DESIGN OPTIMIZATION PROBLEM REFORMULATION USING SINGULAR VALUE DECOMPOSITION

Somwrita Sarkar
Graduate Student
email: ssar3264@mail.usyd.edu.au
University of Sydney, Australia

Andy Dong
Senior Lecturer
email: andy.dong@usyd.edu.au
University of Sydney, Australia

John S. Gero
Research Professor
email: john@johngero.com
George Mason University, USA

ABSTRACT

This paper presents a design optimization problem reformulation method based on Singular Value Decomposition (SVD), dimensionality reduction, and unsupervised clustering. The method calculates linear approximations of the associative patterns of symbol co-occurrences in a design problem representation to infer *induced interaction/coupling strengths* between variables and constraints. Unsupervised clustering of these approximations is used to identify useful reformulations. These two components of the method automate a range of reformulation tasks that have traditionally required different solution algorithms. We explain the method using an analytic model-based decomposition problem, and apply the method to an analytic hydraulic cylinder design problem as an example of heuristic design "case" identification, and to non-analytic problems expressed in FDT and DSM forms as examples of design decomposition. An Aircraft Concept Sizing (ACS) problem is used to empirically validate the method's performance. The results show that the method can be used to infer multiple well-formed reformulations starting from a single problem representation in a knowledge-lean and training lean manner.

Keywords: design methodology, design automation, unsupervised learning, pattern extraction, SVD

INTRODUCTION

Design problem reformulation significantly affects the final results of any design optimization process. Artificial intelligence methods have been proposed to automate some problem formulation tasks in design optimization. Cagan et al. [1] present a review of methods that use artificial intelligence techniques for optimization in engineering design. Schwabacher et al. [2] explore inductive machine learning techniques such as decision tree induction to automate various tasks in design optimization. Other notable research exploring similar issues include work by

Ellman et al. [3] and Gelsey et al. [4]. Campbell et al. [5] present a design synthesis tool called A-Design in which agents in a multi-agent system evolve conceptual design objects using genetic algorithms, asynchronous teams and functional reasoning. A key shortcoming of these methods is that most either require a high level of knowledge engineering in the form of rules, heuristics, grammars and sophisticated computational procedures [3-6] or they require a large training database of solved examples of various problems [2, 7] to acquire or infer knowledge about the problem that is useful for modeling or reformulation purposes.

In contrast to these methods which rely on knowledge engineering or supervised learning, the theoretical basis of our method views design reformulation from an unsupervised learning and pattern extraction perspective. The main hypothesis is that *the mathematical-symbolic representation of a design work captures the structural-behavioral-functional characteristics of the design work being modeled as semantically conceived by the designer. The syntax of representation captures the semantics of a design in explicit and implicit manners.* While this representation may contain relevant knowledge about the design work, the mathematical formulation may not be well formed for optimization. Our conjecture is that the design information available in the design model is sufficient to reformulate the model into a well-behaved form. The pattern recognition perspective suggests that some types of formal rule-based knowledge (for example, interaction or coupling between variables and functions) are inherently encoded in the associative patterns that exist between symbols in a representation. This intuition is developed from ideas in statistical natural language processing (SNLP) [8, 9] and digital image processing (DIP) [10, 11].

First, the method takes a design problem formulation in analytical or non-analytical form and converts it into a common representation form – the *occurrence matrix* **A** that captures how symbols co-occur in functions or interaction/dependency mappings with each other. Second, it uses Singular Value Decomposition (SVD) of this occurrence matrix, $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$, to infer a set of two independent spaces (**U** and **V**) described by orthonormal vectors and singular values (**S**) that relate these two orthogonal spaces and capture the most important association patterns in **A**. The idea is that the explicit association information in the occurrence matrix has been re-represented as a set of linearly independent vectors that capture all the original information, but are themselves uncorrelated to each other. Each orthonormal vector can be used to represent, in an abstract way, the variables and functions as vectors in real space. Third, a dimensionality reduction is performed on this decomposition using the k largest singular values, which produces optimal k-rank linear least squares approximations of the original matrix. This step has the effect of strengthening the most important relationships in the original matrix ("pattern") and weakening the less important

relationships ("noise"). It "pulls together" in a lower dimensional space all those variables and functions that share mutually high interaction/coupling relationships. We call this the capture of implicit, global, associative information because such information is not directly evident from the explicit occurrence matrix representation. Using SVD and dimensionality reduction causes a set of discrete, explicit, local interaction relationships to be re-represented in a continuous real space, where distances and/or angles between design variables and functions are a measure of their interaction/coupling strength. Based on these associative strengths, the vector representations of the variables and functions in this dimensionally reduced space can be clustered to reveal sets of variables and functions that share high interaction/coupling with each other.

Thus, using special approximations of associations that exist between symbols, the method uncovers global, implicit knowledge of the design problem from the local, explicit information in the problem representation. This ability scopes the kind of reformulation tasks that it can perform – those in which a problem needs to be "seen" as decomposed in the process of reformulating it, as having coupled or interacting sub-structures. Example reformulation tasks that the method performs include selection of linked design variables, parameters and constraints, design decomposition analysis, modularity and integration analysis, design "case" identification, and constraint satisfaction tasks such as topology modeling and layout planning. In the next section, we present the method. We then apply it to sample problems to illustrate "case" identification and design decomposition.

METHOD

Step I: Data representation

The first step is to convert an analytical or non-analytical design optimization problem formulation into an occurrence matrix ***A.*** Analytical optimization models are stated in the common form:

$$\text{Min } \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{p}) \text{ subject to } \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{p}) \leq \boldsymbol{0}, \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{p}) = \boldsymbol{0}, \boldsymbol{x}, \boldsymbol{p} \in \chi \subseteq \boldsymbol{R}^n \quad (1)$$

Here, $\boldsymbol{x}$ is the vector of design variables and $\boldsymbol{p}$ is the vector of design parameters kept fixed for one design model; $\boldsymbol{x}$ and $\boldsymbol{p}$ belong to a subset of the real space $\boldsymbol{R}^n$. $\boldsymbol{f}$ is the vector of objective functions. $\boldsymbol{g}$ and $\boldsymbol{h}$ are vectors of inequality and equality constraints. Non-analytical models are expressed as Functional Dependence Tables (FDT) or as a Design Structure Matrix (DSM). In a DSM, the matrix is square, with mappings between the same elements. In an FDT, the matrix is rectangular with mappings between functions/attributes and variables/components. An FDT may be derived from analytical formulations or numerical simulation results. Both forms are subsequently converted into

the occurrence matrix **A**. Variables, parameters and/or design components are *events* forming the functional/interaction/dependency mappings, which are *episodes*. For analytical formulations and FDTs, the matrix **A** is produced as follows: the rows of **A** are events; the columns of **A** are episodes. Each entry A_{ij} is either a 1 or a 0 showing whether or not a particular event *i* occurs in episode *j*. The Jacobian of **f**, **g** and **h** may be calculated to identify a different set of relations as the starting point, but is not necessary for this method because only the syntactical association patterns of symbols are employed to infer interaction/coupling strengths. Further, the Jacobian is only defined for functions that are differentiable, while this method is applicable on any general mathematical form that may or may not be differentiable. Figure 1 shows an example formulation for a problem [12] and the respective occurrence matrix **A**. For DSM formulations, both the rows and the columns of the matrix **A** represent events, and the matrix entries capture interactions/dependency mappings between these events.

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

	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 occurrence matrix A

Step II: Performing SVD on the data matrix

The second step is to calculate the singular value decomposition (SVD) of the occurrence matrix **A**. SVD takes a general rectangular matrix **A** with *m* rows and *n* columns and decomposes it into a product of three matrices, $\mathbf{A} = \mathbf{USV}^T$, where $\mathbf{U}(m \times m)$ and $\mathbf{V}^T(n \times n)$ are the left and right orthogonal matrices and $\mathbf{S}(m \times n)$ is a rectangular matrix with non-negative singular values on the diagonal in order of decreasing magnitude. The number of singular values is *r*, where *r* is the rank of **A**. The mathematical idea [11] is that the row space of **A** is *r*-dimensional and inside $\boldsymbol{R}^n$, and the column space of **A** is *r*-dimensional and inside $\boldsymbol{R}^m$. We choose special orthonormal bases $\mathbf{V} = (v_1, v_2, \ldots v_r)$ for the row space, and $\mathbf{U} = (u_1, u_2, \ldots u_r)$ for the column space, such that $\mathbf{A}v_i$ is in the direction of u_i, with s_i providing the scaling factor, i.e. $\mathbf{A}v_i = s_i u_i$. In matrix form, this becomes $\mathbf{AV} = \mathbf{US}$ or $\mathbf{A} = \mathbf{USV}^T$. **A** is thus the linear transformation that carries orthonormal basis v_i from space $\boldsymbol{R}^n$ to orthonormal basis u_i in space $\boldsymbol{R}^m$ and are related to each other by the magnitudes of the singular values. In any occurrence matrix **A**, each row vector represents which event *i* occurs in *n* episodes, and each column vector represents which of *m* events occur in episode *j*. SVD diagonalizes **A** into orthogonal bases **U** and **V,** where the independent *left singular vectors* in **U** represent an abstract

"event" space, and the independent *right singular vectors* in $\mathbf{V}^T$ represent an abstract "episode" space. That is, the i^{th} event and j^{th} episode are now described respectively by the i^{th} row of **U** and j^{th} column of **V** with *r* components in an *r*-dimensional space. The singular values capture the dominant associative relationships in the original matrix **A**. Thus, **US** and $\mathbf{SV}^T$ describe a scaled event and scaled episode space. The linear combinations (in terms of singular values) of the vectors in **U** and **V** describe the variables, parameters and functions uniquely as linearly independent vectors in an *r*-dimensional space. This description takes into account how each of them is "implicitly" related to all the others whether or not they were explicitly related in **A**. Each variable/parameter or function vector can now be "plotted" in an *r*-dimensional space as a single point. **US** and $\mathbf{SV}^T$ are special linear combinations because the s_i capture the dominant patterns in the data in decreasing order of magnitude. SVD takes the local explicit occurrence relationships and produces what could be described as a linear "global" association map. Thus, if only one entry in the occurrence 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.

$A = USV^T$

$U =$

	1	2	3	4	5	6	7	8
1	0.53249	-0.41233	-0.04704	0.45378	0.22925	-0.09811	0.49052	-0.18847
2	0.60716	0.36751	-0.13496	0.028622	-0.0011342	-0.033455	-0.55389	-0.41151
3	0.18222	-0.085294	-0.5601	-0.26286	0.52379	-0.042052	-0.12848	0.53299
4	0.22697	-0.61822	0.12803	-0.075087	-0.49115	-0.25317	-0.39455	0.28856
5	0.45336	0.39279	0.1079	-0.057774	-0.39971	0.32276	0.34286	0.49367
6	0.062049	-0.22357	-0.40929	-0.62462	-0.28757	0.19517	0.29284	-0.42656
7	0.14283	-0.27888	0.5105	-0.24847	0.37596	0.63716	-0.17462	-0.04083
8	0.18247	0.15299	0.45793	-0.51313	0.21922	-0.6125	0.21592	-0.021455

$S =$

	1	2	3	4	5	6	7	8	9
1	2.9317	0	0	0	0	0	0	0	0
2	0	2.2686	0	0	0	0	0	0	0
3	0	0	1.7481	0	0	0	0	0	0
4	0	0	0	1.5941	0	0	0	0	0
5	0	0	0	0	1.3735	0	0	0	0
6	0	0	0	0	0	0.69813	0	0	0
7	0	0	0	0	0	0	0.46254	0	0
8	0	0	0	0	0	0	0	0.27204	0

$V =$

	1	2	3	4	5	6	7	8	9
1	0.11096	-0.055492	0.55401	-0.47777	0.43333	0.03533	0.089292	-0.22895	0.44721
2	0.30778	-0.57719	0.33837	0.081689	0.083045	0.40949	-0.17006	0.21785	-0.44721
3	0.42399	0.40257	0.24649	-0.34019	-0.13223	-0.46293	0.01058	0.22312	-0.44721
4	0.54338	0.15338	-0.042392	0.26638	-0.12493	0.27367	0.60425	-0.39081	-2.498e-016
5	0.45089	-0.057353	-0.42453	0.13772	0.54744	-0.24869	-0.41478	-0.24625	-1.3878e-0...
6	0.098586	-0.37106	-0.1609	-0.43895	-0.56696	-0.083072	-0.2199	-0.50728	-2.501e-016
7	0.25906	-0.45426	0.046329	0.23756	-0.19068	-0.50317	0.20747	0.36793	0.44721
8	0.36175	0.33514	-0.015482	-0.018288	-0.29184	0.4144	-0.45624	0.30199	0.44721
9	0.083321	-0.13615	-0.55455	-0.55674	0.17198	0.21933	0.35535	0.39124	2.3577e-0...

Figure 2: SVD performed on the occurrence matrix A

Step III: Dimensionality reduction

*The third step is to perform a dimensionality reduction on the decomposed matrices, **U**, **S** and **V**, to produce a linear least squares truncated approximation of **A**.* From the **U**, **S** and **V** matrices in Figure 2, if we retain the first largest *k* singular values in **S** and compute a truncated approximation of **A** as $\mathbf{A'} = \mathbf{U}(m\times k)*\mathbf{S}(k\times k)*\mathbf{V}^T(k\times n)$, it will be an optimal *k*-rank least squares approximation of **A**. Figure 3 shows the k=2 approximation for the example problem.

Any linear combination that is an approximation of the original **US** and $\mathbf{SV}^T$ spaces is a valid approximation to the occurrence relationships in the original design problem representation because the orthonormal vectors and singular values are produced by using that very space. A dimensionality reduction implies that, instead of using *r* dimensions or linear combinations of abstract vectors to describe a variable, parameter or function, a lower number *k* is used as a least squares approximation of the associative relationships between variables, parameters or functions. The decomposition can be viewed in terms of *r* rank one matrices [11]. That is, the best rank 1 approximation to **A** is the matrix $\mathbf{u}_1 s_1 \mathbf{v}_1^T$, the best rank 2 approximation is $\mathbf{u}_1 s_1 \mathbf{v}_1^T + \mathbf{u}_2 s_2 \mathbf{v}_2^T$, and at *r* is $\sum_{i=1}^{r} \mathbf{u}_i s_i \mathbf{v}_i^T$.

$U_{[8 \times 2]} * S_{[2 \times 2]} * V_{[2 \times 9]} = A'$

$U_{[8 \times 2]}$

	1	2
1	0.53249	-0.41233
2	0.60716	0.36751
3	0.18222	-0.085294
4	0.22697	-0.61822
5	0.45336	0.39279
6	0.062049	-0.22357
7	0.14283	-0.27888
8	0.18247	0.15299

$S_{[2 \times 2]}$

	1	2
1	2.9317	0
2	0	2.2686

$V_{[2 \times 9]}$

	1	2
1	0.11096	-0.055492
2	0.30778	-0.57719
3	0.42399	0.40257
4	0.54338	0.15338
5	0.45089	-0.057353
6	0.098586	-0.37106
7	0.25906	-0.45426
8	0.36175	0.33514
9	0.083321	-0.13615

$= A'$

	1	2	3	4	5	6	7	8	9
1	0.22513	1.0204	0.28531	0.70479	0.75754	0.501	0.82934	0.25123	0.25743
2	0.15124	0.066613	1.0903	1.0951	0.75477	-0.13388	0.082384	0.92333	0.034799
3	0.070015	0.2761	0.1486	0.2606	0.25197	0.12447	0.22629	0.1284	0.070856
4	0.15166	1.0143	-0.28249	0.14645	0.38047	0.58601	0.80948	-0.22932	0.24639
5	0.098031	-0.10526	0.92226	0.85889	0.54818	-0.19962	-0.060476	0.77944	-0.010578
6	0.04833	0.34874	-0.12706	0.021047	0.11111	0.20614	0.27753	-0.10418	0.084211
7	0.081572	0.49406	-0.077164	0.13049	0.22509	0.27604	0.39588	-0.060559	0.12103
8	0.040098	-0.035689	0.36654	0.34392	0.2213	-0.076049	-0.019085	0.30983	-0.0026822

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

The approximations capture implicit associations through induction. The *k*-approximation uses a lower number of singular values, that is, a lower dimensional approximation, to induce a "best" guess on whether variable *i* is related to function *j*. Since the approximations based on retaining only the *k* largest singular values will overplay the most important associative patterns and underplay the rarer ones, variables and functions that mutually share high interaction/coupling relations in **A** will be strengthened and those that do not will be weakened. This means that if two events or episodes co-occur strongly with each other or with other common events and episodes in the original formulation, then they would tend to be placed close to each other in this re-represented approximation space. For example, if variable *i* occurs with other variables and in other constraints that have a high mutual occurrence relationship with constraint *j*, then the approximation will have to show a higher than 0 relationship between a variable *i* and a constraint *j* where the original occurrence matrix relationship showed a 0 to retain this global associative pattern. As the *k* values are further increased towards *r*, the approximations **A′** will be increasingly accurate approximations to the original occurrence matrix **A**. At *k=r,* the matrix will become the same again, **A′ = A.**

Thus, values in the $\mathbf{A}'$ approximations approach the limit set by the values in **A** as the k values are increased. However, it is the higher error approximations of **A** as $k \rightarrow 1$ that are potentially useful in finding alternative problem formulations. Rather than being errors *per se*, one can regard the increasing errors as increasing modeling freedom through a process akin to constraint relaxation.

For example, consider from Figure 1 that the variables x_1 and x_4 appear in function h_1, and x_2 and x_5 appear in functions h_2, but x_1, x_4, x_2 and x_5 do not appear directly in objective function *f*. It is intuitive and obvious 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" must be taught to an automated system as an explicit algebraic substitution rule. In the original matrix **A** (Figure 1), entry A_{11} (relationship between x_1 and *f*) is 0, and entry A_{12} (relationship between x_1 and h_1) is 1. These are explicit syntactic relationships. However, it is quite obvious that x_1 and *f* are correlated in an implied way because h_1 appears in *f*. Observing the same two entries in the 2D-truncated $\mathbf{A}'$ (Figure 3) shows that now A'_{11} is 0.22513 and A'_{12} is 1.0204, showing that, in this case, the 0 relationship has scaled up to 0.22 and the 1 relationship has become stronger than 1. Absolute 0-1 relationships have scaled up between the three quantities due to mutual association. The dimensionality reduction step says that using only 2 dimensions to induce a best guess reveals that x_1 appears in *f* 0.22 times, and x_1 appears in *h1* 1.0204 times. Similarly, note from Figure 1 that entry A_{43} (relationship between x_4 and h_2) is 0. Note from Figure 3 that entry A'_{43} is -0.2825. At k=2 the best guess is that x_4 appears in *h2* -0.2825 times – a 0 relationship is changed to an even lower value. In general, note from Figure 3 how all the 1s and 0s have been scaled higher or lower in terms of scaled up or scaled down mutual relations between the entries. The 0 or 1 entries in **A** that signified no relationship or a perfect relationship in a single equation are now changed to show a higher or lower (positive or negative) relationship, depending on occurrence relationships shared by one element with all the others. Table 1 shows, for the example problem, how the *k*-reduced approximations approach the original matrix **A** as the k values are increased. The first row "OM" shows the explicit Occurrence Matrix entries that x_1 shares with the objective function *f* and the functions h_1 to h_8. The other rows show the approximations computed at different k values. Note that as the k values are increased, the approximations increasingly approach the limit set by the k=8, i.e. full rank. At full-rank, only the explicit information, as specified in the occurrence matrix, is returned. The decomposition of the original formulation by SVD exposes alternative "implied" models.

Query: x_1	***f***	***h1***	***h2***	***h3***	***h4***	***h5***	***h6***	***h7***	***h8***
OM	0	1	0	1	1	0	1	0	0
k=2	0.23	1.02	0.29	0.70	0.76	0.50	0.83	0.25	0.26

k=3	0.18	0.99	0.27	0.71	0.79	0.51	0.83	0.25	0.30
k=4	-0.17	1.05	0.02	0.90	0.89	0.20	1.00	0.24	-0.10
k=5	-0.03	1.08	-0.02	0.86	1.06	0.02	0.94	0.15	-0.05
k=6	-0.03	1.05	0.01	0.84	1.08	0.02	0.97	0.12	-0.06
k=7	0.00	1.00	0.00	1.00	1.00	0.00	1.00	0.00	0.00
k=8	0.00	1.00	0.00	1.00	1.00	0.00	1.00	0.00	0.00

Table 1: The entries in A′ approach the limit set by the entries in A as the *k* values are increased; here, the rows show the matrix entries from the approximations for variable x_1 with all the other functions

Reformulation tasks such as problem decomposition can be performed based on this unique mathematical artifact of SVD as a matrix decomposition technique. Since the components of the **US** and $\mathbf{SV}^T$ spaces are vectors in an original r or reduced k dimension space, this induced interaction strength relationship is, in effect, converted into a distance relationship. Any points lying close in this re-represented k-dimensional space are semantically similar and share high interaction/coupling relationships; any points lying far are dissimilar and share low interaction/coupling relationships. At some k values, this step minimizes the distance between strongly connected variables and functions that appear closer in this re-representation space due to the k largest singular values that capture the strongest relations. At the same time, it maximizes the distance between the less strongly connected variables and functions that appear far in space, again due to disregarding the (r-k) smallest singular values and treating the weaker association patterns as "noise".

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 geometric "meaning" of the method. When k=2, this implies, that 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, to get coordinates, say, $\{o_{i1}, o_{i2}\}$ for the m variables or parameters for representing in a 2D plane, i=1 to m. Similarly, if s_1 and s_2 are multiplied respectively with the first and second columns of **V**, we will get coordinates, say (e_{1j}, e_{2j}) for the n functions for representing in a 2D plane, j=1 to n. This 2D representation gives a visual idea of how the variables, parameters and objective and constraint functions are related to each other in the reduced k=2 space. In 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. Each of the events (x_1 to x_6) and episodes (h_1 to h_8) are plotted as points in this 2D space. Note how the episodes that share common events fall close to each other, while those that do not fall distant from each other.

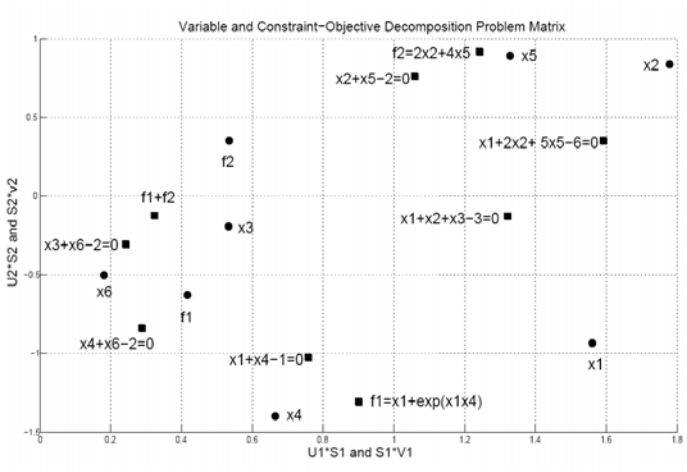


Figure 4: Distributed graph representation for design elements and functions in 2D space

The clustering of events and episodes in the *k* dimensional space provides a way to identify alternative design models implied by the original formulation. One way to check the validity of these alternatives is to assess them against criteria normally associated with problem reformulation including well-formed design "cases", modular sub-problems, or constraint satisfaction among others. The next section describes unsupervised similarity measurements performed as cosine measurements between events and episodes in these k-reduced spaces.

Step IV: Inferring design (re)formulation semantics by unsupervised similarity measurements

The final step involves 'querying' the k-approximations to infer the interaction strengths between events and episodes that becomes the basis for problem reformulation. The position and distance between vectors representing events and episodes in the *k*-reduced space is a measure of the interaction/coupling strength between them. For example, if coupling is defined as the cosine of the angle between them, then two vectors that have a high cosine value share a strong interaction with each other. This mathematical property is useful for reformulation tasks in which a problem needs to be seen in terms of having interacting/coupled sub-structures. Events that share semantic (structural or behavioral) relationships will tend to appear together in same context or episode (functions). This method will convert these local relationships into a continuous representation in space wherein the higher the cosine between two events or episodes, the higher the implied interaction between them. Thus, unsupervised clustering in this space will reveal these sub-structures with high mutual interactions. In this work, we choose the K-means clustering algorithm in the *k*-reduced space using a cosine similarity measure to infer "semantically related groups" of variables, parameters and functions.

Inferring selection of variables, parameters, objectives and constraints using cosine similarity

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, design variables and parameters are often semantically related to each other in statements such as "the wall thickness of the hydraulic cylinder should not be less than the minimum possible thickness from the casting process". As a consequence, the decision hinges upon knowing which set of other variables or parameters become important in conjunction with a particular variable being considered. These groups of related variables and parameters may become design cases or they may suggest that if a variable is 'isolated' from a previously associated parameter, the variable may be treated as an independent parameter.

Consider Figure 4 for the 2D case, where all the points represent either variables or functions. A cosine calculation between any of these two points would reveal their semantic relationship in this space, i.e. how "close" or "distant" they 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. The cosine between x_2 and x_4, calculated as their dot product divided by magnitude, gives $\cos(\theta_{x2\text{-}x4}) = (\boldsymbol{x_2} \,.\, \boldsymbol{x_4}) \,/\, ||\boldsymbol{x_2}||\, ||\boldsymbol{x_4}||$. The following table (Table 2) shows how the k-reduced representation can be used to induce this relationship: x_2 and x_4 have a very low similarity of 0.0050, while x_2 and x_5 show a high similarity of 0.9884 in the k=2 approximation space.

Variable	x-coordinate	y-coordinate	cosine with x_2
x_2	1.780	0.834	1
x_4	0.665	-1.402	0.005
x_5	1.329	0.891	0.988

Table 2: Cosine similarity values used to find variable groupings

If the designer inputs any variable as a query, its cosine similarity with each of the other variables and parameters may be calculated. A cosine similarity threshold may be decided upon, and all variables higher than a certain threshold level can be returned as sharing an association with the query variable. The cosine similarity 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 $h_7\}$.

A similar query is possible for finding relations of variables and parameters to objectives and constraints, i.e., 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, Figure 5 shows the variables and functions that are returned with x_1 as the query variable and 0.7 as the cosine threshold.

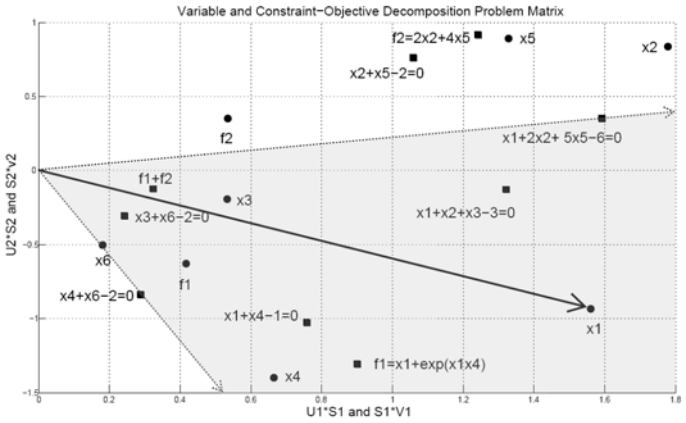


Figure 5: Variables and functions returned with query variable x_1 and cosine similarity threshold of 0.7

Table 3 shows the cosines and the calculated grouping $\{f, h_1, h_3, h_4, h_5, h_6, h_8\}$ for query variable x_1 at various values of k. The value of $k=2$ results in the best reformulation. In this case, note that even though x_1 does not occur directly in h_5 and h_8, but because x_1 has high correlations between both $x4$ and $x3$, which do occur in h_5 and h_8 respectively, these implicit relationships are retrieved. No reformulation is possible using only the original occurrence matrix entries. For example, at k=8, no implicit information is returned and no useful case is identified.

Query: x_1	f	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8
OM	0	1	0	1	1	0	1	0	0
k=2	0.99	0.91	0.39	0.73	0.90	0.76	0.92	0.40	0.94
k=3	0.29	0.84	0.36	0.73	0.81	0.74	0.92	0.40	0.39
k=4	0.00	0.80	0.20	0.75	0.80	0.33	0.92	0.36	0.04
k=5	0.07	0.80	0.18	0.72	0.78	0.18	0.86	0.29	0.07
k=6	0.06	0.78	0.18	0.71	0.78	0.18	0.84	0.28	0.06
k=7	0.07	0.77	0.18	0.71	0.75	0.17	0.84	0.26	0.07
k=8	0.07	0.77	0.18	0.71	0.75	0.18	0.84	0.26	0.07

Table 3: Cosine between variable x_1 and functions

Inferring model-based decomposition using K-means clustering

Many optimization problem models can be decomposed into independent sub-problems characterized by local variables and constraints. It is useful to identify these sub-problems because they can be solved independently. Decomposing large optimization problems into smaller ones is computationally advantageous because large model sizes reduce the reliability and speed of numerical solution algorithms [12]. Dissociable relations between variables leading to decomposition into independent sub-problems are one type of interacting/coupled sub-structures that can

be identified using this method. The example problem in Figure 1 is formulated and solved by Michelena and Papalambros [12] as a model-based decomposition problem. They solve it using hypergraph partitioning. The problem is optimally partitioned into weakly connected sub-graphs.

Cosine measurements were performed in the previous section to observe the implied interaction strengths that one variable may share with all other variables and functions. However, if a simultaneous clustering on this space were performed to identify clusters of highly related variables and functions, this would correspond to a general decomposition definition: the identification of weakly interacting sub-systems with strong local interactions. Here, we show that performing a K-means clustering on the *k*-reduced representation can successfully identify these independent sub-problems, with local variables and constraints and the linking variables and constraints. 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 [13]. The data points are vectors denoted by $\mathbf{x}^{(n)}$, where n=1 to *N*, each **x** has *I* components x_i. Each data point is assigned to the nearest mean (or centroid). A metric is defined for measuring distances between these data points (e.g. cosine, Euclidean, city-block, etc.). For this work, we have used a cosine distance between data points. In an update step, the means are adjusted to match the sample means of the data points within a cluster. This algorithm always converges to a set of clusters, but initializing the means at different points can produce different clusters for the same data set. In the dimensionality reduction step, at some *k* values the approximation maximizes the intra-sub-problem interaction and minimizes inter-sub-problem interaction by making the strongly connected variables and functions appear closer in space to each other, and making the weakly connected ones appear far in space. Performing a K-means clustering on this space will therefore identify these groups of strongly interacting variables and functions as clusters. Each of these clusters will then be a sub-problem of the original problem.

Figure 6 shows the results of applying the K-means algorithm onto the example problem. The results almost exactly match with the results by Michelena and Papalambros [12] – subproblem 1 with local variables $\{x_4, x_6\}$ and functions $\{h_1, h_5, h_6, h_8\}$, subproblem 2 with local variables $\{x_2, x_5\}$ and functions $\{h_2, h_3, h_4, h_7\}$ and $\{x_1, x_3\}$ as linking variables. The only difference is that h_4 is part of the cluster with the linking variables in our results.

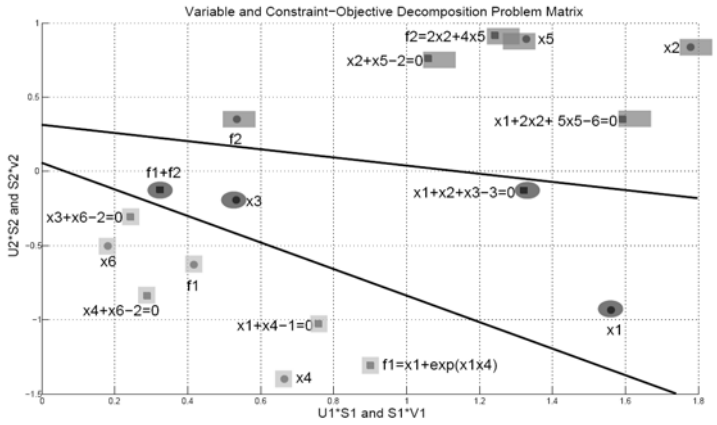


Figure 6: Model-based decomposition using SVD algorithm and K-means clustering

Heuristics for choosing the *k* values

Recall that as the *k* values increase, the approximations approach the limits set by the entries of the original occurrence matrix. That is, there is a continuous gradation of interaction strengths between the most "implicit" end (largest *k* values) and the most "explicit" end (the full rank *k*=*r* approximation). Thus, as the *k* value increases, the information returned by the approximated matrix comes closer and closer to the explicit information contained in the occurrence matrix. The first few singular values *k*=2 to *a* will return implicit information. As the *k* value increases beyond *a*, then all entries in the approximated *k*-reduced matrices would approach the limit values set by the explicit occurrence information in the occurrence matrix. This will cause the cosine values to measure associative distances between events and episodes in the *k*-reduced **US** and $\mathbf{SV}^T$ spaces as approaching those in the *k*=*r* full rank **US** and $\mathbf{SV}^T$ spaces. As shown previously, if the cosine threshold parameter is fixed for each *k*, at some *k*=*a*, and $k \leq r$, the approximation will start to return only the explicit association information as contained in the occurrence matrix. Therefore, as a first rule, *for observing implicit associative information that leads to reformulation decisions, use k=2 to a.* Beyond *a*, the approximation will return only explicit associative information, as can be read off directly from the occurrence matrix, or cosine measurements between **UV** and $\mathbf{SV}^T$ vectors in *k*=*r* space. Therefore, *the search for well-formed reformulations is limited within this range of k*=2 *to a.* As a rule, out of all the 2 to *a* approximations, *a good value of k is one that would help to correctly alter a not well-formulated problem into well-formulated ones.* A heuristic method to identify a well-formulated problem is as follows:

1. Apply the method for *k*=2 to *r*. Identify the value *a* beyond which no implicit information is returned through a parametric study that increases the *k* by steps.

2. Fix a cosine threshold or use the K-means clustering algorithm on the **US** and $\mathbf{SV}^T$ approximations in the *k*-reduced spaces from 2 to *a* to study the reformulations returned for each of the *k* values. If there is a well-formed reformulation, the method will return this.

EXPERIMENTS AND RESULTS

We have applied the method on three archetypal design problems – hydraulic cylinder design (HCD) (a single and multi-objective optimization problem that is poorly formed), non-analytically formulated optimization problems and Aircraft Concept Sizing (ACS) (a medium-to-large scale collaborative optimization problem). We present the experiments and results in this section.

The Hydraulic Cylinder Design (HCD) Problem for heuristic design "case" identification

Using the computations shown in the previous section, the method can provide heuristic insights into design "case" identification [14]. In any problem formulation, the mathematical model contains a set of constraints. This model could be over or under-constrained, i.e. not well-formulated. "Design cases" are sets of active or critical constraints for a design optimization problem formulation which lead to a well-formulated model. One method for identifying these design "cases" [14] is monotonicity analysis [14]. Monotonicity analysis is a problem-solving-by-reformulation method, where constraint activity information is used to reformulate the problem to a simpler form when a problem is over or under constrained. A significant characteristic to note here is that this process of reformulation is actually similar to discovering previously unobserved implicit relationships between variables, parameters and constraints, that when made explicit, make solving the problem possible.

The 4 steps outlined in the previous section can aid the identification of design "cases" in a heuristic manner using a similar conceptual idea – *the inference of previously unobserved implicit but strong relationships between variables, parameters and constraints*. Note that monotonicity analysis is a mathematically rigorous rule-based procedure and it can identify these cases without ambiguity based on formal definitions of criticality, activity, monotonicity and optimality. This method can provide heuristic insights into design case groups in a "static" way, but cannot identify bounded cases as monotonicity analysis does because it employs an unsupervised pattern recognition approach, as opposed to an approach grounded in formal conditions for optimality. However, monotonicity analysis is limited to problems in which regions of monotonic behavior in constraints may be identified. It quickly turns into a very complex solution procedure for even a small-medium sized problem as the number of variables and constraints

increase. The method presented here would be particularly suited for complex problems, where, if used in conjunction with monotonicity analysis, will be able to focus a designer's attention on possible design cases through as associative variables and functions. An added advantage of this method is that it can be used to identify semantically related groups for analytic as well as non-analytic formulations, whether or not functional relationships are available, whether or not the functions are differentiable. This implies that for design problems where monotonicity is not present or the functions are not differentiable, this method can still be used to provide the designer with some insight into how design "cases" may emerge by focusing the designer's attention onto groups of variables and constraints within the main problem.

The HCD problem has been formulated and solved in multiple ways. Here, we consider two formulations of the problem – a single objective formulation [14] and a multi objective formulation [15]. Figure 7(a) shows the design problem and Figure 7(b) shows the two problem statements.

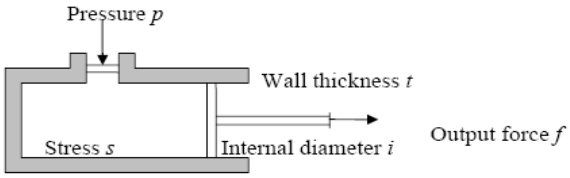


Figure 7(a): The Hydraulic Cylinder Design Problem

Single-objective formulation:

Design variables:	**Design model:**
Internal diameter i	Min $i + 2t$
Wall thickness t	$g1: t - T \geq 0$
Output force f	$g2: f - F \geq 0$
Stress s	$g3: p - P \leq 0$
Pressure p	$g4: s - S \leq 0$
	$h1: f = (\pi/4)i^2p$
	$h2: s = ip/2t$

Design parameters:
Min Wall thickness T
Min Output force F
Max Stress S
Max Pressure P

Multi-objective formulation:

Design variables:	**Design model:**
Internal diameter i	Min $Z = z1w1 + z2w2 + z3w3$
Wall thickness t	$z1 = \pi\,(it + t^2)$
Output force f	$z2 = s/S$
Stress s	$z3 = p/P$
Pressure p	$t - T \geq 0$
Objective Z	$f - F \geq 0$
Objective $z1$	$p - P \leq 0$
Objective $z2$	$s - S \leq 0$
Objective $z3$	$f = (\pi/4)i^2p$
Weight $w1$	$s = p/E\,(i/2t + 0.6)$
Weight $w2$	
Weight $w3$	

Design parameters:
Min Wall thickness T
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 Problem

The method was applied onto these two problem models. Figure 8 and Figure 9 show the 2D graph (k = 2) and 3D graph (k=3) produced for the single-objective and multi-objective cases, respectively, by performing the 4 steps. The performance of the algorithm and the validity of the results were measured by comparing the results returned in each case (groups of variables and constraints) to those from the source papers. The best dimension was selected using the heuristics presented in the previous section, and then compared to the results presented in the source papers.

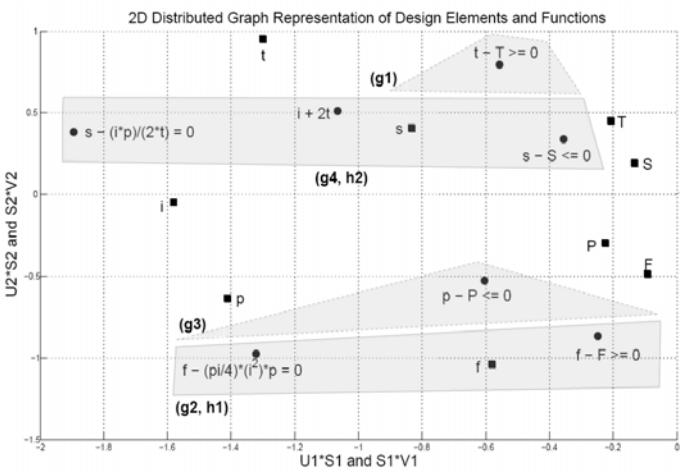


Figure 8: 2D graph of the single objective HCD

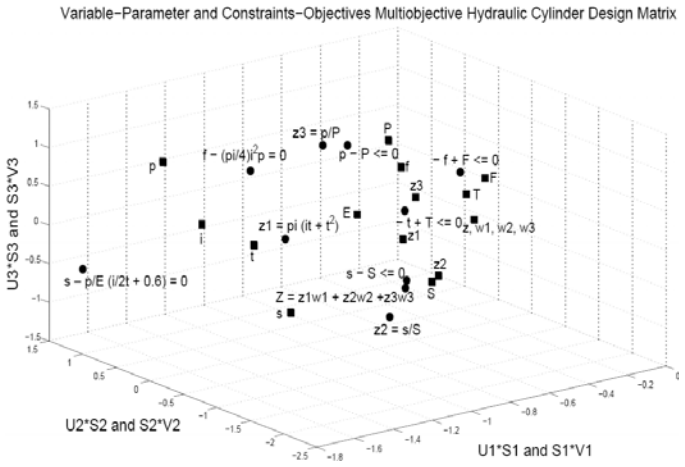
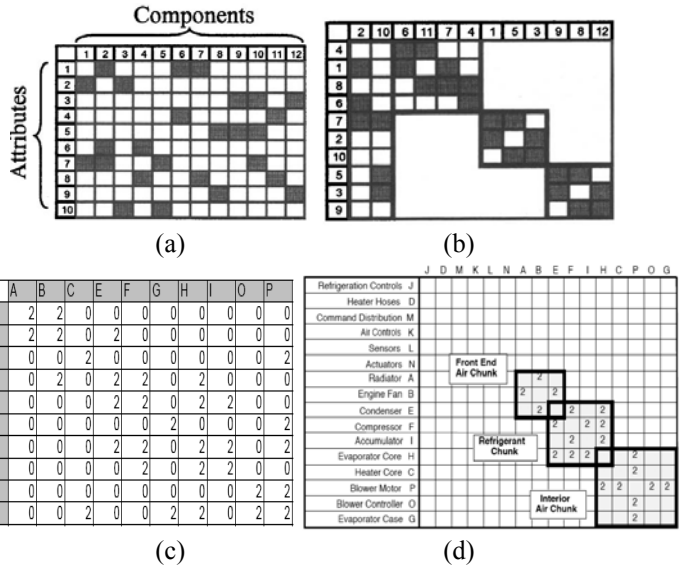


Figure 9: 3D graph of the multi-objective HCD

The groupings provided by the method in both cases matches with the design "cases" identified in the source papers. In the single objective case, the authors [14] report these cases as pressure-bound, stress bound and thickness bound. Their 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 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 [15] 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 9 shows that pressure and stress groups are distinctly apart, with force and thickness falling in the middle.

Design Decomposition for the Incidence Matrix (IM) and Design Structure Matrix (DSM) forms

In this section, we demonstrate that the method performs problem decomposition for any design problem that is stated in a FDT form [16] or design structure matrix (DSM) form [17]. The FDT form is usually a rectangular matrix where the rows represent functions or attributes and the columns represent variables or components. This is the transpose of the form we have used in the paper, but it makes no difference to the results. The matrix entries capture whether there is a relationship between a particular variable and function, Figure 10(a). The DSM matrix form Figure 10 (c) is a square matrix that captures mappings between the elements.



Material	A	B	C	E	F	G	H	I	O	P
A	2	2	0	0	0	0	0	0	0	0
B	2	2	0	2	0	0	0	0	0	0
C	0	0	2	0	0	0	0	0	0	2
E	0	2	0	2	2	0	2	0	0	0
F	0	0	0	2	2	0	2	2	0	0
G	0	0	0	0	0	2	0	0	0	2
H	0	0	0	2	2	0	2	2	0	2
I	0	0	0	0	2	0	2	2	0	0
O	0	0	0	0	0	0	0	0	2	2
P	0	0	2	0	0	2	2	0	2	2

Figure 10: Original incidence matrix problem and (b) its solution; (c) Original DSM and (d) its solution

Figure 10(a) shows a problem with 12 components and 10 attributes. Figure 10(b) shows the results of the decomposition as demonstrated by Li and Li [16] using their method. We follow the commonly accepted decomposition definition as proposed by Simon [18]: the decomposition strategy should lead to a "nearly decomposable system" characterized by weakly interacting subsystems with strong local interactions within each subsystem. In our method, this is interpreted as the mathematical observation that, at the best identified *k*-reduced approximation using the heuristics outlined earlier, all sub-systems with strong local interactions will show high mutual cosine measurements, while showing low cosine measurements with elements from other sub-systems. Figure 11 shows the decomposition based on cosine similarity measurements between the attributes and the components. The results of the K-means clustering have been superimposed on this matrix, that is, the rows and columns have been reorganized. The K-means clustering at k=2 identifies the same decomposition results as reported by Li and Li [16] shown in Figure 10(b). This clustering result corresponds to a cosine threshold to 0.9. We can see that all the

component-attribute pairs that show a mutual cosine measurement of higher than 0.9 can be identified as a sub-problem. There are three blocks that emerge as three sub-problems with the two remaining components as the interaction/coordination chunk. Note that Li and Li solve this problem as a decomposition followed by re-decomposition solution process, making it algorithmically more complicated. Our method finds the solution in one shot. In keeping with their complexity calculations, this solution is reported as the best decomposition that their method finds.

	2	10	6	11	7	4	1	5	3	9	8	12
4	0.85674	-0.078624	0.99486	0.99962	0.99654	0.99486	0.49456	0.49456	0.55605	-0.36219	-0.41707	-0.41707
1	0.95089	0.14794	0.99211	0.96772	0.98965	0.99211	0.67762	0.67762	0.72897	-0.14294	-0.20164	-0.20164
8	0.86461	-0.0632	0.9963	0.99907	0.99771	0.9963	0.50794	0.50794	0.56884	-0.34773	-0.40297	-0.40297
6	0.97498	0.23675	0.97671	0.94098	0.97262	0.97671	0.74133	0.74133	0.78787	-0.052872	-0.11227	-0.11227
7	0.85352	0.8475	0.54995	0.4379	0.53457	0.54995	0.99933	0.99933	0.99409	0.6589	0.61291	0.61291
2	0.89334	0.80116	0.61653	0.51009	0.60201	0.61653	0.99896	0.99896	0.99964	0.59505	0.54612	0.54612
10	0.89334	0.80116	0.61653	0.51009	0.60201	0.61653	0.99896	0.99896	0.99964	0.59505	0.54612	0.54612
5	0.22789	0.97292	-0.20894	-0.33305	-0.22681	-0.20894	0.67539	0.67539	0.62036	0.99828	0.99302	0.99302
3	0.22789	0.97292	-0.20894	-0.33305	-0.22681	-0.20894	0.67539	0.67539	0.62036	0.99828	0.99302	0.99302
9	0.051832	0.9166	-0.37884	-0.49478	-0.39572	-0.37884	0.5341	0.5341	0.47164	0.99288	0.99821	0.99821

Figure 11: Decomposition results based on cosine similarity for the incidence matrix

For the DSM demonstration, we use an automotive climate control system example from Pimmler and Eppinger [17]. They formulate four interaction types between components in their paper that place different constraint relationships between the components: spatial adjacency, material exchange, information exchange and energy exchange. Figure 10(c) shows the material interaction matrix, and Figure 10(d) shows the results as reported by them. Figure 12 shows the results from applying our method onto the DSM form with K-means clustering applied on a k=2 approximation space. Again, note the cosine threshold as 0.9.

	A	B	E	F	I	H	C	G	O	P
A	1									
B	0.972	1								
E	0.834	0.941	1							
F	0.783	0.908	0.997	1						
I	0.732	0.872	0.986	0.997	1					
H	0.535	0.719	0.912	0.945	0.967	1				
C	-0.568	-0.358	-0.020	0.067	0.145	0.391	1			
G	-0.568	-0.358	-0.020	0.067	0.145	0.391	1	1		
O	-0.568	-0.358	-0.020	0.067	0.145	0.391	1	1	1	
P	-0.387	-0.159	0.184	0.270	0.344	0.571	0.979	0.979	0.979	1

Figure 12: Decomposition results for the DSM form

The authors in the original paper do not describe the exact details of the clustering algorithm that they employ to reach their decomposition. If we vary our cosine threshold to 0.7 (or equivalently the number of clusters in K-means to 2), we obtain an alternative decomposition as shown by the large box around the two top sub-problems. Note that both solutions identified by the method are well formed. Solution 1 has three sub-problems of 3, 3 and 4 components with 1 linking component. Solution 2 has no linking components, but sub-problem sizes of 6 and 4 components each. A tradeoff between minimizing the number of linking components (1 or 0) and balancing the sub-problem sizes

(3,4,4 versus 6,4) exists. Solution acceptability will depend upon which one is more cost-efficient when actually assessed for any optimization or design objective. This "multiple result" outcome arises because our method works by measuring distributed patterns of associations between all components. In a lower dimensional space, it reveals implicit, indirect associations as well as explicit, direct ones. Even those components that do not directly co-occur with each other show high semantic relationship with each other on the ground that they are related through co-occurrence with a third component. This is an interesting result because it shows that designers can tune the cosine thresholds and number of clusters to observe multiple, well-formed decompositions. This is an advantage because it is frequently difficult to objectively select an "optimal" decomposition for complex design problems; sub-system boundaries can be decided on non-design related criteria such as organizational boundaries[19].

The Aircraft Concept Sizing (ACS) Domain

Applying the SVD method to the Aircraft Concept Sizing (ACS) problem was an experiment to empirically validate the method's performance on a medium-large sized problem. Given that SVD is known to scale well, 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 or where monotonicity analysis may be too complex. The ACS problem also served as a validation experiment – given a problem with a significant amount of coupling, for which the sub-problems are already decided by a designer, will the method be able to identify the "correct" cases or sub problems?

The source for the ACS problem is Gu et al. [20]. 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 13 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 reformulation, for which this experiment validates whether the method returns the sub-problem decomposition as described by Gu, when all the interaction information is merged and used as input into the method. The objective of the main problem

is to maximize 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^2)
x_3	Fuselage length (ft)
x_4	Fuselage diameter (ft)
x_5	Density of air cruise altitude ($slugs/ft^3$)
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^2)
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 13: The Aircraft Concept Sizing Problem

We applied the method 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 method. *Correct cases* are defined as those variables or design states that are part of the original sub-problem definition and should be returned by the method. *Missed cases* are those that are correct but not returned as an answer. As such, correct cases measure the success, and missed cases measure failure of the method to classify a variable or state into the "correct" sub-problem. The method should produce groupings of shared and independent variables, i.e., which variables and states are shared or occur independently in which sub-problems. Another related aim in this experiment was to study at what reduced dimensionality the method successfully returns semantic groupings as expressed in the problem statement. Note that the problem is characterized by each of the sub-problems sharing many design variables and system state variables. This implies that there will be many explicit as well as implicit relationships between the sub-problems because of these couplings.

The results show that REIFORM captures 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 method, especially at k=2, also returns some extra cases, that is, variables or system states that are not part of the original sub-problem, but show a high interaction relationship with the given sub-problem. We cannot classify these extra cases as "incorrect", because, from previous experiments, we know that the method returns cases on the basis of relationships that both occur explicitly or latently in the problem syntax. Therefore, these "extra" cases show some implied relationships that are not explicitly in the formulation. As future work, these cases could be run through a numerical optimizer to determine whether the alternate formulation is less costly to optimize.

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=*r, r-1, r-2,…,* 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 if there is an optimal error. For example, we calculate the estimation error for cases k=2, 3, 4 and 5, by using infinity norm, 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 14 shows the details of the correct, extra and missed variables and states returned by the method for k = 2, 3, 4 with a cosine threshold of 0.7, and the queries set to the 5 sub-problems.

We used two metrics to measure the performance of the method – 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 15 shows the improvement in performance for k=2 (i.e., Precision_{2D} and Recall_{2D}) 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. Even at k=2 or k=3, the answers returned by the method 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
	Original	$[x_1, x_2, x_3, x_4, x_9, x_{10}, x_{15}]$	$[x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_{11}, x_{12}]$	$[x_2, x_7, x_{10}, x_{12}, x_{13}, x_{14}]$	$[x_2, x_3, x_4, x_6, x_7, x_{16}, x_{17}]$	$[x_6, x_8, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{17}, x_{18}]$
2D, k = 2, cosine threshold = 0.7						
	Exact match	$[x_1, x_2, x_3, x_4, x_9, x_{10}]$	$[x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_{11}]$	$[x_2, x_7, x_{10}, x_{12}, x_{13}, x_{14}]$	$[x_2, x_3, x_4, x_6, x_7]$	$[x_6, x_8, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{17}, x_{18}]$
	Extra	$[x_5, x_7, x_{11}]$	$[x_9, x_{10}]$	$[x_6, x_8, x_{15}, x_{16}, x_{17}, x_{18}]$	$[x_1, x_5, x_9, x_{10}, x_{11}, x_{12}, x_{15}]$	None
	Missed	$[x_{15}]$	$[x_{12}]$	None	$[x_{16}, x_{17}]$	None
3D, k = 3, cosine threshold = 0.7						
	Exact match	$[x_1, x_2, x_3, x_4, x_9, x_{10}]$	$[x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_{11}]$	$[x_2, x_{10}, x_{12}, x_{13}, x_{14}]$	$[x_2, x_3, x_4, x_6, x_7, x_{16}, x_{17}]$	$[x_6, x_8, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{17}, x_{18}]$
	Extra	$[x_5, x_7, x_{11}]$	None	$[x_{15}]$	$[x_5, x_{11}]$	None
	Missed	$[x_{15}]$	$[x_{12}]$	$[x_7]$	None	None
4D, k = 4, cosine threshold = 0.7						
	Exact match	$[x_1, x_2, x_3, x_4, x_9, x_{10}, x_{15}]$	$[x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_{11}]$	$[x_7, x_{10}, x_{12}, x_{13}, x_{14}]$	$[x_2, x_3, x_4, x_6, x_7, x_{16}, x_{17}]$	$[x_6, x_8, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{17}, x_{18}]$
	Extra	None	None	None	$[x_5, x_{11}]$	None
	Missed	None	$[x_{12}]$	$[x_2]$	None	None

Figure 14: Correct, extra and missed cases for dimensions *k*=2, 3 and 4

	SP1	SP2	SP3	SP4	SP5	Avg
$Precision_{2D}$	6/9	8/10	6/12	5/12	9/9	0.67
$Precision_{3D}$	6/9	8/8	5/6	7/9	9/9	0.9
$Precision_{4D}$	7/7	8/8	5/5	7/9	9/9	0.95

(a)

	SP1	SP2	SP3	SP4	SP5	Avg
$Precision_{2D}$	6/7	8/9	6/6	5/7	9/9	0.89
$Precision_{3D}$	6/7	8/9	5/6	7/7	9/9	0.92
$Precision_{4D}$	7/7	8/9	5/6	7/7	9/9	0.95

(b)

Figure 15: (a) precision and (b) recall measurements as ratio of identified to expected

CONCLUSIONS

We developed, demonstrated and tested an SVD-based method for symbolic design optimization problem reformulation on various problem domains and representations. The method proceeds in 4 steps: 1) form an occurrence matrix capturing the 'known' relations between variables and parameters in the row-space and the functions in the column-space; 2) calculate SVD on this matrix; 3) calculate a *k*-reduced approximation of the original occurrence matrix; 4) use cosine measurement or K-means clustering to find relations between variables, parameters and functions to group them to identify sub-problems.

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 capture implied functional-behavioral relationships through the syntactic structure of the formulation. This basic pattern extraction view reduces the need for domain specific knowledge

engineering for specific optimization tasks, or the need for large database of solved examples to train the system specific optimization problem formulation strategies.

As this method operates solely on measurements of co-occurrences of variables and constraints in context of each other, the method is applicable to models where explicit analytical functional relationships are not available (e.g., non-analytical formulations, numerical analysis or bespoke 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 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 order of magnitude issues.

The method is computationally very efficient and simple to implement in numerical programming packages such as MATLAB®. Because numerical implementations of SVD are known to scale well, this method may be applied to a range of problems from various domains, from small scale to very large scale, as long as the optimization problem is formulated in the occurrence matrix form. We believe that potential application areas for this method are medium, large and very large scale problems, where the number of design variables, parameters and functions become too large for choices to be determined by simple reasoning or direct human observation of design relationships.

REFERENCES

[1] 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), pp. 20-34.

[2] Schwabacher, M., Ellman, T., and Hirsh, H., 1998, "Learning to Set up Numerical Optimizations of Engineering Designs," AI EDAM, 12(2), pp. 173-192.

[3] 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), pp. 30-61.

[4] Gelsey, A., Schwabacher, M., and Smith, D., 1998, "Using Modeling Knowledge to Guide Design Space Search," Artificial Intelligence, 101(1), pp. 35-62.

[5] Campbell, M. I., Cagan, J., and Kotovsky, K., 2003, "The a-Design Approach to Managing Automated Design Synthesis," Research in Engineering Design, 14(1), pp. 12-24.

[6] Medland, A. J., and Mullineux, G., 2000, "A Decomposition Strategy for Conceptual Design," Journal of Engineering Design, 11(1), pp. 3-16.

[7] Duffy, A. H. B., and Kerr, S. M., 1993, "Customised Perspectives of Past Designs from Automated Group Rationalisations," Artificial Intelligence in Engineering, 8(pp. 183-200.

[8] Dong, A., 2005, "The Latent Semantic Approach to Studying Design Team Communication," Design Studies, 26(5), pp. 445-461.

[9] 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), pp. 211-240.

[10] Kalman, D., 1996, "A Singularly Valuable Decomposition: The Svd of a Matrix," The College Mathematics Journal, 27(1), pp. 2-23.

[11] Strang, G., 2003, *Introduction to Linear Algebra*, Wellesley-Cambridge Press, Wellesley.

[12] 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), pp. 173-196.

[13] Mackay, D. J. C., 2003, *Information Theory, Inference and Learning Algorithms*, Cambridge University Press,

[14] Papalambros, P. Y., and Wilde, D. J., 2000, *Principles of Optimal Design*, Cambridge University Press,

[15] Michelena, N. F., and Agogino, A. M., 1988, "Multi-Objective Hydraulic Cylinder Design," Journal of Mechanisms, Transmissions and Automation in Design, 110(pp. 81-87.

[16] Li, C., and Li, S., 2005, "Analysis of Decomposability and Complexity for Design Problems in the Context of Decomposition," Journal of Mechanical Design, 127(4), pp. 545-557.

[17] Pimmler, T. U., and Eppinger, S. D., 1994, "Integration Analysis of Product Decompositions," eds., Minneapolis, pp.

[18] Simon, H. A., 1969/ 1981, *The Sciences of the Artificial*, MIT Press, Cambridge, MA.

[19] Sosa, M. E., Eppinger, S. D., and Rowles, C. M., 2003, "Identifying Modular and Integrative Systems and Their Impact on Design Team Interactions," Journal of Mechanical Design, 125(pp. 240-252.

[20] Gu, X., Renaud, J. E., Ashe, L. M., Batill, S. M., Budhiraja, A. M., and Krajewski, L. J., 2002, "Descision-Based Collaborative Optimization," Journal of Mechanical Design, 124(1), pp. 1-13.