Simulation Data Mining for Supporting Bridge Design

Steven Burrows[1] **Benno Stein**[1] **Jörg Frochte**[2]
David Wiesner[1] **Katja Müller**[1]

[1] Web Technology and Information Systems
Bauhaus-Universität Weimar
99421 Weimar, Germany
Email: <first>.<last>@uni-weimar.de

[2] Electrical Engineering and Computer Science
Bochum University of Applied Science
42579 Bochum, Germany
Email: joerg.frochte@hs-bochum.de

Abstract

We introduce simulation data mining as an approach to extract knowledge and decision rules from simulation results. The acquired knowledge can be utilized to provide preliminary answers and immediate feedback if a precise analysis is not at hand, or if waiting for the actual simulation results will considerably impair the interaction between a human designer and the computer. This paper reports on a bridge design project in civil engineering where the motivation to apply simulation data mining is twofold: (1) when dealing with real-world bridge models the simulation efficiency is inadequate to gain true interactivity during the design process, and (2) the designers are confronted with a parameter space (the design space) of enormous size, from which they can analyze only a small fraction. To address both issues, we propose that a database of models (the design variants) should be pre-computed so that the behavior of similar models can be used to guide decision making. In particular, simulation results based on displacement, strain, and stress analyses are clustered to identify models with similar behavior, which may not be obvious in the design space. By means of machine learning, the clustering results obtained in the simulation space can be transferred back into the design space in the form of a highly non-linear similarity measure that compares two design alternatives based on relevant physical connections. If the assessments of the measure are reliable, it will perfectly address the mentioned issues above. With this approach we break new ground, and our paper details the technology and its application for a real-world design setting.

Keywords: Engineering Design Support, Simulation Data Mining, Cluster Analysis, Similarity Measures.

1 Introduction

Simulation data mining combines the systems simulation and data mining fields to develop knowledge and intelligence from simulated data. Stein (2001) describes a diverse rationale for simulation data mining, which includes data generation when real-world sensor data is unavailable, the identification of heuristic shortcuts for complex analyses, the semi-automatic evaluation and ranking of design alternatives, and the identification of heuristic design rules. Simulation data mining has been successfully applied in different areas such as diagnosis of fluidic systems (Stein, 2003), automotive crash simulation (Painter et al., 2006), and aircraft engine maintenance (Mei and Thole, 2008). This paper deals with interactive bridge design and demonstrates the potential of simulation data mining for both improving interactivity and simplifying analyses (Burrows, 2011). Our research is part of a large, interdisciplinary project that investigates new modeling, simulation, and visualization methods for optimum bridge design.[1]

1.1 Simulation Data Mining for Design

An ideal interactive design workflow will let a designer specify design considerations in a familiar way, usually in a CAD (computer-aided design) program, and instantaneously provide the designer with feedback. An established feedback form is to render numerical simulation results obtained via a FEM (finite element method) analysis by coloring the CAD model. Feedback might also be given in the form of design alternatives, or as meta information such as reliability assessments concerning the simulation results. From this feedback, the designer evaluates the results and draws appropriate conclusions. We point out that a designer is planning and reasoning using a mental model in a kind of *design space*, while receiving feedback by interpreting selected results in a *simulation space*. See Figure 1 for an illustration.

The driving forces behind an ideal workflow are threefold: analysis performance, result comprehensibility, and ease of model manipulation. With simulation data mining, certain performance issues can be addressed: if the dialog between the human designer and the machine is impeded due to the model complexity or insufficient computing power, data mining technology can be applied to shift computation effort from the interactive (runtime) phase into a preprocessing phase. With preprocessing, design variants are instantiated and simulated offline, and the generated simulation data is exploited to learn heuristic connections between the design space and the simulation space. Of course, such heuristics are not intended to replace a deep analysis, but to guarantee a fluent dialog, since they can be evaluated at a fraction of the simulation runtime; the tentative results are superseded if the actual simulation results are at disposal. That is, the outlined advantage of simulation data mining disappears if performance bottlenecks are bypassed due to the use of less advanced models or the

This is research is supported by the German Thuringian State Ministry Grant B514-090521.

[1] "Strategies for the Robust Design of Structures", Thuringian Program on Excellence in Germany.

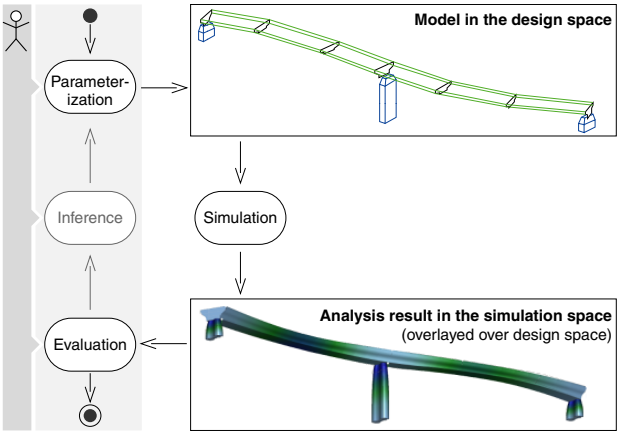

Figure 1: Interactive design workflow. Based on experience, a designer selects a model by parameterization, simulates it, evaluates the design decisions in the simulation space, and continues with a purposeful parameter modification in case of unsatisfactory results.

introduction of additional computing power, but also if the acquired heuristics are highly unreliable.

A second very interesting application of simulation data mining is not concerned with simulation speed, but with the increased volume of generated data in general. Today, in times of high computing power and seemingly endless storage capabilities, people expect that every aspect of a planned system (such as a bridge) that *could* be simulated also *should* be simulated. In other words, a design space may not be investigated for just a handful of points but for a wide range of parameter combinations. Simulation data mining then is applied to filter the set of simulation results and to identify the interesting design variants to be presented to the human designer. Even more, an exhaustive range of simulations can be used to organize the design space in the form of a topological map, exhibiting regions of good solutions, below average solutions, and unacceptable solutions. In the end, simulation data mining can drive a design optimization strategy.

1.2 Task, Approach, and Contributions

In bridge design, the task of a designer is to carefully manage a series of competing demands. The work by Spector and Gifford (1986) summarizes six kinds of such demands very well, concerning functionality, serviceability, ultimate strength, aesthetics, long-term maintainability, and cost-effectiveness, from which the last three are not relevant in our study. Conversely, the strength and serviceability of a bridge are interesting, as thresholds for key simulation outputs such as displacements, strains, and stresses can indicate likely cracking and failure, which relates to the weights and materials chosen. Moreover, the functionality of the bridge is also interesting, as it directly relates to the geometry chosen for the models to be simulated. Since the geometry design space is effectively infinite, some subset has to be selected, and we consider alternate geometries for the flat surface of a bridge model spanning a valley with three pillars. For the time being the dimensionality of the entire design space, that is, the number of parameters from a conceptual design perspective can be regarded as 10; Section 3.1 discusses the relevant dimensions of our design space in detail.

We now introduce an approach to support the outlined bridge design task using simulation data mining.[2] A first but very simple strategy to acquire design knowledge is to learn a mapping from the design space M onto selected dimensions of the simulation space Y. If such a mapping existed it could be used to answer design-relevant questions quickly, thereby circumventing an FEM simulation. But, except for simple design tasks, even sophisticated machine learning methods will fail to capture design constraints directly from raw simulation data. Instead, we start with the observation that two models $\mathbf{m}_1, \mathbf{m}_2$ in M are similar iff their simulation results (that is, their entire sets of displacement, strain, and stress values) $\mathbf{y}_1, \mathbf{y}_2$ in Y are similar:

$$\|\mathbf{y}_1 \ominus \mathbf{y}_2\| < \varepsilon \quad \Leftrightarrow \quad \varphi_{\text{Design}}(\mathbf{m}_1, \mathbf{m}_2) \approx 1, \qquad (1)$$

where $\ominus$ denotes a difference operator, and $\varphi_{\text{Design}} : M \times M \to [0;1]$ denotes a similarity function in the design space. A value of φ_{Design} close to one indicates a high similarity between two models and a value close to zero indicates a low similarity. This understanding of design similarity appears naturally and is rooted in the theory of case-based design problem solving (Maher and Pu, 1997; Antonsson and Cagan, 2001). Having φ_{Design} at our disposal we can address various issues within an interactive design workflow, such as the following:

- Given a model $\mathbf{m} \in M$, the most similar designs (in terms of behavior) can be looked-up by querying $\varphi_{\text{Design}}(\mathbf{m}, \cdot)$ in a k-nearest-neighbor fashion.
- Likewise, from the simulation results of its "design neighbors" a behavior valuation for $\mathbf{m}$ can be stated without an FEM simulation.
- From a set of models $M' \subseteq M$ that forms an equivalence class under φ_{Design}, one can learn design rules for cost optimization. Moreover, from the cardinality of M', information about the robustness of its elements may be derived.

The main contribution of our work relates to the construction of φ_{Design}. We propose to consider M as a subset of $\mathbf{R}^d$, where $d < 10$ is the dimensionality of the design space. The treatment of Y, however, is more involved: the dimensionality of the simulation space is determined by the output of an FEM analysis; that is, it depends on the resolution of the mesh discretization and introduces the according number of degrees of freedom. We hence apply an aggregation function $\alpha : Y \to Z$, which maps the original quantities of a simulation result vector $\mathbf{y} \in Y$ onto aggregate quantities $\mathbf{z} \in Z$, $\mathbf{z} \mapsto \alpha(\mathbf{y})$. While the dimensionality of Y is in 10^4 order of magnitude, that of Z is in 10^2. The function α considers physical connections and is optimized to capture as much as possible of a model's behavior characteristics. Based on these preliminaries we suggest the following steps to derive φ_{Design}, as visually summarized in Figure 2:

1. Sampling of m models $\{\mathbf{m}_i \in M \mid i = 1, \ldots, m\}$.
2. Simulation of the sampled models, yielding the set $\{\mathbf{y}_i \in Y \mid \mathbf{y}_i = \mu(\mathbf{m}_i)\}$, where μ denotes an FEM simulation algorithm.
3. Aggregation of the simulation results, yielding the set $\{\mathbf{z}_i \in Z \mid \mathbf{z}_i = \alpha(\mathbf{y}_i)\}$, where α denotes an aggregation function.

[2] Mathematical notation: sets, such as design or simulation spaces, are denoted by capital Latin characters, vectors are column vectors and denoted by small bold-faced Latin characters, and functions are denoted by small Greek letters.

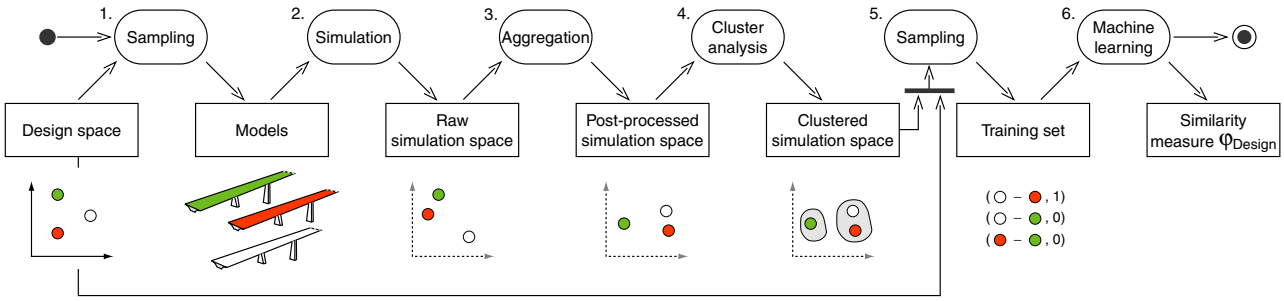

Figure 2: Six-step process to derive the similarity measure φ_{Design} for the design space.

4. Determining groups of similar models by clustering the $\mathbf{z}_i \in Z$ in the aggregated simulation space.

5. Sampling of n data points $D := \{\langle \mathbf{m}_k \ominus \mathbf{m}_l,\, c_j \rangle \mid k, l \in (1, \ldots, m),\ k < l,\ j = 1, \ldots, n\}$, where $\ominus$ denotes a difference operator and c_j is defined as follows:

$$c_j = \begin{cases} 1, \text{if } \alpha(\mu(\mathbf{m}_k)), \alpha(\mu(\mathbf{m}_l)) \text{ in same cluster,} \\ 0, \text{otherwise.} \end{cases}$$

μ and α denote the simulation algorithm and the aggregation function respectively.

6. Computing of φ_{Design} as a class probability estimator from the data points in D by means of machine learning. In particular, φ_{Design} is determined by a weight vector $\mathbf{w}^*$ that minimizes a combined error measure:

$$\mathbf{w}^* = \underset{\mathbf{w} \in \mathbf{R}^d}{\operatorname{argmin}} \sum_{(\mathbf{x}, c) \in D} L(c,\, \mathbf{w}^T \mathbf{x}) + \frac{\lambda}{2} \|\mathbf{w}\|^2,$$

where L is a loss function and λ is a non-negative regularization parameter that penalizes model complexity.

Other contributions of our research relate to the algorithmic and methodological details of the above six-step process: (1) the application of density-based clustering technology to determine equivalence classes in the simulation space, and (2) a means to evaluate φ_{Design} with machine learning by evaluating the accuracy of the mapping of the equivalence classes in both the design and the simulation spaces.

The remainder of this paper is organized as follows. Section 2 provides a literature review of simulation data mining, Section 3 details the above six-step process to derive φ_{Design}, Section 4 reports on selected experiments and analyses that illustrate the effectiveness of our approach, and Section 5 offers concluding remarks.

2 Related Work

Data mining is deployed when large numbers of data records are created and knowledge can be distilled from this data. So it seems to be quite natural to apply data mining techniques to the field of simulation. The field of possible application is wide and includes assistant systems for choosing an optimal or at least suitable simulation approach out of the set of possible algorithms (Ewald et al., 2008) as well as competing variables in semiconductor manufacturing in semiconductor factories (Brady and Yellig, 2005) or studies concerning aircraft engine maintenance (Painter et al., 2006). On closer inspection, it can be seen that the studied simulations are concentrating more on discrete event simulation fields or ones that are closely related to computer science research fields such as agent based modeling and simulation (Baqueiro et al., 2009). However, the interdisciplinary field of continuous or hybrid simulation is given less attention. An active field, where data mining is applied to finite element simulations, are crash test scenarios (Kuhlmann et al., 2005; Mei and Thole, 2008; Zhao et al., 2010), which form quite different points of view.

It is important to recall that simulation just gains knowledge about the model, and not about the real system. This fact requires additional verification and validation steps as described by Baqueiro et al. (2009). When dealing with continuous models that arise typically in engineering and natural science, like the finite element method in this paper, this step can be of less importance when dealing with pure mathematical continuous models. Also, the effects of the continuous model itself are the goal of data mining. In the work by Mei and Thole (2008), data mining techniques are used to find the reason for uncertainties in numerical simulation results. A typical reason is a change concerning regularity in the finite element model, caused for example by changes of the angle of elements around 90 degrees (Mei and Thole, 2008). So this is not a property of the real world system, but a property of the numerical model in combination with an unsatisfactory implementation.

It is quite common to pre-process the finite element method data, as noted by Kuhlmann et al. (2005): "To the authors best knowledge no approach for data mining on raw finite element data exists." The approach of Kuhlmann et al. (2005) to restore the physical properties from the finite element method data is related to our data pre-processing, but with different improvements and adoptions to the application area. In contrast to the research mentioned above, we develop an assistant system for engineers that provides real-time feedback during the design process. The process will finally be validated by a full numerical simulation, so our approach deals with both: on one hand, the real world system is the one the engineer has in mind, but on the other hand, the numerical model simulation will be the last step in this part of the design process: the user assumes that the advice given based on the data mining knowledge is close to the numerical model he or she will finally solve.

3 Development of φ_{Design}

The development of φ_{Design} follows the six steps summarized in Figure 2, which comprise sampling (from the design space), simulation, aggregation, cluster analysis, sampling (for training), and machine learning. The following subsections present all of these steps in detail.

Figure 3: A typical bridge from the design space, comprised of a curved solid surface and three pillars.

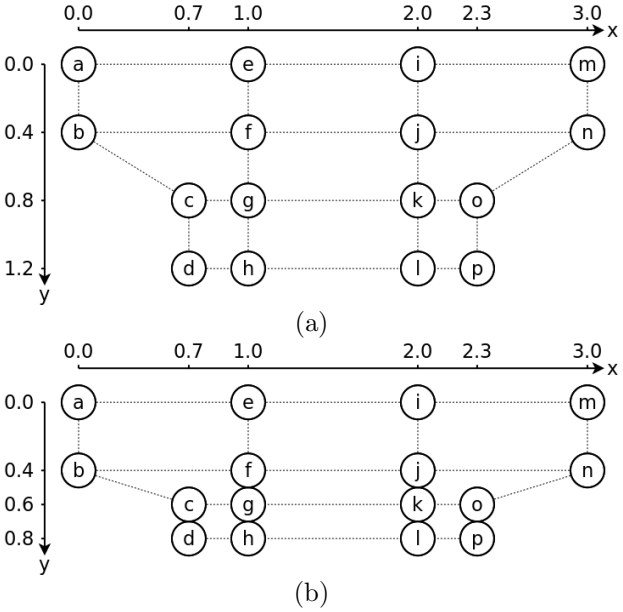

Figure 4: Two bridge surface cross-sections approximated by a trapezoid-like shape each using 16 points.

3.1 Sampling from the Design Space

All of our generated bridge models are represented using the IFC (Industry Foundation Classes) standard for data in the construction and building industries (Liebich, 2009) along with the IFC-BRIDGE extension (Lebegue et al., 2007). In this regard, researchers in our project group have added a novel extension using NURBS (non-uniform rational basis spline) solids (Hughes et al., 2005). Figure 3 shows a design variant from the design space. It is comprised of a curved solid surface and three pillars. The surface is comprised of a spline with seven 16-point cross sections evenly interspersed along the length of the spline. The two unique cross-sections are shown in Figure 4. The thicker cross-section shown in Figure 4a is positioned above each of three pillars. The remaining four cross sections shown in Figure 4b are positioned evenly between the two pairs of pillars. Observe how the NURBS modeling creates the smooth curves in Figure 3 from the original partial specifications that lack curves in Figure 4. Though the pillars are equally interesting, we have decided to not focus on these in this paper. Finally, we note that the curvature of the bridge surface is not completely symmetrical between the two pairs of pillars, so we expect different simulation results for each segment.

The design space covers the key parameters that are considered during the conceptual design phase: material properties and the geometry. Though the sensible variations in these parameters have been discussed with expert colleagues, each design dimension is sampled independently from the others to avoid both an implicit encoding of design preferences and search biases.

The material properties we explore are Young's modulus, which measures material stiffness (Mitchell and Green, 1999), and density. The values we use for these parameters are in the ranges [2e+10, 3e+10] (gigapascal) and [23, 25] (kilogram per cubic meter) respectively. The derived geometry features relate to the two bridge cross-sections as explained in Figure 4. From the figure it is clear that various cross-section heights and widths can be altered to create new geometries, such as $height_{\text{row2}-\text{row1}} = b_y - a_y$. We devise four rules for widths and three rules for heights and contract or expand the resulting values within the range [-0.3, 0.3] units to create new geometries.[3] We note that the four widths are always expanded or contracted by the same amount at any given time to preserve the rough shape of the cross section. This is less important for the three heights, but we again apply the adjustments evenly so that we can achieve the same number of height and width adjustments. Altogether, we sample 14 641 ($= 11^4$) models from the design space comprising:

1. eleven values for Young's modulus {2.0e+10, 2.1e+10, 2.2e+10, 2.3e+10, 2.4e+10, 2.5e+10, 2.6e+10, 2.7e+10, 2.8e+10, 2.9e+10, 3.0e+10}.
2. eleven values for density {23.0, 23.2, 23.4, 23.6, 23.8, 24.0, 24.2, 24.4, 24.6, 24.8, 25.0}.
3. eleven bridge surface cross-section height adjustments {-0.30, -0.24, -0.18, -0.12, -0.06, 0.00, 0.06, 0.12, 0.18, 0.24, 0.30} applied uniformly.
4. eleven bridge surface cross-section width adjustments {-0.30, -0.24, -0.18, -0.12, -0.06, 0.00, 0.06, 0.12, 0.18, 0.24, 0.30} applied uniformly.

We believe this design space is interesting for the defined model as it represents a good mixture of model properties, input parameters, and increments. However, it must nevertheless be noted that the model is not industrial-scale, so proving that our methods work is of more interest than the defined design space itself.

3.2 Simulation

A lot of phenomena in engineering are modeled using partial differential equations. One of the most popular approaches to simulate these models is the finite element method (FEM). For stationary models such as ours, the FEM only requires a discretization in space. Traditional FEM simulation uses triangles or rectangles in 2D and tetrahedrons or cuboids in 3D. Nowadays, the usage of NURBS is becoming increasingly popular. Independent of the used technique, the geometry is described by elements that are given by nodes. The number of nodes and the modeled physical phenomenon influence the number of degrees of freedom. Beyond this, the number of nodes together with the used base functions affects the quality of the simulation results. It is an oversimplification that more nodes always leads to more accurate results, because there are various conditions to fulfill, as described by Brenner and Scott (2002) and Quarteroni (2009). If these necessary conditions are fulfilled, then generally one can say that a higher number of nodes increases the quality of the results. This happens mainly because of two effects: The first is that the approximation quality of the geometry might be increased. The second is that even if the geometry has

[3] Geometries use a relative and user-interpreted unit of measurement in the simulation environment, which could be meters.

already been perfectly captured, the numerical approximation is increased. In our work, most aspects such as the number of nodes and the mathematical model have not been changed for the data mining. We in part concentrated on changing the geometry by moving the nodes. Moving the nodes on the boundary means changing the physical model, which obviously leads to different simulation results. Nevertheless, one must keep in mind that different geometries not only imply different physical behavior, but a different numerical behavior. This means that some attributes of the geometry itself might increase or decrease the numerical error, which might influence data mining processes. If the boundary changes regarding regularity (Brenner and Scott, 2002; Quarteroni, 2009), this will influence the simulation results. The aspects of the angles are strongly limited by the constraints we have given to geometry changes. If the physical parameters such as materials inside the volume are changed from one run of the simulation to the next, the FEM will have different properties. The FEM has the best properties if the parameters change very smoothly over the volume. A very rough or unfavorable discretization of the geometry might also influence the simulation results beyond the physical effects that ought to be simulated.

3.3 Aggregation

The simulation output returned from the FEM simulation uses the VTK (Visualization Toolkit) framework (Schroeder et al., 1996), which allows derived output (such as bridge displacements, strains, and stresses) to be viewed with three-dimensional models in visualization tools such as ParaView.[4] The VTK files include different sections with data concerning nodes, the resulting elements, and the simulation results at these nodes. Because the Visualization Toolkit framework is not designed to work with NURBS, the visualization and the data are mapped on a traditional mesh using hexahedrons as elements. The output in the VTK legacy format is not only used for visualization, but also for the input for our clustering and data mining processes.

The simulation output features available in our framework are displacements, strains, and stresses. A displacement is the amount of movement at any given point expressed as a vector in the three-dimensional space. Strains and stresses are instead expressed as matrices, or second-order tensors.[5] We reduce the second-order tensors to vectors using a vector triple product in order to have identical expressions for displacements, strains, and stresses. Individual measurements for displacements, strains, and stresses are expressed in the simulated output, with one measurement per point in the output mesh. Since interpolations of the finite element method simulations can provide results for thousands of points, it is therefore necessary to aggregate or combine these in some fashion, such as expressing maximums, averages, variances, standard deviations, and so on of vector measurements.

We focus on maximum measurements in the output, which is based on the idea that the most severe localized results are of high interest. The output granularity level we use interpolates 12 064 measurements from each finite element method simulation of our model, which we aggregate to 45 ($3 \times 3 \times 5$) dimensions:

- three types of output variables comprising displacements, strains, and stresses
- all three X, Y, and Z dimensions
- five divisions of the model to capture localized results comprising three at the pillars and two in between

All values are normalized in the range [0,1], so that no individual measurement completely dominates any other in a similarity measurement such as cosine. Other possible measurements (averages, variances, standard deviations) are left as future work. We note that care is needed when dealing with sets of values that differ in many orders of magnitude.

3.4 Cluster Analysis

In order to identify groups of models with similar simulated behavior, we apply the k-means (Hartigan and Wong, 1979) and Hierarchical Agglomerative Clustering (HAC) (Gowda and Krishna, 1978) clustering algorithms as competing alternatives. Both of these algorithms are *hard* clustering algorithms, meaning that each instance is assigned to only one cluster each, unlike in *soft* clustering, where each instance may be assigned to multiple clusters. The number of clusters used by each algorithm needs to be carefully considered. For k-means, the clustering must be done on a pre-defined number of clusters, which requires evaluation of many alternatives. For HAC, the clustering begins with each instance in its own cluster, and the clusters are merged one at a time in a bottom-up fashion until some optimum configuration is found, or a given threshold is reached.

In both cases, the quality of each clustering needs to be evaluated and compared with the alternatives. In this work, we use the Expected Density (Stein et al., 2003) measure to measure the quality. This measure gives a value beginning from 1.0 where a higher value denotes higher quality. A value of 1 is assigned to any clustering where the instances are either all in one cluster or are each in their own cluster, but the optimum is expected to be somewhere in between for meaningful scenarios. In practice, the expected density measure may peak at some configuration providing a meaningful recommendation for a cluster configuration to be adopted.

After some set of optimal clusterings are chosen, each pair of samples is assigned either a 1 or 0 denoting whether they appear in the same cluster or not. This data forms part of the machine learning training data described in Section 3.6.

3.5 Sampling for Training

There are many considerations for selecting a meaningful training set from the full set of pairs generated above. Perhaps most obvious is the fact that the number of within-cluster and between-cluster pairs is rarely even, so this needs attention or our machine learning algorithms may be adversely affected by the class imbalance problem (Ertekin et al., 2007). Furthermore, many clusters are likely to be of unequal size meaning that any simple random sampling strategy would overly represent the largest clusters at the expense of the smaller ones, therefore the sampling should be balanced between the clusters. A possible refinement for future work could apply bias towards clusters with the highest density. Finally, the cluster sizes are not relevant for sampling the negatives, so we simply randomly sample an equivalent number of negatives in this case. A possible refinement for

[4] `http://www.paraview.org`

[5] Note that a vector is a first-order tensor.

future work for the negative samples is to introduce bias towards the clusters that are furtherest apart.

The training samples for the machine learning step are the tuples in the form $\langle \mathbf{m}_k \ominus \mathbf{m}_l, c_j \rangle$ as described in Section 1.2. The $\mathbf{m}_k \ominus \mathbf{m}_l$ component is the vectorial difference of the four features in the design space for a pair of designs with each dimension normalized in the range [0,1], so that no individual dimension dominates the others. This leaves the c_j component, which takes on the value 1 or 0 for within-cluster or between-cluster pairs respectively, as determined by the clustering algorithm.

3.6 Machine Learning

For the machine learning step, we consider the class probability estimates of two classifiers that were shown to produce meaningful class probability estimate distributions in previous unrelated work (Burrows et al., n.d.): naive bayes and maximum entropy. Suitable implementations were taken from the Weka machine learning toolkit.[6] We apply these classifiers to generate the class probability estimates by applying ten-fold cross validation. When evaluating our approach, the machine learning accuracy scores can provide a strong endorsement of our methodology and similarity measure φ_{Design}. Also, using the ranked lists from class probability estimates, there is potential to evaluate the rankings using a rank correlation coefficient to those of the simulation space computed by the cosine distance between the aggregated results in future work.

4 Experiments and Analysis

This section first provides intermediate experiment results and analysis obtained from the clustering results and from sampling the simulation data. Then, we analyze the overall performance of φ_{Design} using our machine learning classification scores. Finally, we review other methods that may be applied to further validate our work in the future.

4.1 Clustering Results

Our expected density results of our clustering indicate that the optimum number of clusters for k-means and HAC is 12 and 37 respectively. Both distributions of results form bell-curves, making the decision simple. Ideally, these numbers should be closer, so we err on the side of caution and proceeded with both results for both cluster algorithms, making four combinations. The trends are given in Figure 5, which show that the peaks are not steep, so having some variation between clustering algorithms is tolerable.

4.2 Sampling Strategy Analysis

As described in Section 3.5, we want an equal number of positive and negative pairs, and for the positive pairs to be evenly sampled from all clusters. The minimum cluster sizes that are created is 220 instances when 12 clusters are built, and 110 instances when 37 clusters are built. Given that the number of pairs in any cluster is $\frac{n(n-1)}{2}$, this gives us 66 and 666 positive pairs per cluster, and 14 520 and 73 260 positive pairs in total respectively to satisfy our rule. These numbers then double when an equivalent number of randomly selected negative samples are chosen. Larger samples can be drawn if the smaller clusters are underrepresented.

[6] We used the classifiers weka.classifiers.bayes.NaiveBayes and weka.classifiers.functions.Logistic from Weka 3.6.5.

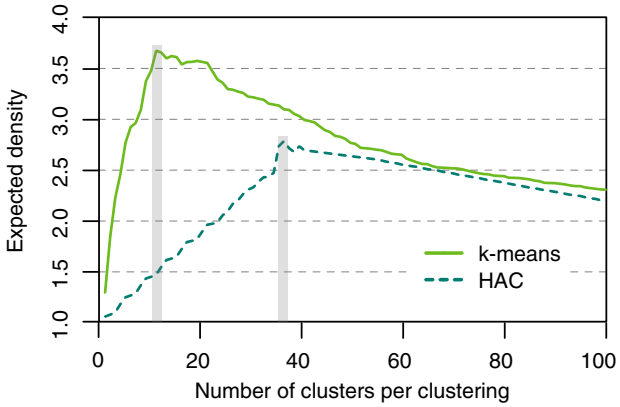

Figure 5: Expected density results for k-means and HAC clusterings from 1 to 100 clusters. These results show variation between the optimum, however there are several good choices for each clustering, so some variance is acceptable.

Algorithm	k-means		k-means		HAC		HAC	
Cluster	605	1	110	8	220	6	220	16
distribution	660	1	220	11	330	3	330	6
(size and	726	1	330	3	440	2	440	12
frequency)	990	3	363	1	11 451	1	550	1
	1 320	2	440	2			1 331	1
	1 760	4	484	2			1 980	1
			550	2				
			880	8				
Clusters	12		37		12		37	

Table 1: Cluster sizes and frequencies for k-means and HAC clustering algorithms for 12 and 37 clusters.

Table 1 shows the actual distribution of cluster sizes formed. There is a clear outlier of 11 451 instances for the largest cluster for HAC when 12 clusters are formed. This is one of the two non-optimal clusterings as far as expected density is concerned, so this anomaly further justifies the use of the expected density measure. Overall, for our two best clusterings concerning expected density, we have 10 064 780 positive and 97 107 340 negative instances for k-means with 12 clusters, and 4 865 410 positive and 102 306 710 negative instances for HAC with 37 clusters. The sampling strategy therefore only takes a small fractions of all instances available, with the proportion of available positives around 5-10% compared with the pool of negatives.

4.3 Classification Results

A consequence of generating class probability estimates for the purposes of ranking results is that we can additionally test the effectiveness of our approach by evaluating the mapping from the design space onto the within-cluster and between-cluster class labels as a binary classification experiment. We explore these accuracy results with two alternatives for each choice of clustering method, cluster size, and classification algorithm as given in Table 2. The combinations explored are the k-means and HAC classifiers using the optimum cluster size for each as determined by the expected density measure, plus the reverse setting. These four combinations of clusterings are combined with the Naive Bayes and Maximum Entropy classification algorithm choices for generating the class probability estimates. These results show accuracy scores around 92–93% for several settings, including the configurations using maximum entropy and 12 clusters in

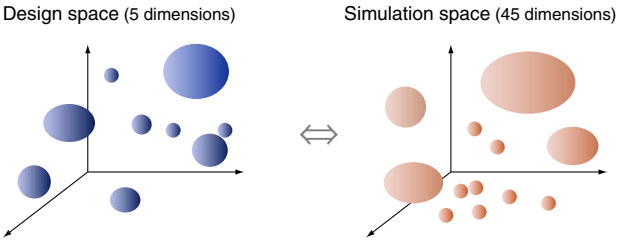

Figure 6: The purpose of φ_{Design} is to demonstrate a mapping between the design and simulation spaces.

particular. In comparison to the naive random-chance baseline of 50% accuracy, we can conclude that we have good initial results to support φ_{Design}.

4.4 Rank Correlation Co-efficients

In future work, our next step to justify φ_{Design} will be with the use of a rank correlation co-efficient such as Spearman's *rho*, Pearson's *r*, or Kendall's *tau* to compare the correlation of the ranks of φ_{Design} with the ranks of the cosine similarity taken from the simulation space. That is, we want to demonstrate some kind of correlation between the design space and simulation space as shown in Figure 6. This additional evidence will be of much value since our results demonstrating the mapping of the design space to the cluster class labels from Section 4.3 are coarse. An open problem we are working on is the generation of class probability estimates with less duplicates, as these are far from ideal for calculating correlation co-efficients relying on ranks. We will consider methods to interpolate the aggregated values from the simulation space (discussed in Section 3.3) since there are often very small differences in the values. Alternatively, different choices for clustering algorithms or sampling strategy methods (Sections 3.4 and 3.5 respectively) may also alleviate this problem.

An alternative to the rank correlation co-efficients is to consider the design space to simulation space correlations in classes such high, medium, neutral, and not similar, to treat duplicates in a more uniform way if many remain after exploring the above ideas. Therefore a rank correlation co-efficient will have to deal with duplicates in this case. Another option is to apply clustering instead of ranking for the evaluation. In this regard two cluster analyses have to be performed, one in the design space and the other in the simulation space, whereas the underlying similarity measures are φ_{Design} and the cosine similarity respectively. The two resulting clusterings are then compared by a weighted F-measure analysis, which quantifies the quality of the coverage based on weighted precision and recall values (this idea is also suggested by Figure 6). Though this comparison involves a lot of hyperparameter tuning, it should be considered as one of the most comprehensive approaches to evaluate the entire six-step process shown in Figure 2. We hence will focus on coverage-based evaluation in the near future.

5 Summary and Conclusions

To recap, real-life solid models are too complex to be processed within a reasonable amount of time and interactive design can only work if updated results triggered by model modifications from the user can be made available almost instantaneously. Since equation solving for complex FEM simulations requires a huge amount of computing power, alternative methods are consequently required to provide simulation results for complex models. With our work we introduce simulation data mining as such a method. The starting point of simulation data mining is to pre-compute a large number of models. Upon model modification by the designer, a finite element simulation as well as a φ_{Design}-search in the model database is launched, and the results of the first-completed operation are sent to the user. Depending on the size of the database, the exact model requested may not be available, but the closest available model could be returned as an approximation instead. For sufficient model complexity, the database search is order of magnitudes faster than the real simulation; the intermediate results are replaced as soon as the real simulation finishes without disturbing the designer's workflow.

We have proposed the development of φ_{Design} based on the idea that two models in the design space are similar if their simulated behavior is similar. This is necessary as the mapping is otherwise highly complex due to the numerical analysis in FEM simulation. Our similarity measure φ_{Design} is constructed based on a six-step approach comprising choosing a design space, simulating the models, aggregating the simulated results, clustering the simulated models to identify similar behavior, sampling some subset of the clustering pairs using an appropriate selection strategy, and then developing class probability estimates to show that we can map unseen examples from the design space to the similarity space.

To date, the first iteration of φ_{Design} is essentially complete. We have demonstrated that we can map the design space to our binary class labels from our

	Accuracy for k-means				Accuracy for HAC			
	12 clusters		37 clusters		12 clusters		37 clusters	
Training size	Naive bayes	Entropy	Naive bayes	Entropy	Naive bayes	Entropy	Naive bayes	Entropy
100	94.0	94.0	82.0	81.0	86.0	87.0	94.0	96.0
200	92.5	93.0	82.5	83.5	88.5	92.5	90.0	91.0
500	85.4	90.6	83.0	85.4	89.2	92.2	89.8	90.8
1 000	91.4	94.3	84.5	86.4	91.5	93.0	90.8	91.2
2 000	88.6	92.4	84.9	85.9	88.4	92.0	89.8	91.0
5 000	89.4	92.7	84.3	84.6	88.5	92.1	89.3	90.5
10 000	89.6	92.4	84.6	85.2	89.7	92.7	88.5	89.4
20 000	89.8	92.3	84.0	84.8	89.9	93.0	89.6	90.3
50 000	89.9	92.7	84.6	85.2	89.5	92.4	89.1	90.1
100 000	89.7	92.4	84.6	85.4	89.6	92.6	89.1	89.7
200 000	89.8	92.5	84.6	85.2	89.5	92.5	89.1	89.8
500 000	89.8	92.5	84.6	85.3	89.4	92.4	89.1	89.7

Table 2: Classification accuracy for k-means (columns 2-5) and HAC (columns 6-9) clustering algorithms, clustering sizes of 54 (columns 2-3 and 6-7) and 110 (columns 4-5 and 8-9), and naive bayes and maximum entropy classifiers (alternating columns). Many classification accuracy scores around 92–93% are achieved.

clusterings with high accuracy. Future work is needed to extend the verification to the class probability estimate rankings, whether we use alternative methods to develop the class probability estimates, or group or cluster the rankings. Other intermediate results are of interest where we have been able to provide some initial estimates about the number of equivalence classes in our design space and their sizes from our clustering results.

There is much scope for further development of the ideas presented in this paper. Indeed, the methodology comprises six distinct steps, where we have evaluated a small number of alternate options at each step, or simply just made a single decision. We plan to explore the available alternatives in much depth in the future.

Acknowledgements

We thank Fabian Gerold, Tim Gollub, Katja Müller, Michael Schwedler, and David Wiesner for their technical support on this project beyond the research contributions of this paper.

References

Antonsson, E. and Cagan, J. (2001), *Formal Engineering Design Synthesis*, Cambridge University Press.

Baqueiro, O., Wang, Y. J., Mcburney, P. and Coenen, F. (2009), Integrating Data Mining and Agent Based Modeling and Simulation, *in* P. Perner, ed., 'Proceedings of the Ninth Industrial Conference on Advances in Data Mining', Springer-Verlag, Leipzig, Germany, pp. 220–231.

Brady, T. F. and Yellig, E. (2005), Simulation Data Mining: A New Form of Computer Simulation Output, *in* M. E. Kuhl, N. M. Steiger, F. B. Armstrong and J. A. Joines, eds, 'Proceedings of the Thirty-Seventh Winter Simulation Conference', Orlando, Florida, pp. 285–289.

Brenner, S. C. and Scott, L. R. (2002), *The Mathematical Theory of Finite Element Methods*, second edn, Springer, Berlin, Germany.

Burrows, S. (2011), Simulation Data Mining in Artificially Generated Data, *in* H. Cunningham, O. Etzioni, N. Fuhr and B. Stein, eds, 'Proceedings of the Schloss Dagstuhl Seminar on Challenges in Document Mining', Leibniz-Zentrum für Informatik, Wadern, Germany. Invited talk.

Burrows, S., Potthast, M. and Stein, B. (n.d.), 'Paraphrase Acquisition via Crowdsourcing and Machine Learning'. In submission.

Ertekin, S., Huang, J. and Giles, C. L. (2007), Active Learning for Class Imbalance Problem, *in* W. Kraaij, A. P. de Vries, C. L. A. Clarke, N. Fuhr and N. Kando, eds, 'Proceedings of the Thirtieth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', ACM, Amsterdam, The Netherlands, pp. 823–824.

Ewald, R., Himmelspach, J. and Uhrmacher, A. M. (2008), An Algorithm Selection Approach for Simulation Systems, *in* F. Quaglia and J. Liu, eds, 'Proceedings of the Twenty-Second Workshop on Principles of Advanced and Distributed Simulation', IEEE Computer Society, Rome, Italy, pp. 91–98.

Gowda, K. C. and Krishna, G. (1978), 'Agglomerative Clustering using the concept of Mutual Nearest Neighbourhood', *Pattern Recognition* **10**(2), 105–112.

Hartigan, J. A. and Wong, M. A. (1979), 'Algorithm AS 136: A K-Means Clustering Algorithm', *Journal of the Royal Statistical Society* **28**(1), 100–108.

Hughes, T. J. R., Cottrell, J. A. and Bazilevs, Y. (2005), 'Isogeometric Analysis: CAD, Finite Elements, NURBS, Exact Geometry and Mesh Refinement', *Computer Methods in Applied Mechanics and Engineering* **194**(39–41), 4135–4195.

Kuhlmann, A., Vetter, R.-M., Lübbing, C. and Thole, C.-A. (2005), Data Mining on Crash Simulation Data, *in* P. Perner and A. Imiya, eds, 'Proceedings of the Sixth International Conference of Machine Learning and Data Mining', Springer Berlin / Heidelberg, Leipzig, Germany, pp. 635–635.

Lebegue, E., Gual, J., Arthaud, G. and Liebich, T. (2007), IFC-BRIDGE V2 Data Model, Technical Report Edition R8, buildingSMART International Modeling Support Group.

Liebich, T. (2009), IFC 2x Edition 3 Model Implementation Guide, Technical Report Version 2.0, buildingSMART International Modeling Support Group.

Maher, M. and Pu, P., eds (1997), *Issues and Applications of Case-Based Reasoning in Design*, Lawrence Erlbaum Associates, Mahwah, New Jersey.

Mei, L. and Thole, C.-A. (2008), 'Data Analysis for Parallel Car-Crash Simulation Results and Model Optimization', *Simulation Modelling Practice and Theory* **16**(3), 329–337.

Mitchell, B. and Green, A. (1999), 'The Young Modulus', `http://www.matter.org.uk/schools/content/YoungModulus`. Accessed 21 August, 2011.

Painter, M. K., Erraguntla, M., Hogg, G. L. and Beachkofski, B. (2006), Using Simulation, Data Mining, and Knowledge Discovery Techniques for Optomized Aircraft Engine Fleet Management, *in* D. Nicol, R. Fujimoto, B. Lawson, J. Liu, F. Perrone and F. Wieland, eds, 'Proceedings of the Thirty-Eighth Winter Simulation Conference', IEEE Press, Monterey, California, pp. 1253–1260.

Quarteroni, A. (2009), *Numerical Models for Differential Problems*, first edn, Springer, Milan, Italy.

Schroeder, W. J., Martin, K. M. and Lorensen, W. E. (1996), The Design and Implementation of an Object-Oriented Toolkit for 3D Graphics and Visualization, *in* R. Crawfis, C. Hansen, N. Max and S. Uselton, eds, 'Proceedings of the Seventh Conference on Visualization', IEEE Computer Society Press, San Francisco, California, pp. 93–99.

Spector, A. and Gifford, D. (1986), 'A Computer Science Perspective on Bridge Design', **29**, 267–283.

Stein, B. (2001), Model Construction in Analysis and Synthesis Tasks, Habilitation, University of Paderborn, Germany, Department of Mathematics and Computer Science.

Stein, B. (2003), Model Compilation and Diagnosability of Technical Systems, *in* M. Hanza, ed., 'Proceedings of the Third International Conference on Artificial Intelligence and Applications', ACTA Press, Anaheim, Calgary, Zurich, pp. 191–197.

Stein, B., Meyer zu Eißen, S. and Wißbrock, F. (2003), On Cluster Validity and the Information Need of Users, *in* M. Hanza, ed., 'Proceedings of the Third International Conference on Artificial Intelligence and Applications', ACTA Press, Anaheim, Calgary, Zurich, pp. 216–221.

Zhao, Z., Jin, X., Cao, Y. and Wang, J. (2010), 'Data Mining Application on Crash Simulation Data of Occupant Restraint System', *Expert Systems with Applications* **37**, 5788–5794.