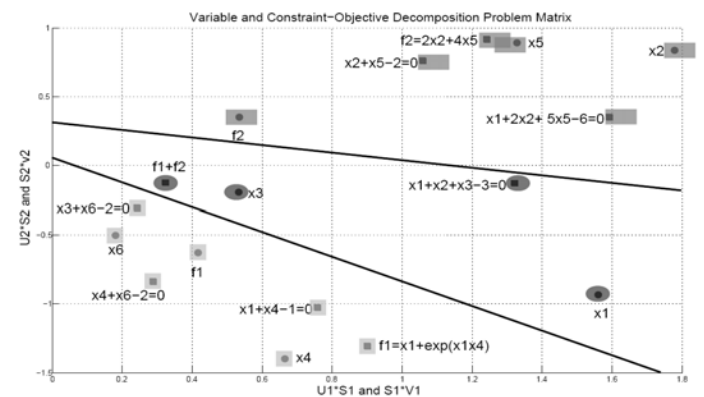


Figure 6: Model-based decomposition using SVD algorithm and k-means clustering

## EXPERIMENTS AND RESULTS

The methodology section presented the learning and inference mechanism in detail, with a demonstration and results on an example model-based decomposition problem. We tested the performance of the method in two ways – (i) testing the generality of the method by applying it to models from varied design domains and (ii) studying parametric effects of the method in relation to variations in problem size and dimensionality reduction.

We have applied the method on two other design domains – Hydraulic Cylinder Design (HCD) domain and Aircraft Concept Sizing (ACS) domain. We present the experiments and results in this section.

### The Hydraulic Cylinder Design (HCD) Problem

The HCD problem has been formulated and solved in multiple ways. Here, we consider two formulations of the problem – a single objective formulation (Papalambros and Wilde, 2000) and a multi objective formulation (Michelena

and Agogino, 1988). Figure 7(a) shows the design problem and 7(b) shows the two problem statements with the design variables, parameters, constraints and the corresponding data matrices.

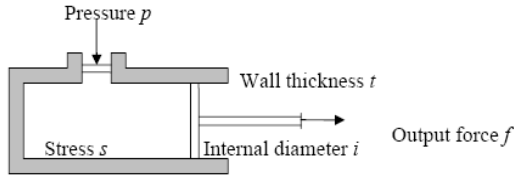


Figure 7(a): The Hydraulic Cylinder Design Problem

<b>Single-objective formulation:</b>		<b>Multi-objective formulation:</b>	
<b>Design variables:</b>	<b>Design model:</b>	<b>Design variables:</b>	<b>Design model:</b>
Internal diameter $i$	Min $i + 2t$	Internal diameter $i$	Min $Z = z1w1 + z2w2 + z3w3$
Wall thickness $t$	$g1: t - T \geq 0$	Wall thickness $t$	$z1 = \pi (it + t^2)$
Output force $f$	$g2: f - F \geq 0$	Output force $f$	$z2 = s/S$
Stress $s$	$g3: p - P \leq 0$	Stress $s$	$z3 = p/P$
Pressure $p$	$g4: s - S \leq 0$	Pressure $p$	$t - T \geq 0$
	$h1: f = (\pi/4)i^2p$	Objective $Z$	$f - F \geq 0$
	$h2: s = ip/2t$	Objective $z1$	$p - P \leq 0$
		Objective $z2$	$s - S \leq 0$
		Objective $z3$	$f = (\pi/4)i^2p$
		Weight $w1$	$s = p/E (it/2t + 0.6)$
<b>Design parameters:</b>		Weight $w2$	
Min Wall thickness $T$		Weight $w3$	
Min Output force $F$		<b>Design parameters:</b>	
Max Stress $S$		Min Wall thickness $T$	
Max Pressure $P$		Min Output force $F$	
		Max Stress $S$	
		Max Pressure $P$	
		Joint efficiency $E$	

Figure 7(b): Single and multi-objective formulations for the HCD Problem

### Figure 7: The Hydraulic Cylinder Domain

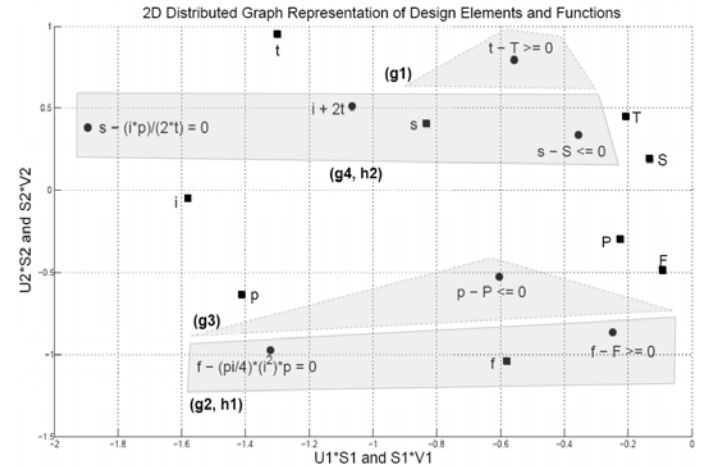
The SVD based algorithm was applied onto these two problems formulations. As can be observed, the single objective form (matrix  $A$  has dimensions  $9 \times 7$ ) has fewer number of variables and parameters as compared to the multi-objective form (matrix  $A$  has dimensions  $17 \times 10$ ). We report the following results:

### Results provided by SVD algorithm compared to results from source examples:

Figure 8(a) shows the 2D graph ( $k = 2$ ) produced for the single-objective case by performing SVD analysis on the original problem matrix followed by cosine measurements and applying the k-means algorithm for semantic groupings. Figure 8(b) shows the 3D graph ( $k = 3$ ) for the same dimensional reduction performed on the multi-objective formulation. The performance of the algorithm and the validity of the results was measured by comparing the results returned in each case (semantic groups of variables and constraints) to those from the source documents. The best dimension was decided on the basis of how the results match the cases given in the source documents, because the source documents offer proof that this solution method provides optimal solutions.

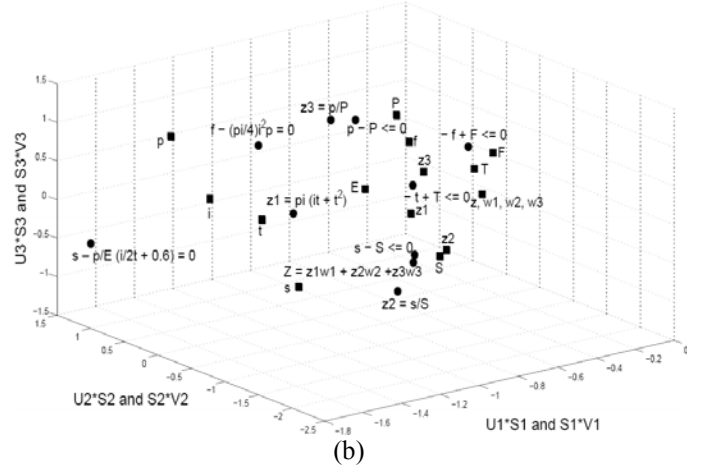
The original sources for these problems use monotonicity analysis to solve these problems. In each formulation, there are 5 design variables and 6 design constraints. There will be “design cases” as the number of active, non-redundant constraints cannot exceed the number of design variables. The most interesting result to report is that the semantic groupings provided by the SVD based algorithm in both cases matches with the design “cases” identified in the source documents. For example, in the single objective case, Papalambros and Wilde (2000) report these cases as pressure-bound, stress bound and thickness bound. The results show that for design

variable  $i$  (internal diameter) either constraints ( $g3, (g2, h1)$ ) or ( $(g4, h2), (g2, h1)$ ) will be active, and for variable  $t$  (wall thickness) either constraints ( $(g4, h2), (g2, h1)$ ) or ( $g1$ ) will be conditionally critical. Figure 8(a) shows that the SVD analysis followed by cosine measurements shows similar conclusions by purely a syntactic analysis of design formulation – constraints ( $g2, h1$ ) and ( $g4, h2$ ) form distinct visual groups, with constraints  $g3$  and  $g1$  falling close to these two groups. A similar result may also be observed for the multi-objective case. Michelena and Agogino (1988) identify three design cases: Case P (pressure inactive), Case S (stress inactive), and Case PS (pressure and stress inactive) with all of them having force  $f$  and thickness  $t$  active. Figure 8(b) shows that pressure and stress groups are distinctly apart, with force and thickness falling in the middle (as supported equivalently by cosine or k-means calculations).



8(a)

Variable-Parameter and Constraints-Objectives Multiobjective Hydraulic Cylinder Design Matrix



(b)

Figure 8: (a) 2D graph, single-objective form (b) 3D graph, multi-objective form

### Effect of problem size and number of dimensions retained $k$ :

Figure 9(a) shows that when the number of dimensions  $k$  is fixed at 2, the multi-objective version returns a greater number of variables and parameters as answers to queries, as compared to the single objective form. Figure 9(b) shows that for the multi-objective form, when  $k$  is increased from 2 to 3 to 4, the number of answers to the queries on design variables

reduces. These results, in combination, imply that (i) for different sized problems of the same family, the algorithm will return different cases; and, (ii) by increasing the number of dimensions retained  $k$ , one can observe different semantic patterns in these groupings. The implication is that when designers query different samples from the database, they will be able to see how different designers have semantically grouped variables and constraints. It is a matter of trial and error and experimentation to decide for one particular design domain what the best value of  $k$  is. For example, we show in the previous section that for the HCD domain, we found dimensions 2 and 3 the best ones, with performance of the algorithm deteriorating for 4 and above.

Two general heuristics were identified based on these simple experiments – (i) A good value of  $k$  is one that would help to “correctly” decompose a not-well-formulated problem into smaller well-formulated ones (as in the case of the model-based decomposition problem, or in identifying “cases” in the hydraulic cylinder problem); and, (ii) the smaller the size of the problem, the smaller the required  $k$  value; the larger the size of the problem, the larger the required  $k$  value for identifying relevant semantic groups.

It is obvious that when  $k$  equals the original number of dimensions (as matrix A) then the semantic groups return only the explicit information contained in the original data matrix. With reduced dimensionality representations, the algorithm captures semantic relationships that are not explicitly observable from the original matrix. The significance of the method lies in its ability to use the syntax of similar problems to provide help in stating well-formulated design optimization problems in terms of “good” groupings of variables and constraints. One of the biggest problems with formulating problems in optimization is that the designer does not know in advance what the best formulation is.

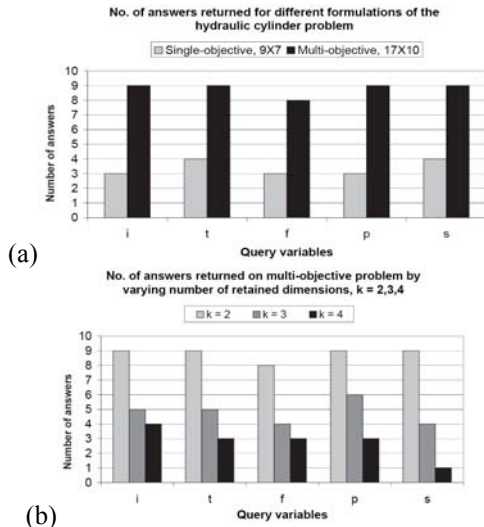


Figure 9: (a) Effects of problem size (b) Effects of number of dimensions  $k$

### The Aircraft Concept Sizing (ACS) Domain

Applying the SVD algorithm to the Aircraft Concept Sizing (ACS) problem was an experiment in applying the method to a medium-large sized problem. Given the statistical basis of the algorithm, a primary significance of the method presented here is its potential application to very large scale

problems, where the design models are too large in terms of the numbers of variables, parameters, and constraints for the designer to take formulation decisions based on direct observations. This method can prove to be a valuable preliminary analysis method based on previous experiences to provide the designer insight on semantic groupings of variables, parameters and constraints for new problems in a similar domain.

The source for the ACS problem is Gu et al. (2002). Gu formulates it as a decision based collaborative optimization problem, bringing in collaborative optimization (CO), decision based design (DBD) and multi-disciplinary optimization (MDO) in a single framework. Figure 10 shows the formulation of the problem, with 8 variables ( $x_1 - x_8$ ), 10 system states ( $x_9 - x_{18}$ ) and 5 sub-problems (SP1 – SP5) that represent 5 domains (aerodynamics, weight, performance, cost and business). System states identify couplings between the design sub-domains, as outputs from one sub-domain can be inputs to other domains. The problem has been solved numerically in their paper using a Sequential Quadratic Programming method. However, the focus for our method is primarily symbolic modeling and (re)-formulation, for which we intend to show how syntactic arrangements of variables and constraints can show semantic groupings intended by the designer and design characteristics of the design object being modeled. The objective of the main problem is to maximize the Net Revenue (or minimize the negative of the Net Revenue). The main problem contains compatibility constraints that become objectives for the sub-problems.

Design Variables	Description
$x_1$	Aspect ratio of wing
$x_2$	Wing area (ft <sup>2</sup> )
$x_3$	Fuselage length (ft)
$x_4$	Fuselage diameter (ft)
$x_5$	Density of air cruise altitude (slugs/ft <sup>3</sup> )
$x_6$	Cruise speed (ft/sec)
$x_7$	Fuel weight (lbs)
$x_8$	Price

Figure 10(a): Design variables for the ACS problem

Design states	Description
$x_9$	Total aircraft wetted area (ft <sup>2</sup> )
$x_{10}$	Max lift to drag ratio
$x_{11}$	Empty weight (lbs)
$x_{12}$	Gross take off weight (lbs)
$x_{13}$	Aircraft range (miles)
$x_{14}$	Stall speed (ft/sec)
$x_{15}$	Fuselage volume
$x_{16}$	Demand
$x_{17}$	Total cost
$x_{18}$	Net Revenue

Figure 10(b): States for the ACS problem

Design Variables	Sub-problem 1 (Aerodynamics)	Sub-problem 2 (Weight)	Sub-problem 3 (Performance)	Sub-problem 4 (Cost)	Sub-problem 5 (Business)
$x_1$	1	1	0	0	0
$x_2$	1	1	1	1	0
$x_3$	1	1	0	1	0
$x_4$	1	1	0	1	0
$x_5$	0	1	0	0	0
$x_6$	0	1	0	1	1
$x_7$	0	1	1	1	0
$x_8$	0	0	0	0	1

Figure 10(c): Shared and independent variables in the individual sub-problems

Design States	Sub-problem 1 (Aerodynamics)	Sub-problem 2 (Weight)	Sub-problem 3 (Performance)	Sub-problem 4 (Cost)	Sub-problem 5 (Business)
$x_9$	1	0	0	0	0
$x_{10}$	1	0	1	0	0
$x_{11}$	0	1	0	0	0
$x_{12}$	0	1	1	0	1
$x_{13}$	0	0	1	0	1
$x_{14}$	0	0	1	0	1
$x_{15}$	1	0	0	0	1
$x_{16}$	0	0	0	1	1
$x_{17}$	0	0	0	1	1
$x_{18}$	0	0	0	0	1

Figure 10(d): Shared and independent states in the individual sub-problems

### Figure 10: The Aircraft Concept Sizing Problem

We applied the SVD algorithm to this problem, varying the number of dimensions from 2 to 4 ( $k = 2, 3, 4$ ) in order to study the variations in the number of correct cases returned and the number of missed cases not returned by the algorithm. Since this is a collaborative optimization problem example, similar to the decomposition problem, the algorithm should produce groupings of shared and independent variables, i.e., which variables and states are shared or occur independently in which sub-problems. The main aim in this experiment was

to study at what reduced dimensionality the algorithm successfully returns semantic groupings as expressed in the problem statement.

Figure 11 shows that the problem is characterized by overlapping variables and states between sub-problems – the shaded regions in each reduced dimensionality graph ( $k = 2$ ) show the variables and states shared by each individual sub-problem. Visually overlapping two or more regions to visualize these relationships would show that there are many hidden and explicit relationships shared between the sub-problems.

We found that the algorithm manages to capture at  $k = 2$  or 3 almost all the correct groupings, with very few missed cases. Performance only slightly improves over this at  $k = 4$ , showing that the reduced dimensionality representations capture the relationships between variables, states and sub-problems correctly. The algorithm, especially at  $k = 2$ , also returns some extra cases. We cannot classify these extra cases as correct or incorrect, as from previous experiments we know that the algorithm returns cases on the basis of relationships that both occur explicitly or latently in the problem syntax. Further research is required to explore the relevance of these extra cases returned by the algorithm. Figure 12 shows the (variables, states) by (sub-problems) cosine matrices for  $k = 2$  and 3.

We note again that, in the SVD method presented here, the  $k$ -reduced approximation is a linear least squares approximation of the original occurrence matrix, but unlike traditional “error” reduction approaches, a larger error in this case can be useful. As  $k$  decreases ( $k = n, n-1, n-2, \dots, 1$ ) the degree of error will increase, but the size of this error may result in latent relationships that may not be perceivable when the error is small. It is not a priori obvious is there is an optimal error. For example, we calculate the estimation error for cases  $k = 2, 3, 4$  and 5, by using infinity norm (largest row sum for the matrix), between the original matrix and the  $k$ -reduced approximations. For the original matrix, the norm is 4, for the 2-reduced matrix the norm is 3.8496, and for the 3-reduced matrix the norm is 3.9589, for the 4-reduced matrix the norm is 3.9404, for the 5-reduced (or original matrix) the norm is 4.0000 again. As expected, the error reduces (0.1504 to 0) when  $k$  increases (from 2 to 5). However, as observed above, the results obtained in terms of semantic groupings for the  $k = 2$  or 3 case were more, or at least as, relevant in terms of semantic groupings obtained for variables and constraints as for the  $k = 4$  or 5 cases.

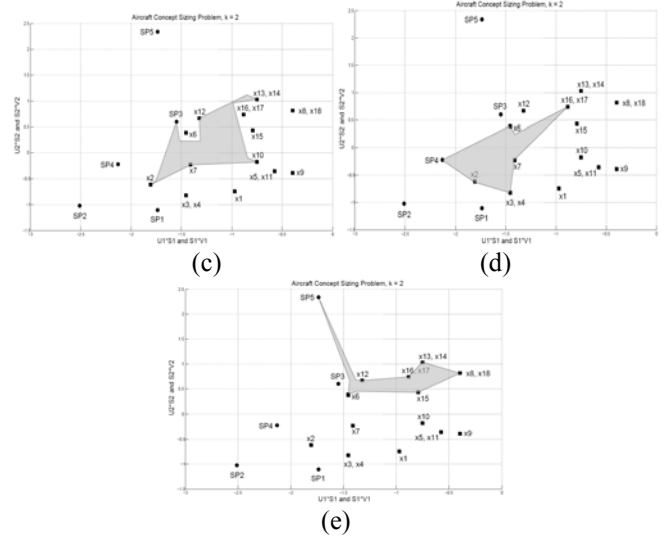
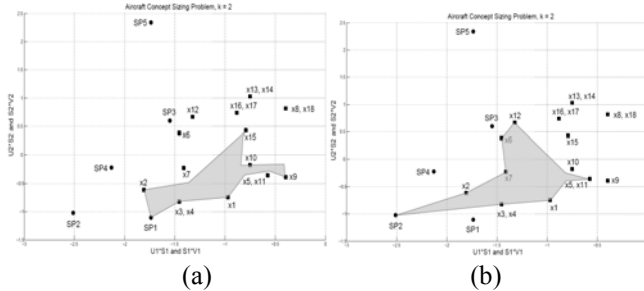


Figure 11: Shared variables and states in the ACS problem (a) SP1 (b) SP2 (c) SP3 (d) SP4 (e) SP5

	1	2	3	4	5	1	2	3	4	5
1	0.99597	0.96363	0.51875	0.85106	-0.018297	0.97116	0.90366	0.51536	0.6431	-0.032207
2	0.97174	0.99834	0.7666	0.97487	0.30481	0.94816	0.93785	0.69876	0.75127	0.28393
3	0.99871	0.99158	0.83348	0.91627	0.12205	0.8245	0.99153	0.36561	0.8781	0.13234
4	0.96871	0.99158	0.83348	0.91627	0.12205	0.8245	0.99153	0.36561	0.8781	0.13234
5	0.99997	0.98502	0.59931	0.89803	0.078875	0.75921	0.97707	0.25456	0.89258	0.096636
6	0.67907	0.79953	0.99382	0.93541	0.78074	0.33881	0.76383	0.34669	0.94999	0.71501
7	0.91836	0.97453	0.86347	0.99843	0.46039	0.81821	0.98899	0.62885	0.90077	0.46183
8	-0.1175	0.062878	0.72963	0.33835	0.98213	-0.1515	0.078228	0.47896	0.34829	0.98158
9	0.97937	0.92527	0.41404	0.78302	-0.13638	0.95344	0.61948	0.65268	0.24581	-0.15113
10	0.9454	0.98858	0.82332	0.99144	0.38268	0.82729	0.42839	0.99052	0.10781	0.16204
11	0.99997	0.98502	0.59931	0.89803	0.078875	0.75921	0.97707	0.25456	0.89258	0.096636
12	0.50911	0.65549	0.99489	0.83989	0.8949	0.55209	0.58997	0.89351	0.60103	0.84636
13	0.061271	0.23964	0.83977	0.50055	0.99995	0.22171	0.1534	0.84963	0.20811	0.87809
14	0.061271	0.23964	0.83977	0.50055	0.99995	0.22171	0.1534	0.84963	0.20811	0.87809
15	0.48384	0.83333	0.99155	0.8238	0.90746	0.58927	0.53093	0.93546	0.51155	0.81951
16	0.29952	0.4661	0.9457	0.69407	0.97306	0.037023	0.47567	0.30893	0.76388	0.88255
17	0.29952	0.4661	0.9457	0.69407	0.97306	0.037023	0.47567	0.30893	0.76388	0.88255
18	-0.1175	0.062878	0.72963	0.33835	0.98213	-0.1515	0.078228	0.47896	0.34829	0.98158

Figure 12: Cosine matrices between  $x_1 - x_{18}$  and SP1 – SP5 for (a)  $k = 2$  (b)  $k = 3$

Figure 13 shows the details of the correct, extra and missed variables and states returned by the algorithm for  $k = 2, 3, 4$  with a cosine threshold of 0.7, and the queries set to the 5 sub-problems (tally observations for  $k = 2$  and 3 using Figure 12).

	Original	SP1	SP2	SP3	SP4	SP5
		$[x_1, x_2, x_3, x_4, x_6, x_{10}, x_{13}]$	$[x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_{11}, x_{12}]$	$[x_2, x_7, x_{10}, x_{12}, x_{13}, x_{14}]$	$[x_3, x_4, x_6, x_7, x_{16}, x_{17}]$	$[x_6, x_6, x_{12}, x_{13}, x_{16}, x_{17}, x_{18}, x_{16}, x_{13}, x_{18}]$
2D, $k = 2$ , cosine threshold = 0.7						
Exact match		$[x_1, x_2, x_3, x_4, x_6, x_{10}]$	$[x_5, x_6, x_7, x_{11}]$	$[x_{12}, x_{13}, x_{14}]$	$[x_3, x_4, x_6, x_7]$	$[x_6, x_6, x_{12}, x_{13}, x_{16}, x_{17}, x_{18}, x_{16}, x_{13}, x_{18}]$
Extra		$[x_5, x_7, x_{17}]$	$[x_8, x_{10}]$	$[x_6, x_6, x_{13}, x_{16}, x_{17}, x_{18}]$	$[x_1, x_3, x_6, x_{10}, x_{12}, x_{13}]$	None
Missed		$[x_{12}]$	$[x_{12}]$	None	$[x_{16}, x_{17}]$	None
3D, $k = 3$ , cosine threshold = 0.7						
Exact match		$[x_1, x_2, x_3, x_4, x_6, x_{10}]$	$[x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_{11}]$	$[x_2, x_{10}, x_{12}, x_{13}, x_{14}]$	$[x_3, x_4, x_6, x_7, x_{16}, x_{17}]$	$[x_6, x_6, x_{12}, x_{13}, x_{16}, x_{17}, x_{18}, x_{16}, x_{13}, x_{18}]$
Extra		$[x_5, x_7, x_{17}]$	None	$[x_{12}]$	$[x_3, x_{17}]$	None
Missed		$[x_{12}]$	$[x_{12}]$	$[x_7]$	None	None
4D, $k = 4$ , cosine threshold = 0.7						
Exact match		$[x_1, x_2, x_3, x_4, x_6, x_{10}, x_{13}]$	$[x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_{11}]$	$[x_2, x_{10}, x_{12}, x_{13}, x_{14}]$	$[x_3, x_4, x_6, x_7, x_{16}, x_{17}]$	$[x_6, x_6, x_{12}, x_{13}, x_{16}, x_{17}, x_{18}, x_{16}, x_{13}, x_{18}]$
Extra		None	None	None	$[x_5, x_{17}]$	None
Missed		None	$[x_{12}]$	$[x_7]$	None	None

Figure 13: Correct, extra and missed cases for dimensions  $k = 2, 3, 4$

We use two metrics to measure the performance of the algorithm – a *precision* metric and a *recall* metric. *Precision* is defined as the ratio of correct cases to total cases found. *Recall* is defined as the ratio of correct cases to the total number of correct cases that should have been found. Figure 14 shows the improvement in performance for  $k = 2$  (i.e.,

Precision<sub>2D</sub> and Recall<sub>2D</sub>) and  $k = 3$  and then a similar level of performance for  $k = 4$ . At  $k = 5$ , the matrix becomes the original matrix; Precision and Recall become 1. But, even at  $k = 2$  or  $k = 3$ , the answers returned by the algorithm capture the patterns of variable-state clusters in the problem. This is a significant advantage for very large scale problems, when the problem size is very large in terms of variables, parameters and states.

	SP1	SP2	SP3	SP4	SP5	Average
Precision <sub>2D</sub>	6/9 = 0.6667	8/10 = 0.8000	6/12 = 0.5000	5/12 = 0.4167	9/9 = 1.0000	0.6767
Precision <sub>3D</sub>	6/9 = 0.6667	8/8 = 1.0000	5/6 = 0.8333	7/9 = 0.7778	9/9 = 1.0000	0.8556
Precision <sub>4D</sub>	7/7 = 1.0000	8/8 = 1.0000	5/5 = 1.0000	7/9 = 0.7778	9/9 = 1.0000	0.9556

(a)

	SP1	SP2	SP3	SP4	SP5	Average
Recall <sub>2D</sub>	6/7 = 0.8571	8/9 = 0.8889	6/6 = 1.0000	5/7 = 0.7143	9/9 = 1.0000	0.8921
Recall <sub>3D</sub>	6/7 = 0.8571	8/9 = 0.8889	5/6 = 0.8333	7/7 = 1.0000	9/9 = 1.0000	0.9159
Recall <sub>4D</sub>	7/7 = 1.0000	8/9 = 0.8889	5/6 = 0.8333	7/7 = 1.0000	9/9 = 1.0000	0.9444

(b)

**Figure 14: (a) Precision and (b) recall measurements**

## CONCLUSIONS

We developed, demonstrated and tested an SVD-based method for design optimization problem (re)-formulation on various problem domains for optimization tasks focused on modeling, (re)-formulation, decomposition, “case” identification, and identification of shared, dependent and independent variables. The method was theoretically inspired by the intriguing observation that, in the domains of statistical natural language processing and digital image processing, the use of SVD brings out explicit or latent semantic patterns embedded in the syntax of representation. We hypothesized that SVD captures and reveals a more general characteristic – the semantic relations within a domain are embedded in the syntax of representation. In design optimization, variables and parameters representing occurrences in “context” of each other, capturing functional-behavioral relationships through the syntactic structure of the formulation.

As this method operates solely on measurements of co-occurrences of variables and constraints in context of each other, we are experimenting with other kinds of models where explicit analytical functional relationships may not be available (e.g., by numerical analysis or simulation models), models with discrete variables, and models with mixed continuous and discrete variables. The method in its current form is insensitive to the strength of mathematical relationship between two variables, focusing basically on the symbolic coupling of variables and constraints with each other. A case where two variables share a linear relationship and a case where two variables share an exponential relationship with each other would be treated as equivalent by the method in its current form. We are exploring ways in which the method may become sensitive to such order of magnitude issues.

Given the statistical nature of the algorithm, we believe that a potential application area for the method is for problems where the number of design variables and constraints and complexity of design relationships far exceed direct human observation and reasoning.

## ACKNOWLEDGMENTS

This research is supported by a Faculty of Architecture International Research Scholarship, University of Sydney.

## REFERENCES

- Cagan, J., Grossman, I.E. and Hooker, J., 1997, A Conceptual Framework for Combining Artificial Intelligence and Optimization in Engineering Design, *Research in Engineering Design*, **9**(1), 20-34.
- Campbell, M.I., Cagan, J. and Kotovsky K., 2003, The A-Design approach to managing automated design synthesis, *Research in Engineering Design*, **14**(1), 12-24.
- Dong, A. and Agogino, A.M., 1997, Text analysis for constructing design representations, *Artificial Intelligence in Engineering* **11**(2), 65-75.
- Dong, A., 2005, The Latent Semantic Approach to Studying Design Team Communication, *Design Studies* **26**(5), 445-461.
- Ellman, T., Keane, J., Banerjee, A. and Armhold, G., 1998, A transformation system for interactive reformulation of design optimization strategies, *Research in Engineering Design*, **10**(1), 30-61.
- Gelsey, A., Schwabacher, M. and Smith, D., 1998, Using modeling knowledge to guide design space search, *Artificial Intelligence*, **101**(1), 35-62.
- Gu, X., Renaud, J.E., Ashe, L.M., Batill, S.M., Budhiraja, A.M. and Krajewski, L.J., 2002, Decision-Based Collaborative Optimization, *Journal of Mechanical Design*, **124**(1), 1-13.
- Kalman, D., 1996, A singularly valuable decomposition: The SVD of a matrix, *The College Mathematics Journal*, **27**(1), 2-23.
- Landauer, T.K. and Dumais S.T., 1997: A Solution to Plato’s Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge, *Psychological Review*, **104**(2), 211 – 240.
- Michelena, N.F. and Agogino, A., 1988, Multiobjective Hydraulic Cylinder Design, *Journal of Mechanisms, Transmissions and Automation in Design*, **110**, 81-87.
- Michelena, N.F. and Papalambros, P.Y., 1997, A Hypergraph Framework for Optimal Model-Based Decomposition of Design Problems, *Computational Optimization and Applications*, **8**(2), 173-196.
- Moss, J., Cagan, J. and Kotovsky, K., 2004a, Learning From Design Experience in an Agent-based System, *Research in Engineering Design*, **15**(2), 77-92.
- Moss, J., Kotovsky, K. and Cagan, J., 2004b, Cognitive Investigations into Knowledge Representation in Engineering Design, in *Design Computing and Cognition '04*, J. Gero, ed., 97-116.
- Papalambros, P.Y. and Wilde, D.J., 2000, *Principles of Optimal Design*, Cambridge, MA, Cambridge University Press.
- Schwabacher, M., Ellman, T. and Hirsh, H., 1998, Learning to Set up Numerical Optimizations of Engineering Designs, *AIEDAM*, **12**(2), 173-192.
- Strang, G., 2003, *Introduction to Linear Algebra*, Wellesley, Wellesley-Cambridge Press.
- Williams, B.C. and Cagan, J., 1994, Activity Analysis: The Qualitative Analysis of Stationary Points for Optimal Reasoning, *Proceedings of 12th National Conference on Artificial Intelligence*, Seattle, Washington, 1217-1223.