Applied Intelligence (2025) 55:614
https://doi.org/10.1007/s10489-024-05945-6

A modified SimRank++ approach for searching crash simulation data

Anahita Pakiman[1,2] · **Jochen Garcke**[1,3] · **Axel Schumacher**[2]

Accepted: 10 October 2024 / Published online: 2 April 2025

Abstract
Data searchability has been utilized for decades and is now a crucial ingredient of data reuse. However, data searchability in industrial engineering is essentially still at the level of individual text documents, while for finite element (FE) simulations no content-based relations between FE simulations exist so far. Additionally, the growth of data warehouses with the increase of computational power leaves companies with a vast amount of engineering data that is rarely reused. Search techniques for FE data, which are in particular aware of the engineering problem context, is a new research topic. We introduce the prediction of similarities between simulations using graph algorithms, which for example allows the identification of outliers or ranks simulations according to their similarities. With that, we address searchability for FE-based crash simulations in the automotive industry. Here, we use SimRank-based methods to predict the similarity of crash simulations using unweighted and weighted bipartite graphs. Motivated by requirements from the engineering application, we introduce SimRankTarget++ an alternative formulation of SimRank++ that performs better for FE simulations. To show the generality of the graph approach, we compare component-based similarities with part-based ones. For that, we introduce a method for automatically detecting components in the vehicle. We use a car sub-model to illustrate the similarity ansatz and present results on data from real-life development stages of an automotive company.

Keywords FE analysis · Automotive · Searchability · Semantic data · Outlier detection · CAE knowledge · Knowledge graph · Graph database · SimRank · SimRank++

1 Introduction

The introduction of the semantic web at the beginning of the 20th century supported a 'web of data' rather than a 'web of documents' [21]. Semantics enhanced searchability, making data more usable. Further, graph algorithms using interconnectivity and semantics can rank the similarity of existing entities and predict missing links in the data. However, many engineering domains still do not utilize these modern data analysis technologies.

In recent decades, computer-aided engineering (CAE) has become crucial in automotive R&D, with OEMs running 10,000 to 30,000 simulations weekly [23]. Despite this, CAE data often remains disconnected and underutilized, hindering collaboration and problem-solving, with much data becoming dark data [22]. In particular, the lack of searchability limits the re-use of data and the gain of insight across many simulations. This at a time where even a single engineer can barely remember simulations created over the span of a week. The underlying data access and processing issues have significant implications across teams and organizations. Therefore we introduce an approach for searchability to an automotive CAE domain, namely crash simulation, where we furthermore address clustering of results, identification of outliers, assessing solution robustness, and ranking behaviors according to their similarity to existing designs.

To the best of our knowledge, no methods are currently available that focus on searchability in CAE simulations. In order to introduce the semantics required for this, we ear-

✉ Anahita Pakiman
p.anahita@gmail.com

Jochen Garcke
jochen.garcke@scai.fraunhofer.de

Axel Schumacher
schumacher@uni-wuppertal.de

[1] Fraunhofer SCAI, Sankt Augustin, Germany

[2] Bergische Universität Wuppertal, Wuppertal, Germany

[3] Institut für Numerische Simulation, Universität Bonn, Bonn, Germany

lier established a knowledge graph (KG) for the automotive industry, called car-graph, with a focus on the use case of CAE crash simulations [18, 19]. Graphs allow more flexibility than typically possible in a relational setting, which makes it an ideal fit for the fast-evolving CAE process in the automotive industry [18]. In this work, we focus on using graph algorithms on the car-graph to provide a similarity score for simulation pairs. Ranking and clustering of these scores then provides searchability for simulations.

We are basing the prediction of similarities in the car-graph on the SimRank method [12]. SimRank estimates the similarity of two nodes based on their connectivity. Since in our application it is crucial to include edge weights, we study SimRank++ [2], a SimRank extension that allows this. Furthermore, we introduce SimRankTarget++, a modification of SimRank++ that is better suited to our use case. Due to a change in how edge weights are used, SimRankTarget++ focuses on how each car part performs in different simulations, rather than on how car parts perform in the same simulation, as is the case with SimRank++.

As we see later, treating groups of car parts as components in the similarity assessment can be more physically meaningful. Consequently, we introduce a method for the automatic detection of components in the CAE model of a vehicle. With that, we have two variants for the similarity prediction: the finite element (FE) parts individually and a group of parts representing components.

There are no public benchmark data available for simulation similarity prediction. Furthermore, labeling the data to characterize the crash behavior is a complex engineering task; the behaviors are usually not classified and typically multi-criterial. To evaluate the methods, we therefore introduce an illustrative example that replicates real development changes and allows at least a relative labeling of the crash behavior.

Figure 1 provides an abstract visualization of the analysis process presented in this paper. The process begins with the car-graph, where the first step involves incorporating parts grouping into the data model through component detection. The next stage involves evaluating the SimRank method for part-based and component-based graphs, utilizing both labeled and unlabeled data. For the unlabeled data, we introduce a similarity distribution approach and select specific groups of simulations to verify the methods based on domain expert analyses.

In Section 2 we recapture related work, followed by a description of the simulation setups for an illustrative example in Section 3. Then we introduce the SimRank methods and our extension in Section 4. We present our method for component detection in Section 5 and show results for the illustrative example. Next, we introduce an energy diagram that follows the crash behavior in Section 6, and in Section 7 we use these relative labels to assess the similarity predictions and rankings. Further, we summarize the outcome of similarity prediction for the illustrative example in Section 7, where we also evaluate our approach on unlabeled industrial data from several development stages in a project of China Euro Vehicle Technology AB (CEVT) In Section 8 we show an example of user workflow and the conclusion and outlook are in Section 9.

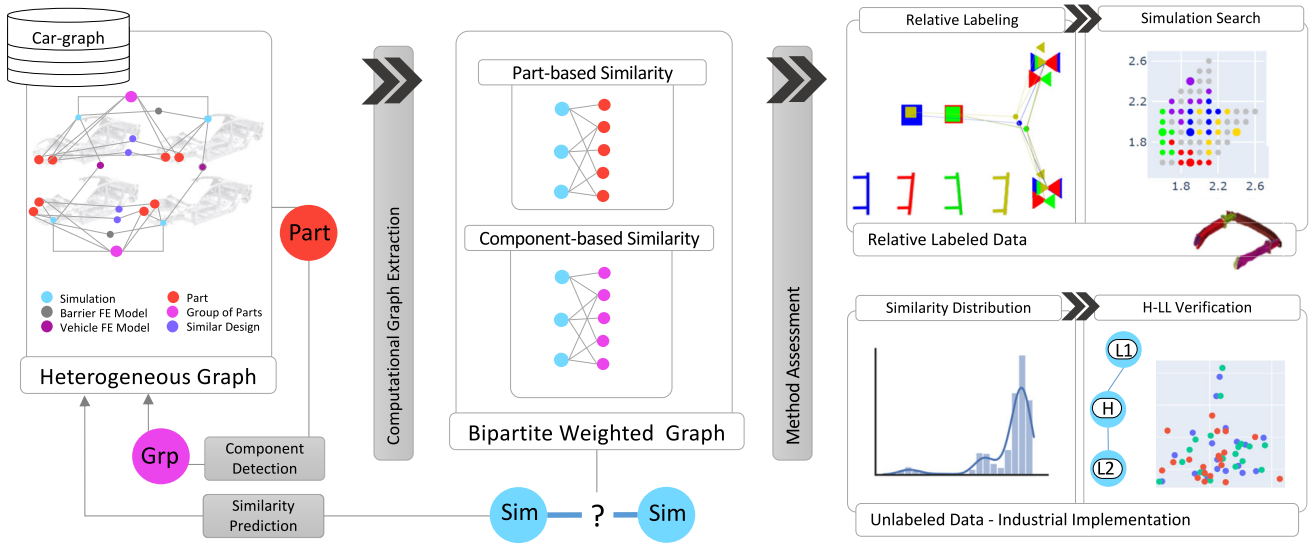


Fig. 1 Similarity prediction workflow for relative labeled and unlabeled data. This workflow considers parts and part groups (components) and predicts the similarity based on two variants of a bipartite graph. The nodes' coloring in the heterogeneous graph reflects different node types.

2 Related work

To the best of our knowledge, there are no existing methods that specifically address the searchability of simulations. Similarly, no methods exist to use graph algorithms, e.g., on car-graph, to predict similarity. Consequently, we outline other available methods for assessing simulations' similarities and graph algorithms for link prediction.

2.1 Simulation similarities

Dimensionality reduction methods are one of the popular techniques in machine learning (ML) for comparing and getting an overview of data [15]. In crashworthiness, these methods have been studied since 2008 [1] for exploration and cluster identification [17] of FE simulations. Note that one can consider the distance of the embeddings as a similarity measure of the simulations. However, the focus of the available research has been on outlier detection or behavior classification [11, 14].

Studies using dimensionality reduction usually look into deformations of the FE model [10, 11, 14, 17, 24]. The challenge with these methods is their computational cost and sensitivity to capture local differences compared with global deformation. For example, these methods focus on realizing an occurrence of a buckling, a global deformation, in comparison to characterizing the buckling mode, e.g., its timing, a local feature. Furthermore, the integration of these methods into the OEM's workflow has been limited [24], despite the long period of time that these methods have been available.

While these embedding approaches can be used to obtain similarities between simulations using the same part in two simulations, a systematic way to compare and rank simulations using embeddings of several parts was so far not achieved. Here, one challenge is that the selection of the main features among the generated embedding vectors is inconsistent [11, 14]. This apparent lack of consistency makes it so far impossible to select a comparable representation for a different configuration of simulations, or a different set of parts. Even if all these challenges are overcome, the question of how to effectively use similarity measures to rank simulations from an engineering perspective remains unresolved.

Therefore, consideration of more scalable methods and other input measures is beneficial. One example for crash simulations is the FE solver outputs of energy absorption that gives internal energy (IE) per part over time, a so-called energy curve. Studies show that energy absorption characteristics enable quantifying component performance for the design of experiment (DOE) feedback in optimization studies [5, 6]. However, to our knowledge, there is no research on using features generic to the problem, such as energy features, to calculate the similarity of simulations. Another possibility is to use key performance indicators such as firewall intrusion or occupant injury criterion, but these reflect behavior on a much coarser level, which limits their analytical capabilities.

In [19], we investigated features derived from energy curves and studied their capability for summarizing the differences between the simulations. These features have enough resolution to identify the differences of simulations, and the number of parts included plays a role in localizing the differences. As a result, the differences are more global if more parts are considered compared to including a limited number of parts. Compared to the deformation-based approaches, the main benefit is that these features detect local and global differences. An additional advantage is the computation efficiency that supports more dynamic user interactions.

Another aspect of similarity assessment is the selection of FE entities for comparison, e.g., node, element, or part. Due to the modeling techniques, most of the FE entities are smaller than a component of the vehicle. Here, a component refers to a group of parts whose structural functionality depends on its parts, e.g., two welded plates of a crash-box, where each plate alone has much less axial stiffness than the welded ones together. Therefore, components can be seen as more physically meaningful.

Note that the grouping information is available in the computer-aided design (CAD) stage of development. However, this data is lost in today's workflow when generating a FE model from CAD information. Detecting these components automatically from a FE model is challenging. In [9], semantics are introduced for FE entities that enable identifying part splits during the development, but the grouping of FE entities from a structural aspect is not addressed. Consequently, we introduce a method to automatically detect the grouping of the FE entities.

2.2 Link prediction

Identifying similar objects based on the link structure in a graph is a fundamental operation in various domains such as web mining, social network analysis, and spam detection [25]. Considering our unlabeled data and car-graph characteristic, a weighted heterogeneous graph, there are limited methods available for link prediction.

The term learning to rank is used to describe supervised machine learning methods that construct ranking models for information retrieval systems. Here labelled training data is used to learn models minimizing suitable loss functions [16]. As observed, due to the complex engineering tasks and typically multi-criterial evaluations, there is no labelled or ranked training data available, which prevents the use of these supervised approaches.

Recent advancements in unsupervised methods for link prediction in weighted heterogeneous graphs have demonstrated

substantial progress. Methods such as Graph Autoencoders (GAEs) and their variants have been extensively employed to capture the complex relationships within these graphs [20]. While adaptations like Heterogeneous Hypergraph Variational Autoencoder (HeteHG-VAE) exist [8], due to the intricate structures and numerous parameters, HeteHG-VAE can be prone to overfitting, especially when the amount of training data is limited. This overfitting can result in poor generalization to unseen data.

Additionally, matrix factorization approaches, such as Weighted Non-negative Matrix Factorization (WNMF), have been adapted to leverage node and edge heterogeneity effective [13]. However, these methods often struggle to capture the complex structural information inherent in heterogeneous graphs and loose structural information. Recent studies also explore leveraging random walk-based techniques, like node2vec, modified to accommodate edge weights and heterogeneity [13], enhancing the embedding quality for better link prediction. However, it may still miss capturing complex structural dependencies and higher-order proximities inherent in heterogeneous graphs.

Amid the existing similarity approaches, SimRank [12] has emerged as a powerful tool for assessing structural similarities between two objects. Similar to the well-known PageRank [3], SimRank scores depend merely on the link structure, independent of the textual content of objects. The major difference between the two methods is the scoring mechanism. PageRank assigns an authority weight for each object, whereas SimRank assigns a similarity score between two objects.

The Car-graph is relatively small compared to typical social networks, with fewer than 1000 nodes as opposed to tens of thousands. This smaller dataset limits the effectiveness of other methods, and the presence of multiple semantics adds to the challenge of model generalization. Therefore, we simplify the graph to a weighted bipartite graph with two types of nodes and use the widely recognized SimRank method [12] and its variant SimRank++ [2].

3 Simulations setups

Simulation similarity assessment is a new concept within CAE for which no public benchmark data is available. Using a submodel based on the Yaris FE model from CCSA [4], we generate simulation data that allows easy relative labeling of the crash behavior, which is the first benchmark data in the domain[1]. Figure 2a shows the selected components of the submodel, where the main components are the front bumper beam, crash-boxes, and side-members.

[1] The simulations and databases are available at: github.com/Fraunhofer-SCAI/GAE-vehicle-safety

In this dataset, each simulation includes 28 parts. Originally the right-hand side (RHS) and left-hand side (LHS) were asymmetric regarding the xz plane. Therefore, the FE model is modified to make it symmetric. The specific changes consist of removing the toe hook from the RHS, Fig. 2c, and making the bumper beam, crash-box and side-member reinforcement symmetrical. To achieve a slight deformation also in the side-members, the side-members end is constrained in x displacement and the moment along the x axis; and a total of 500 kg is added to increase the kinetic energy, Fig. 2b.

Our study consists of 66 simulations of a full-frontal impact against a rigid wall with a speed of 56.3 km/h. These simulations have around $30-40\%$ increase in total internal energy compared to the complete FE-model. Furthermore, the simulation setups for the submodel have minor design changes that are replicating real development process changes. Consequently, the differences in the outcome of the simulation are in the scale of CAE development processes. The simulations vary in crash-box plate thicknesses, where the crash-box is built of two sheet metal thicknesses, Fig. 2d. For all simulations, the crash-box outer plate thickness, T_1, is dependent on the inner plate thickness, T_2, by

$$T_2 - T_1 = 0.6. \tag{1}$$

Here, T_2 increases from its minimum value in equidistant steps of 0.1 $[mm]$. These variations are implemented equally on RHS and LHS. In a symmetric setup, T_2 of RHS and LHS increase equally in thickness, while for an asymmetric setup RHS and LHS thicknesses increase unsymmetrically. Figure 3a shows the employed distribution of the T_2 thickness value for LHS and RHS of crash-boxes for 66 simulations.

These changes cause asymmetrical and symmetrical absorption, which results in three crash modes for the deformation, Fig. 4. The crash mode indicates the yaw angle of the bumper beam. For this symmetric load-case a symmetric structural stiffness results in a yaw angle of zero. In Fig. 3a the simulations with equal thicknesses on LHS and RHS are on the diagonal. For simulations below the diagonal, the LHS is stiffer, causing the crash mode to have a negative yaw $-v_z$. For those above, the stiffer RHS causes a positive yaw $+v_z$ for the crash mode.

Among these 66 simulations, we pick five simulations as reference simulations for the investigation in Section 6. Figure 3b summarizes the crash modes for the selected reference simulations, including one zero mode simulation and two simulations for each negative and positive mode. Simulation three is the base model with zero crash mode. The negative and positive modes have mirrored thicknesses, i.e. 30 with 31 and 60 with 61. They also have different stiffness ratios, $T_{2_{RHS}}/T_{2_{LHS}}$, i.e. 60-61 is stiffer than 30-31. The reference simulations will later be used to find the most similar ones, see Section 7.

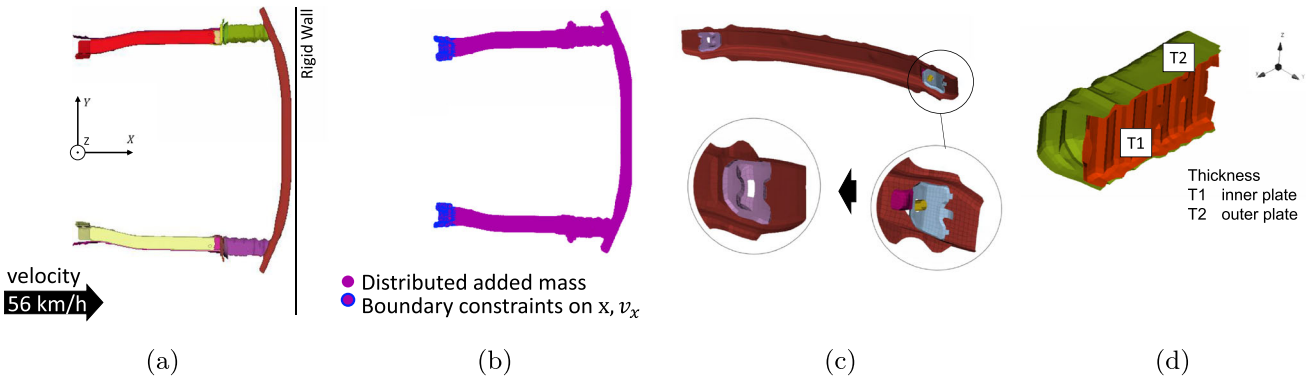
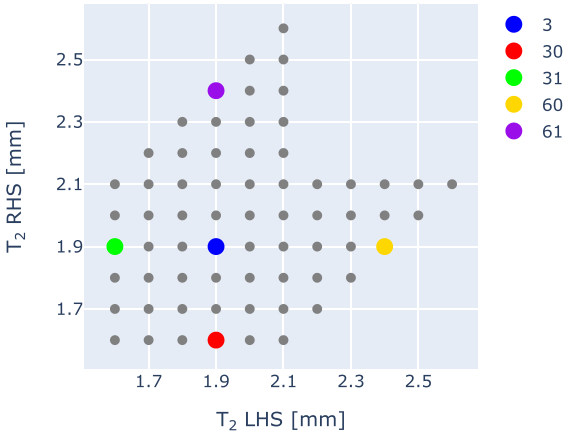


Fig. 2 FE-sub-model setup, (a) top view of the included components with applying symmetry on crash-box, side-member and bumper beam, (b) add boundary conditions and mass, (c) removing toe hook from the bumper beam. (d) crash-box thickness parameters.

4 Simulation similarity prediction

SimRank is an approach that is applicable in any domain with object-to-object relationships. It measures the similarity of the structural context in which objects occur, based on their relationships with other objects. Effectively, it computes a measure that says "two objects are similar if they are related to similar objects" [12]. As the base case, SimRank considers an object maximally similar to itself, to which we can assign a similarity score of 1.

For crash simulations, we could say that two simulations are similar if the energy absorption of the most energetic parts of the vehicle are similar to each other. This definition has two parts: first, the similarity of the parts associated with each simulation, and second, the similarity of the amount of energy absorbed. Consequently, formulating the crash simulation as a bipartite graph, simulation and included parts, requires evaluating the similarity of the parts referenced by each simulation and using suitable features reflecting the energy absorption. This emphasizes the need to look for something more than what the objective of the SimRank method provides. Note, we use the energy features introduced in [19].



(a) Spread of thickness T_2 of crash-boxes RHS vs LHS.

Crash Modes					
Ref. Simulations	3	30	31	60	61

(b) Crash modes for the reference simulations.

Fig. 3 Simulation setup consisting of 66 simulations (a), where five are chosen as reference simulations (b). Colored points in (c) are the reference simulations in (d).

4.1 Basic SimRank

Considering the basic SimRank, the similarity $s(a, b) \in [0, 1]$ between objects a and b is defined by a recursive equation. If $a = b$ then $s(a, b)$ is defined to be 1, otherwise,

$$s(a,b) = \frac{C}{|E(a)||E(b)|} \sum_{i \in E(a)} \sum_{j \in E(b)} s(i,j), \qquad (2)$$

where the set $E(a)$ contains the edges of node a, and C is a constant between 0 and 1. C gives the rate of decay, since $C < 1$, as similarity flows across edges [12], and we use $C = 0.8$. A solution to SimRank equations is reached by iteration to the fixed point [12].

4.2 SimRank++

In [2], it was shown that SimRank scores are not intuitively correct for complete bipartite graphs[2] and the authors intro-

[2] Note that, a complete bipartite graph is a bipartite graph, where every vertex of the first node-set connects to every vertex of the second node-set.

duced SimRank++, a so-called evidence-based SimRank. Additionally, SimRank++ uses edge weights and the so-called spread to achieve similarity scores consistent with the weights of the graph. Note that in our application, we typically obtain a complete bipartite graph and have edge weights.

In particular, [2] introduces the notion of evidence of similarity between nodes a and b

$$evidence(a,b) := e_{a,b} := \sum_{i=1}^{|E(a)\cap E(b)|} \frac{1}{2^i} \tag{3}$$

as an increasing function in the number of common neighbors. By using evidence in the score definition, the similarity between two nodes increases with the number of common neighbors [2].

Furthermore, to include the edge weights in the similarity calculations, normalization and scaling according to the local variance is included in SimRank++ [2]. This modification is intended to distinguish between the importance of edges with a small weight and those with a large weight. Specifically, the weights W are defined as

$$W_{a,i} = \underbrace{e^{-variance(i)}}_{spread(i)} \underbrace{\frac{w(a,i)}{\sum_{j\in E(a)} w(a,j)}}_{normalized\_weight(a,i)}, \tag{4}$$

where $variance(i)$ is the variance of the edge weights w of node i. All together, SimRank++ utilizes the number of common neighbors and the edge weights to define similarity scores iteratively by

$$s^{++}(a,b) = e_{a,b} \cdot C \sum_{i\in E(a)} \sum_{j\in E(b)} W_{a,i} W_{b,j} s^{++}(i,j).$$

Computationally, SimRank++ uses iterative methods similar to SimRank, however it has additional updates [2].

4.3 SimRankTarget++

We observe that the normalization (4) in SimRank++ implicitly imposes a direction on the edges even though the graph is undirected. Note that in a bipartite graph, a pair of nodes $(v,w) \in V \times V$ is associated with each edge $e \in E$; v belongs to set A (source) and w belongs to set B (target), where V is a list of nodes and E is a list of edges in a graph. While SimRank++ performs its normalization using edges that have a common node in the set A, we propose SimRankTarget++ (s^{++}_{trgt}), where we normalize the weights according to a common node in the set B. As a result, s^{++}_{trgt} does the normalization based on the target nodes instead of the original SimRank++, which does it based on the source nodes. We

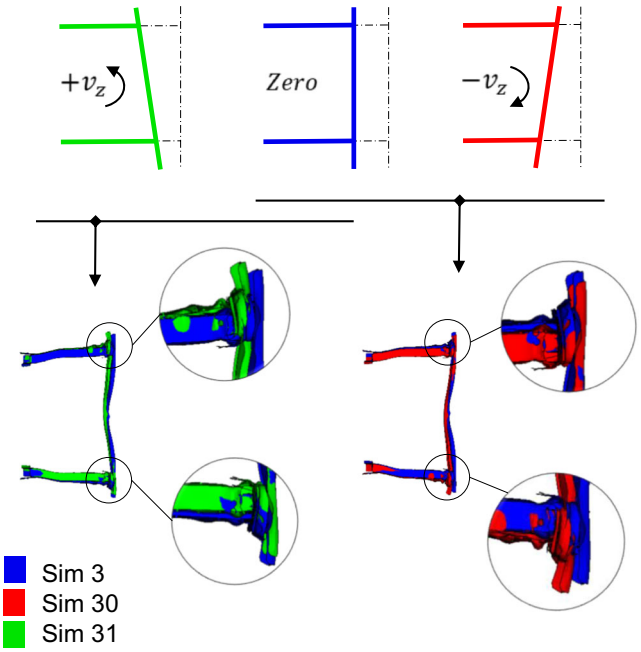


Fig. 4 Defined relative label for crash mode based on Yaw axis rotation. FE simulations result follows the color coding of the crash modes.

believe that how the weights are to be normalized depends on the physical meaning of the source and target.

For crash simulations, this means normalizing each part's energy absorption characteristics with similar parts of other simulations, set B, instead of parts related to one simulation, set A. In other words, with target normalization we consider the energy distribution of each part, while with source normalization we consider the distribution of parts of a vehicle. As in similar load-cases, the total amount of IE is almost the same for all analyses[3], the effect of the parts is expected to disappear. We will discuss this further in Section 7.2.

Consequently, we introduce Q, where we normalize the edges concerning the target nodes

$$Q_{a,i} = \underbrace{e^{-variance(i)}}_{spread(i)} \underbrace{\frac{w(a,i)}{\sum_{j\in E(i)} w(j,i)}}_{normalized\_weight(a,i)}, \tag{5}$$

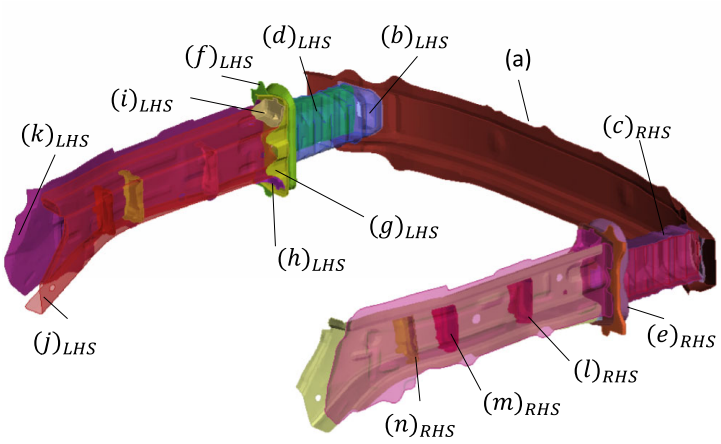
and using Q we define iteratively

$$s^{++}_{trgt}(a,b) = e_{a,b} \cdot C \sum_{i\in E(a)} \sum_{j\in E(b)} Q_{a,i} Q_{b,j} s^{++}_{trgt}(i,j).$$

4.4 Matrix formulation

Note that one can reformulate the SimRank problem using matrices. Given a graph one can define the column normal-

[3] Minor design changes will cause variations in the weight of the vehicle that will change the total IE of the impact, but these changes are minor.



(A) bumper beam	
	(a) bumper beam
	(b) frame front cap
(B) crash-box	
	(c) crash-box inner
	(d) crash-box outer
(C) connector plate 1	(e) connector plate 1
(D) connector plate 2	(f) connector plate 2
(E) connector plate 3	
	(g) connector plate 3
	(h) reinforcement 1
	(i) reinforcement 2
(F) side-member	
	(j) side-member inner
	(k) side-member outer
	(l) side-member reinforcement 1
	(m) side-member reinforcement 2
	(n) side-member reinforcement 3

Fig. 5 Grouped component for the FE sub-model, 27 parts and eleven corresponding components. In the table, uppercase letters are referring to the component and lowercase letters to the parts. In the figure, only one of the symmetric parts is marked and correspondingly subscripted with either LHS or RHS.

ized adjacency matrix U:

$$U_{i,j} = \begin{cases} 1/|E(j)| & \text{if } i \in E(j), \\ 0 & \text{otherwise} \end{cases}. \tag{6}$$

The SimRank matrix S is then

$$S = \max\{(C \cdot U^T S U), I\},$$

where I is a $|V| \times |V|$ identity matrix, and the maximum is to be understood entry-wise.

Now define, here assuming an undirected graph with symmetric weights for simplicity,

$$d(i) = \sum_{j \in E(i)} w(i,j) = \sum_{j \in E(i)} w(j,i).$$

We observe $d(i) = |E(i)|$ when $w(i,j) = w(j,i) = 1$ for $j \in E(i)$, and 0 otherwise, so we can write (6) as $U_{i,j} = w(i,j)/d(j)$.

If we consider (4) and (5) without the spread, which is the same for both cases, we can write in (4) $W_{a,i} = w(a,i)/d(a)$ and in (5) $Q_{a,i} = w(a,i)/d(i)$. The former is a row normalized adjacency matrix, while the latter a column normalized adjacency matrix, as it is used for the original SimRank matrix formulation.

5 Component detection

FE-modeling techniques require arranging vehicle components into several parts, e.g., due to material and thickness differences[4]. Consequently, FE-solvers output result quantities per defined parts. With components, we refer to a group of parts that in their structural analysis from a CAE analysis perspective are highly dependent. For example, the stiffness of the side-member component, (F) in Fig. 5, depends on three reinforcement plates as well as the inner and outer side-members.

One application of using components is in selecting the most essential parts for ML pipelines. In [19], we showed that using the maximum of the internal energy is capable to filter the essential parts. However, this approach sometimes excludes smaller parts that are of interest from crashworthiness aspects, e.g., the reinforcement plates, (l), (m), (n) in Fig. 5.

Due to the existing split into parts the post-processing of the results per component becomes challenging. Since the parts connectivities, which describe a component, are unavailable, it is vital to develop a method for component detection to facilitate post-processing. Therefore, we propose a method for component detection, verify its result for the illustrative example, and discuss its scalability.

Component detection outputs lists of parts of an FE simulation. The parts in a list describe a component, whose functionality in a load case should be evaluated jointly for the parts forming the component. In this way, it allows us to design bipartite graphs that consider a group of parts instead of a single part, see Section 7.1. In particular, the functionality of entities is considered in order to bundle them. Conse-

[4] For further background on crashworthiness, see e.g. [7].

quently, this section also discusses ways to evaluate energy properties for components.

5.1 Component detection method

There are several possibilities for component detection. One option is to preserve this information from CAD to CAE by introducing a corresponding workflow within the company. However, this option is for now not feasible due to the involved process dependencies that are time-consuming to establish in big OEMs.

A second approach is to develop an interpreter for the specific FE solver that transfers each connection type to a generic connection for component buildup. In [18], we partially employed this method to identify connection changes in the model. The limitations of this approach are: a) time-consuming development due to the dependencies on the specific solver and b) the need to use several modeling representations for different types of connections.

Additionally, recent computational power allowed FE-models to include more details. As a result, connectivities, e.g., bolts and clips, are modeled with generic FE entities, e.g., shell solid elements, instead of a solver-specific abstraction for connections. Therefore, developing the interpreter would be even more complicated. Moreover, both outlined approaches deliver several connections per pair of parts due to assembly requirements, e.g., several bolting or welding. The high number of connections requires additional filtering to distinguish a component's assembly from a component-to-component connection. Thus, a more automatic method is beneficial.

Therefore, we develop a geometrical search method that detects components. We consider each part in the vehicle as a box and then group highly overlapped boxes. The geometrical features of the parts define the box, including length, width, and height, along with the coordinate system of the FE-model (L-x, W-y, H-z). In grouping these boxes, we make the following procedural decisions

- Include specific entities from the FE-model[5].
- Define FE-modeling guidelines to differentiate parts from connections[6].
- Decide on a box merge in a pairwise comparison.
- Define batches for pairwise comparison via two-dimen sional (2D) k-means clustering[7] to reduce computational time.
- Consider two-stages in merging: complete and partial overlap. Complete overlap is the scenario, where a smaller box (child) is located entirely in a bigger box (parent). By contrast, partial overlap refers to situations where boxes are not completely overlapped.
- Skip merges in the direction of the impact/loading for partial overlapping to capture the load path.

We start the investigation on the FE-sub-model presented as an illustrative example, in Section 3. This model includes 28 parts, and 27 parts match the prescribed entity selection. From the crash analysis engineering view, this model contains eleven components: (A) bumper beam, (B) crash-boxes on right hand side (RHS) and left hand side (LHS), (C)(D)(E) connector plates on RHS and LHS, and (F) side-members on RHS and LHS. Thus, the intended outcome is eleven components. Connector plates between crash boxes and side members could be one component. However, as mentioned, in our grouping procedure, we prevent merging boxes with overlaps in the direction of the impact/loading.

For grouping boxes, we compare boxes pairwise. Pairwise comparison of all parts for the complete FE-model is computationally expensive. Therefore, we use k-means clustering[7] on the boxes' centre of gravity (COG). Pairwise comparison takes 346 seconds for the complete model of Yaris with 728 initial parts, and with k-means clustering, we reduce the time to 23 seconds. In this way, the clustering of the boxes creates batches to reduce the number of pairwise comparisons. We consider the boxes COG distances in pairwise comparison. However, the three-dimensional (3D) distance of parts clusters the parts locally and will skip some desired merges.

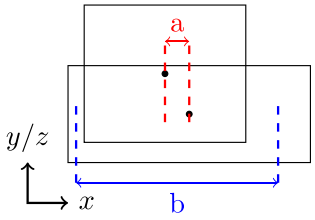
For example, the bumper beam component in Fig. 5 (A) will encounter this issue since the bumper beam, part (a), and the RHS and LHS frame front cap, part (b), have a significant COG distance in 3D space. In this example, the 3D distance will cluster the frame front cap with the crash-box, not the bumper beam. Consequently, we consider the 2D space for clustering the boxes and assess each plane separately, top-view (xy), front-view (zy), and side-view (xz).

In this example, the bumper-beam and the frame front cap have an apparent distance in the top-view and front-view clusters, these boxes will not be compared. However, the boxes COG are close in the side-view, and will be in the same cluster to be assessed. As a result, we have an additional iteration loop to assure all required pairs are assessed, in

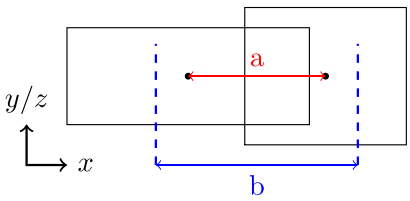
[5] Only shell elements since beam and solid elements usually represent the connections and have a single properties ID (PID) for all same type of connection in the model.

[6] Require null shell elements for components modeled with solid elements. null shell is a recommended method for better contact modeling, MAT_NULL in LS-DYNA.

[7] Using the implementation from, sklearn.cluster.KMeans python package.



(a) Complete overlap in impact direction x. It is not classified as being in impact direction due to $a/b < \alpha$ and will go to the next check for merging. Note that we here use a larger threshold α for illustration purposes.



(b) Partial overlap in impact direction x. It is classified as being in impact direction due to $a/b > \alpha$ and therefore not approved for a merge.

Fig. 6 Two examples of classifying the overlaps in direction of the impact with big and small ratios of a/b, subfigure (b) and (a) respectively.

each loop we are switching between 2D views to change the clustering. After several iterations, we compare all remaining parts pairwise to be sure all boxes have been compared.

The pairwise comparison investigates two scenarios: complete and partial overlap. Complete overlap is when a box is entirely inside a larger box, Fig. 6a. Here, the merged box takes over the dimensions of the larger box. For partial overlap, we use as the comparison quantity

$$\frac{\overbrace{|c_{cog_{b_1}} - c_{cog_{b_2}}|}^{a}}{\underbrace{(L_{c_{b_1}} + L_{c_{b_2}})/2}_{b}} \leq T, \tag{7}$$

i.e., the fraction of the COG distance and the average of the boxes' side lengths in the chosen direction.

$c_{cog_{b_1}}$ and $c_{cog_{b_2}}$ are the extracted 2D COG coordinates for the compared boxes, box one and two respectively. More specifically, the 2D coordinates are in the respective global axis direction: x, y, and z. The $L_{c_{b_1}}$ and $L_{c_{b_2}}$ are the box dimensions in the corresponding axis, for boxes one and two respectively. The dimension L_c is aligned with global axis: L_x, L_y, and L_z.

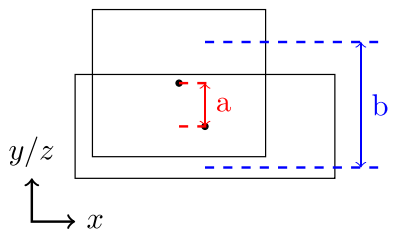


Fig. 7 Decision-making of box-merge in impact plane, yz.

The thresholding with (7) is used for two scenarios. First, we classify if the overlaps are in the impact direction and, second, we decide on overlap merging. Therefore, two different thresholds are applied for each scenario: α and β, respectively.

For impact direction classification, threshold α is set to a low value, e.g., 0.01, which prevents box merging if a/b is bigger than this value, Fig. 6b. Afterwards, if the overlap is not classified as being in the impact direction, the percentage of overlap is evaluated to decide on the merging of the boxes.

Algorithm 1 Box merge for impact direction x.

```
if a_x/b_x ≤ α and a_x > 0 then
  if a_y/b_y ≤ β and a_z/b_z ≤ β then
    merge boxes
  end if
end if
```

For the overlap check, the impact plane directions are assessed with the threshold β set close to one, e.g., 0.97, Fig. 6a. Here the boxes will merge for a large ratio of a/b. In a full-frontal impact in the x plane, the considered overlaps are only in the y and z plane, Fig. 7.

5.2 Component verification

The outcome of this method for the illustrative example matches the table in Fig. 5. Initially, 27 boxes represent the 27 parts and after merging eleven new boxes are generated that include the parts as in Fig. 5. Here we run the merging for the frontal impact that will skip the merges in the x-direction; see the coordinate system in Fig. 2a. Consequently, the connector plates even with the existing overlap in the x-direction are not merged, components C, D, and E.

Afterward, we implement this method on the full Yaris model to assess the scalability of this method. One obstacle in the full model application is the dominance of the exterior parts that is acting as a wrapper for a big proportion of the parts, e.g., the front facia and outer layer of the vehicle body. A solution for this is to exclude the exterior parts defined by the user. An additional option is to combine part filtering with grouping as,

- Filter the most energetic parts.
- Apply the component detection with the limited search for filtered energetic parts and their neighbors.
- Update the filtering with the most energetic components.

In this way, the focus will remain on the structural parts and exterior parts will not cause an issue.

5.3 Component features

In [19], we introduced energy features for the IE of each part with initial absorption time [t_i], maximum internal energy [IE_{max}], and end of absorption time [t_n]. Since these features are part-based, enabling the post-processing of the results at the component level requires an additional step to combine the features. Two approaches exist to generate component-based features. First, combining the extracted features of the grouped parts belonging to a component. Here, combining features refers to determining the minimum, maximum, sum or average of a group of features. The other alternative is summing up the energy curves of the parts before feature extraction.

For IE curves, the effect for IE_{max} is minor since IE saturates after the maximum. Consequently, the sum of IE_{max} for parts and max of IE summation curve differs only slightly. However, the time features t_i and t_n of IE curves deviate more between the two approaches. In curve summation, the early ramp-up or late saturation of the IE vanishes for the parts with the smaller energy due to the dominance of the more energetic parts. As a result, we propose using the combined features of parts for the time features, specifically the minimum value of t_i and the maximum value of t_n, while utilizing IE_{max} from the summation curve.

6 Energy diagram

We introduce an energy diagram to illustrate simulation behaviors in a crash simulation. For simplification, we have a 2D view using IE_{max} and t_n, where t_n contains the t_i feature and relates to absorption time, $\Delta t = t_n - t_i$. An additional benefit of t_n is that it is easier to understand visually than Δt for processing the sequence of behaviors, i.e., the part's relative behavior [19].

To build the energy diagram, we select the five most energetic parts for each simulation and add the mean of the energy features to the plot and connect the parts to it. Note that, considering the 28 parts included in each simulation of the illustrative example will make the visualization challenging. But, the five most energetic parts turn out to be the same for all simulations, including the four plates of the crash-box and the bumper beam.

6.1 Diagram examples

Figure 8 shows the energy diagram for the base simulation. Left and right directed arrows indicate the LHS and RHS parts of the crash box, respectively, where a square represents the bumper beam. The final energy diagram is obtained by connecting each part to a point reflecting the average of the energy features of the five parts.

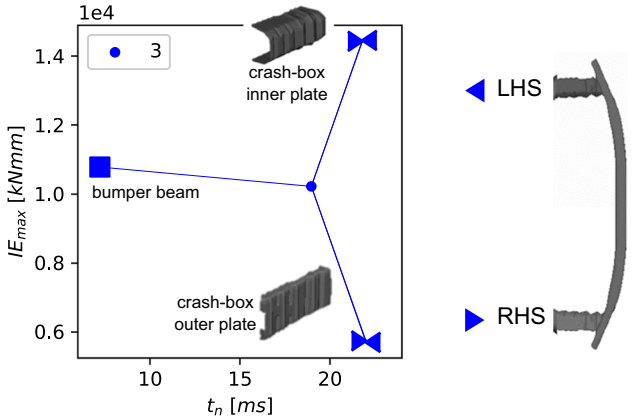


Fig. 8 Energy diagram for the base simulation, number 3 in Section 3, considering five parts.

Figure 9a displays the energy diagrams for simulations 30 and 31. These simulations have the same thickness value change but on opposite sides. As a result, the corresponding energy diagrams are essentially mirrored. Their structures look identical except for the switch between RHS/LHS, reflecting the change in producing negative or positive yaw.

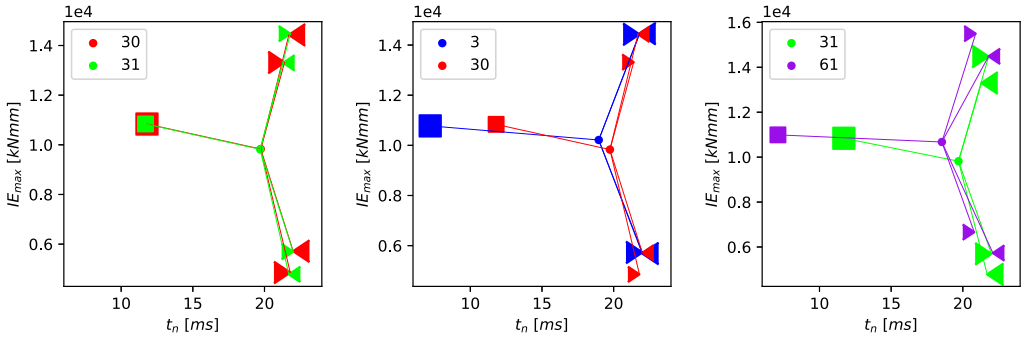
In Figure 9b we compare one of these mirrored simulations to the base model. We observe that the IE_{max} has decreased for the RHS crash-box plates, which is due to the stiffness reduction. However, in comparison, t_n has not changed. The reduction of IE_{max} while t_n is unchanged indicates a so-called stack-up state[8]. However, the average of the energy features shows lower IE_{max}. Therefore, the side-members are absorbing the remaining energy since the total IE should remain the same over all the parts in the simulation. Another noticeable observation is that the bumper beam absorption energy is independent of the crash-box, however, the t_n is dependent on it.

Figure 9c presents the energy diagrams for simulations 31 and 61, wherein both the RHS crash-box is stiffer than the LHS resulting in the negative yaw crash mode. We observe an offset in the diagrams, while the structures are similar since they reflect the similarity of the crash mode. The angle differences in each energy diagram correspond to the yaw angle. Note, the offset is due to more energy absorption in simulation 61, reflecting its higher thickness values.

6.2 Diagram similarities

Representing simulations as a diagram with energy features enables the comparison of simulations. The illustrative exam-

[8] Maximum possible deformation in a component.



(a) simulations 30 and 31 (b) simulations 3 and 30 (c) simulations 31 and 61

Fig. 9 Energy diagram for selected simulations from Section 3.

ple highlights two scenarios: a change in the structure of the diagram and an offset of the whole diagram. From an engineer's perspective, these two aspects can be considered as the crash mode and absorption factor.

A crash mode reflects the absorption of the parts relative to each other, represented by the structure of the diagram. On the other hand, one looks at how much energy is absorbed with the absorption factor. Thus, the absorption factor operates as an offset factor in the energy diagrams in this data representation. Consequently, the same crash mode, but different absorption factors exist if the relative stiffness of the components is similar.

With these visual investigations of similarity between simulations using energy diagrams, we have the following research question: how to implement these in graph analytic methods to estimate simulation similarities? Working with unsupervised learning methods on these energy diagrams would involve treating these as separate data objects and individual weighted graphs, an ongoing open research question [26]. Instead, we directly employ the energy features as weights in a graph to be able to use established methods for link prediction. In this way, we use the edge weights to detect slight differences between simulations as presented in the examples we discussed in Section 6.1.

7 Similarity results

In this section, we compare different SimRank methods for predicting the similarities of simulations. We start with a short description of our bipartite graph, both part-based and component-based, and its connection to our graph database [18]. Afterward, we evaluate the SimRank methods and compare them with the root mean square error (RMSE) of displacements and internal energy as a similarity baseline. We use two approaches for evaluating the similarity predictions:

- Comparing the relative labeled ranking. In this case, we order the five reference simulations from an engineering perspective and compare this ranking with one based on the computed similarities.
- Searching for similar simulations, where for the five reference simulations we find the most similar ones among the remaining 61 simulations.

Finally, we present results on an industrial crash simulation data set.

7.1 Bipartite graph

Let us provide a brief overview of our graph modeling for both part- and component-based bipartite graphs. Figure 10 shows the data schema that we here employ, which is part of the entire graph modeling we introduced in [18].

In this schema, a (Sim) node reflects a FE simulation outcome, where its properties stem from global entities of the simulation, e.g., total mass or impact energy. An FE-model contains many elements, where a group of elements with the same properties is identified as one part. Consequently, one simulation includes several parts, and we model it with a (Part) node as the main entity representing the simulation.

The edge ○— NRG_PART —○ relates (Part) nodes to (Sim) nodes and includes certain energy features of the parts as

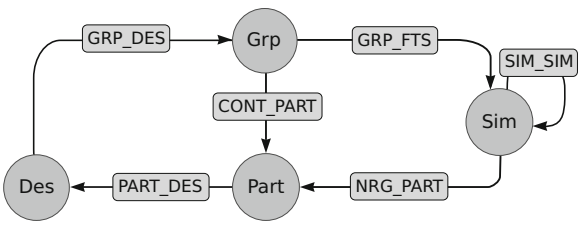


Fig. 10 Graph data schema used for simulations similarity prediction, full schema available in [18].

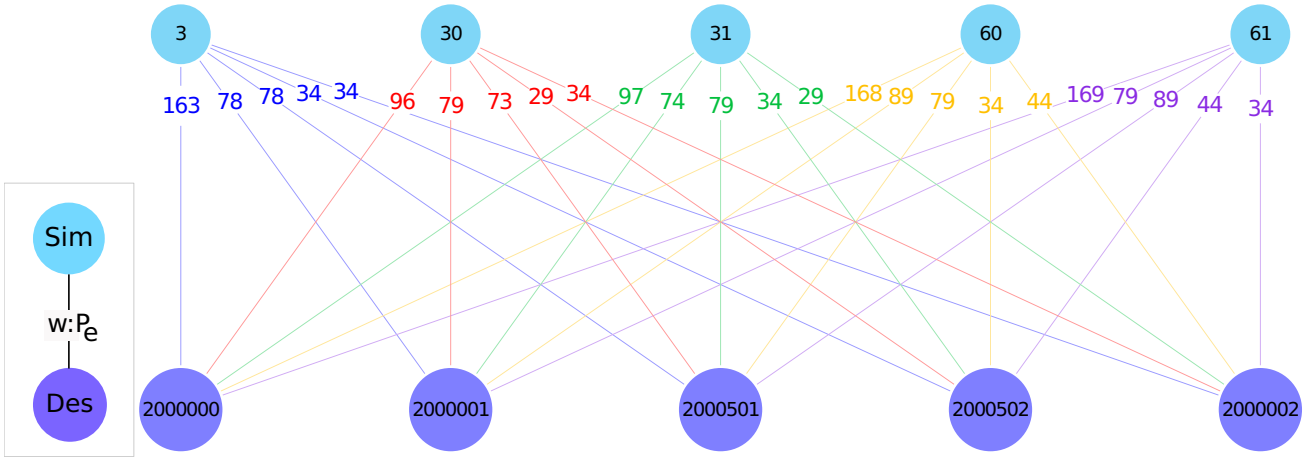


Fig. 11 Bipartite graph showing the edge weights P_e, measured in $[10^7 \cdot Nm/s]$. The edge colors represent the reference simulations discussed in section 3. This graph highlights the distribution of edge weights for each part versus each simulation, emphasizing the importance of the s^{++}_{trgt} normalization technique in analyzing and comparing simulation data.

weights. Design (Des) nodes bundles parts that have similar FE-model features. A (Grp) node is a group of different (Part) nodes in the database that reflect the outcome of our component detection method Section 5.

For simulations similarity prediction, we are looking to predict the ○– SIM_SIM –○ edge. Both bipartite graphs, part- and component-based, respectively, have with (Sim) and (Des) wo node types, see Fig. 11 for the part-based graph. Therefore, the similarity of connections between (Sim)-(Des) is used to predict a (Sim)-(Sim) connection.

We have two scenarios for structuring the bipartite graph with (Sim)– SIM_DES –(Des). First, for the part-based similarity, we follow (Sim)– NRG_PART –(Part)– PART_DES –(Des). Here, we obtain the edge weight from the energy features of the parts and predict the similarities of simulations based on the similarity in the energy absorption of the parts. Second, for the component-based similarity, we go along (Sim)– GRP_FTS –(Grp)– GRP_DES –(Des) and use the grou ped features as weights and predict the similarities of the simulations based on the components absorption similarities.

7.2 Part-based similarity

To study part-based similarity, we compare the results from different SimRank formulations with a defined relative labeled ranking for the Yaris simulations from Section 3, where we focus on five simulation pairs for ease of presentation. Table 2 summarizes the results. In this table, we order the columns according to the desired ranking based on engineering judgment as follows:

- Simulations (30)-(31) are the most similar due to the symmetric changes.
- Simulations (3)-(61) are the least similar since there is the most significant stiffness change among all simulations.
- The pairwise similarity of (30) or (31) to simulations (3) and (61) should be equal. Equality comes from symmetrical behavior that acts as a mirrored weight on nodes.
- Simulations (30) and (31) are more similar to (3) than (61) since the stiffness difference is less in (3)-(31)/(30) compared to (61)-(31)/(30). As a result, simulations (3)-(30) and (3)-(31) have the second-order ranking with equal values, and simulations (61)-(31) and (30)-(31) have the third-order ranking.

Table 1 Part names for PID in Fig. 11

PID		Part Name
2000000		bumper beam
LHS	RHS	
2000001	2000501	crash-box inner plate
2000002	2000502	crash-box outer plate

The used methods include SimRank (s), weighted SimRank (s_w), weighted SimRank with evidence ($s_{w,evd}$), weighted SimRank with evidence and spread (s^{++}), and weighted SimRank with evidence and spread that is normalized over target nodes (s^{++}_{trgt}). We employ the energy power absorption ($P_e = IE_{max}/\Delta t$) of the parts as the weight for the ○– SIM_DES –○ edges. Another alternative for the edge weight is the IE_{max}; however, P_e gives better results. In the study, we select the five most energetic parts, which are the bumper beam and two plates of the crash-box on LHS and RHS, Table 1.

Table 2 Similarity prediction from different methods when considering the five most energetic parts in the illustrative example, Fig. 11. The order of columns is the expected order as described in Section 7.2, ($C = 0.8$, $weight = P_e$)

Method	30-31	3-30	3-31	61-31	61-30	3-61
s	0.4444	0.4444	0.4444	0.4444	0.4444	0.4444
s_w	0.4749	0.4795	0.4798	0.4789	0.4783	0.4892
	6	3	2	4	5	1
$s_{w,evd}$	0.4379	0.4427	0.4430	0.4420	0.4414	0.4527
	6	3	2	4	5	1
s^{++}	0.2084	0.2104	0.2102	0.2267	0.2260	0.2295
	6	4	5	2	3	1
s^{++}_{trgt}	0.3119	0.2719	0.2717	0.2695	0.2695	0.2399
	1	2	3	4	5	6

Table 2 presents the (Sim)-(Sim) similarity predictions[9] for the illustrative example. Fig. 11 shows the five most energetic parts for the five simulations. This results in a fully bipartite graph, and disregarding weights means that two simulations are similar if the energetic parts are similar. Therefore, as expected from Section 4, SimRank predicts that all simulations are similar, which shows that the method is insufficient to evaluate the similarity between simulations with similar energetic parts but different absorption distributions.

With the P_e weight, the predicted similarities still differ from our expectations for s_w, $s_{w,evd}$ and s^{++}. However, the s^{++}_{trgt} method provides a result that reflects our relative labeling. Note that to observe an effect using the weight factors, they need to be scaled to be smaller than 2 (P_e scaled with 10^9 based on this model unit system, energy $[N - mm]$ and time $[s]$). If the spread is more expansive than two, then all similarities become zero, and if it is smaller than one, the result is similar to the weighted graph without spread.

It is interesting to further discuss the difference of s^{++}_{trgt} and s^{++} in this use case. Considering s^{++} and normalizing the edges regarding the sum of the edges in the source nodes refers to normalizing each part absorption with the total IE in a model, which is more or less constant for all the simulations when considering one load-case.

Alternatively, normalizing for edges in the target nodes, we are normalizing the edge weight with the total absorbed energy for that specific part in all simulations, which highlights the absorption efficiency of that part for each simulation. Consequently, the second approach is more relevant for comparing simulations since we can weigh the parts relative to each other instead of the first approach, which looks into the relation of the parts in one simulation.

[9] We modify the SimRank similarity calculation in the NetworkX Python package to evidence-based SimRank with spread consideration.

To further study the performance of the proposed algorithm, we compare with rankings based on similarity or distance measures for simulations. A common approach is to assess differences in the simulations is by looking at mesh-based function differences, e.g., [10, 11]. Therefore, we use the differences in the displacement between the simulations and use the RMSE as the distance.

Table 3 summarizes the RSME of the displacements and the corresponding ranking for the illustrative example in three scenarios: all parts at the last time step t_{max}, all parts in all the time steps, and the five most energetic parts in the last time step. From Table 3 we gather that none of three approaches can capture the expected crash behavior in the order prescribed previously. The top three and the last three similarities are invariant. However, there is a different order within each cluster. Differences in the ranking show that this method is time-step dependent.

In addition, the meshes of the parts should be the same for this method to be able to evaluate the RMSE. Further, looking at all time steps is computationally expensive, and the time sequence of events can lead to high differences while the final crash mode is similar, e.g., stack-up situations. While looking at only the last step would solve this issue, but the sequence of events will still be missing in the similarity calculation.

Another approach is to evaluate the RMSE for the internal energy of the parts with more global features than displacement, see the last row in Table 3. Here for simulation pairs, P_e are compared for the five most energetic parts. The main difference is the ranking of (3)-(61) as the second.

So far, we have investigated different configurations of the SimRank++ method for similarity prediction between simulations. An additional hyperparameter to evaluate is the number of employed parts from each simulation that are used in the bipartite graph. Table 4 summarizes s^{++}_{trgt} prediction for 2, 5, 15, and 28 (all) parts being considered. The order of the predicted similarity between simulations has the expected pattern for the relative labeled data upwards from including five parts. Noteworthy, the similarity score spread declines when including more parts.

7.3 Component-based similarity

Initially, we constructed a bipartite graph based on the FE parts to predict the similarity of simulations. We now consider similarity predictions based on the components, where we use the outcome of the component detection presented in Section 5. Table 4 summarizes the prediction result of s^{++}_{trgt} for the component-based bipartite graph while increasing the number of most energetic components included. This evaluation requires at least three components to fulfill the previous section's pre-defined ranking. Parts included in these three components are similar to the five parts in the similarity pre-

Table 3 RMSE for the difference of displacement ($Displ\ [mm]$) and P_e ($[MNm/s]$) in each pair of simulations, considering a different number of parts and time steps in the illustrative example. The even rows show the ranking of the distances. The order of columns is the expected order described in Section 7.2

Scalar	Time Step	Parts	30-31	3-30	3-31	61-31	61-30	3-61
	t_{max}	All	5.55	7.60	7.82	21.16	21.96	15.25
			1	2	3	5	6	4
$Displ$	t_{all}	All	7.97	7.02	7.09	17.35	20.45	13.82
			3	1	2	5	6	4
	t_{max}	Five	5.92	4.74	4.67	9.06	9.83	6.03
		energetic	3	2	1	5	6	4
P_e	t_n	All	4.5	299.1	295.1	327.5	331.3	69.5
			1	4	3	5	6	2

diction of Table 4. However, using components, the predicted similarities have higher values.

Additionally, for the component-based similarity, the maximum range of the spread is achieved by including five components, 15 parts, whereas for part-based similarity, it is with five parts. Note that the 15 parts involved in component-based similarity are not equivalent to 15 parts in part-based similarity. Six parts are the side-member reinforcement plates when filtering with components and these are not selected as the most energetic parts when using parts. Consequently, component-based similarity performs adequately with a more stable result; however, the part-based similarity is more sensitive Table 5.

Moreover, the similarity range drops less with increasing the number of components than with the number of parts. If we compare the full model prediction, 28 parts for the part-based compared with eleven components, the component-based has a broader range than the part-based. Overall component-based similarity shows better results in this use case.

7.4 Searching simulations

In this section, we use the similarity prediction methods to search for simulations similar to the five reference simulations. First, we compare the capabilities of the s^{++}_{trgt} and RMSE of P_e methods as a search tool. In Fig. 12 we visualize for each of the reference simulations from Section 3 the corresponding top seven similar simulations.

We expect to have the diagonal points, $x = y$, similar to simulation three with zero modes. This is because the points have the same relativity of the thickness of the crash-box plates. Likewise, the simulations under $x = y$ should be similar to simulations 30 and 60 and mode $+v_z$ based on their stiffness range, while the ones above should have mode $-v_z$.

Figure 12a shows that the s^{++}_{trgt} method achieves the expected clustering, e.g., the modes stay on one side of the diagonal. On the other hand, Fig. 12b shows that the RMSE of P_e method fails to recognize the crash modes as the colored points for simulations 30, 31, 60, and 61 are on both sides of the diagonal. This result shows that this method primarily works by averaging the energy absorption of the parts.

Next, we compare part-based and component-based results. Figure 13 visualizes the comparison of the two methods while increasing the number of target nodes in the bipartite graph, (Part) and (Grp) for part-based and component-based methods, respectively. This comparison also highlights the deviation between the two methods by increasing the number of parts.

Additionally, the color coding for the zero mode deformation, blue scatter points, is captured the best for the minimum included target nodes, Fig. 13a, e. This observation emphasizes that finding a similar simulation differs from ordering the similarities. For finding the most similar simulation, including the least number of target nodes or part-based assessment, perform satisfactorily. However, from the robustness aspect and the ranking of the prediction, the component-based is more promising.

7.5 Industrial application

After investigating the SimRank++ method for the illustrative example, we apply the approach to data from CEVT, using the

Table 4 Part-based s^{++}_{trgt} similarity prediction deviation regarding changing the number of energetic parts included in the illustrative example, Fig. 11, ($C = 0.8, weight = P_e$)

No. Parts	30-31	3-30	3-31	61-31	61-30	3-61	Range
2	0.4239	0.4243	0.4243	0.4235	0.4228	0.4234	0.0016
5	0.3119	0.2719	0.2717	0.2695	0.2695	0.2399	0.0719
15	0.2763	0.2568	0.2565	0.2518	0.2517	0.2393	0.0370
28	0.3086	0.2976	0.2973	0.2947	0.2945	0.2902	0.0184

Table 5 Component-based s^{++}_{trgt} similarity prediction deviation regarding changing the number of energetic components included in the illustrative example, ($C = 0.8$, $weight = P_e$)

No. Parts	30-31	3-30	3-31	61-31	61-30	3-61	Range
2	0.4198	0.4211	0.4212	0.4180	0.4166	0.4182	0.0046
3	0.4125	0.4071	0.4071	0.4009	0.4003	0.3967	0.0157
5	0.2440	0.2260	0.2262	0.1997	0.1990	0.1858	0.0583
11	0.2517	0.2394	0.2395	0.2266	0.2260	0.2166	0.0351

so far best-performing configuration, i.e., SimRankTarget++. This real-life industrial data includes a total of 611 simulations of three different load-cases from frontal impact (ffo: full front overload, foU: front oblique overlap, and foI: front small overlap) in four successive development stages (primary, early, middle, and late)[10]. For confidentially reasons we cannot give details about the simulations, but the presented results still show the capability and potential application of the proposed analysis procedure for real development data.

For this data, the bipartite graph is part-based with as before energy power as the weight factor, $P_e = IE_{max}/\Delta t$. The similarity prediction considers each load-case for each development stage separately for a specified number of parts, i.e, 20. The load-case separation of simulations is due to the use-case that similarity between different load-cases is usually out of interest. Furthermore, the grouping of development stages is due to PID changes between development stages to avoid connecting two irrelevant parts. The necessary parts mainly vary between load-cases and slightly among developmental stages.

Here we present an overview of these results and deep dive into the result for one load-case in a single development stage. The industrial data is unlabeled, so it is challenging to assess the result of the similarity predictions. To tackle this, we introduce two approaches. First, we visualize the similarity prediction result using a histogram and a kernel density estimation (KDE)[11]. Second, we select specific simulation pairs in each batch to further analyse the similarity prediction, H-LL simulations. We present these approaches for the foI load-case in the primary development stage.

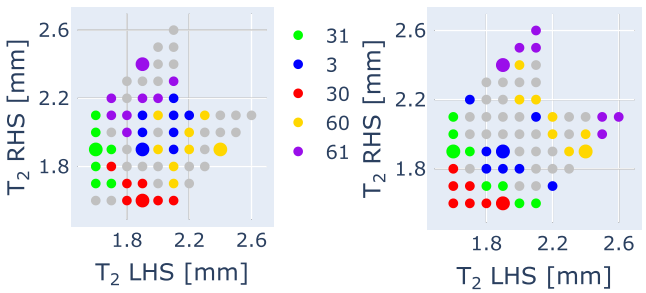
7.5.1 Similarity density

The similarity prediction score depends on the selection of simulations in the batch and the number of included designs. We use the KDE to visualize the probability of the data being in a given range in the area under the density curve. First, we focus on the similarity predictions for the foI load-case in the primary development stage when considering the 20 most energetic parts, Fig. 14. The total number of simulations is 115; consequently, the number of similarities pairs is 6555. The horizontal axis refers to the predicted similarity score, whereas the vertical axis reflects the number of simulation pairs for each value.

A noticeable outcome is that the density of similarity prediction shapes clusters of simulations. In this way, density clustering detects the groups of simulations with similar scores of similarities. Figure 14 has three clusters at (a), (b), and (c), where (a) includes mainly the outliers.

Based on that, we claim that the cluster with only low predicted similarities includes simulations that are outliers, which allows anomaly detection. To confirm, for the considered foI load-case during the primary stage, we manually identified and relatively labeled simulation runs with early termination due to errors or unrealistic high internal energy, respectively. Afterward, we evaluated the similarity score with and without these simulations. We observed that the low-range similarity cluster, i.e., zone (a) in Fig. 14, is removed from the density plot when excluding the outlier simulations. The corresponding similarity distribution is plotted in Fig. 15a.

Figure 15 presents the similarity distribution for the different load-cases in different development stages. Here, we analyse each load-case in separate development stages. These KDE plots highlight the differences between each batch, e.g., the score range and the number of peaks. To exclude outliers, each KDE plot is without simulations with a maximal similarity of less than 0.2. We observe that 0.2 seems low for some batches to disregard the outliers, e.g., Fig. 15a and 15c. Nonetheless, SimRankTarget++ low computational cost, less



(a) s^{++}_{trgt} (b) RMSE of P_e.

Fig. 12 Most similar simulations to the reference simulations according to two similarity estimations. The used color code of simulations is from Section 3.

[10] Detailed data description in Section 3 of [19].

[11] seaborn.distplot python package with KDE=True

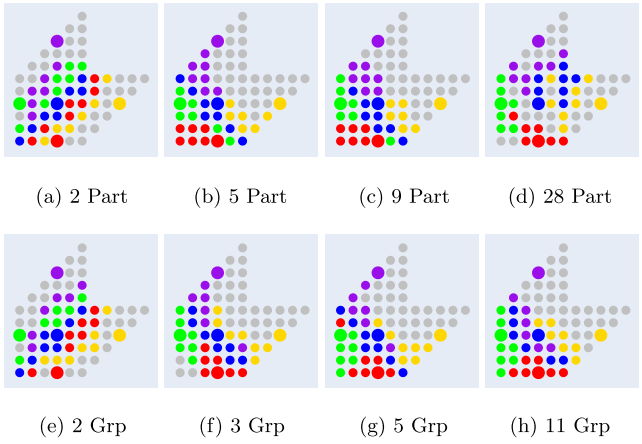


Fig. 13 Comparing relative labeling of part-based (a)(b)(c)(d) with component-based (e)(f)(g)(h) while varying the number of (Part) and (Grp) nodes respectively. x-axis, y-axis, and coloring are the same as in Fig. 12.

than a second, allows users to run the computation interactively with different filtering.

In particular cases, the scores spread is small between simulations, Fig. 15e and 15l. The narrow range can highlight limited exploration of the designs. Moreover, the singular high density of similarity prediction is related to the number of parts included in the similarity calculation, Fig. 15c and 15h.

Like in the illustrative example, a fully-connected bipartite graph has tighter prediction scores, and the effect of the weights is not as strong as the structure. For example, in a group of FE simulations, a fully bipartite graph means all 20 most energetic parts are the same for all the FE simulations; however, there is a difference between them due to the difference in energy distribution. One approach to extend the

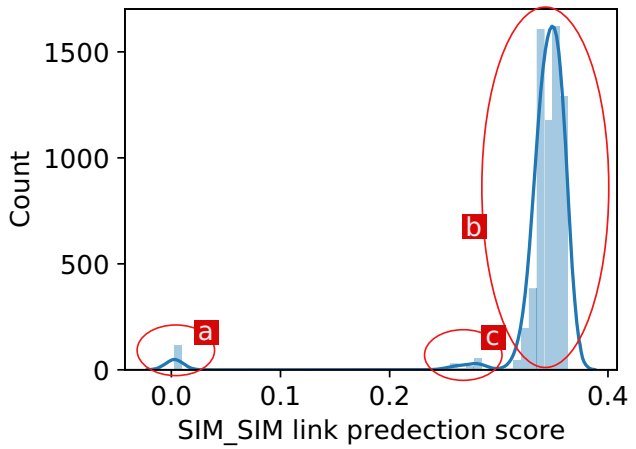


Fig. 14 s^{++}_{trgt} link prediction histogram and estimated density for the foI load-case in the primary stage, CEVT data. (a) ,(b), and (c) highlights the three cluster of similarity distributions.

deviation of the link prediction score is to include fewer parts to avoid having a fully bipartite graph.

7.5.2 H-LL simulations

We propose to compare the energy features [19] of several simulations to verify the similarity distances in detail. For that, we select what we call H-LL edges, where H is the edge with the highest similarity, that is the nodes H_1 and H_2, connected by edge H, are the most similar pair of simulations. The edge is according to the maximum predicted score of ○— SIM_SIM —○. In a second step, we select the corresponding least similar simulations, that is both H_1-L_1 and H_2-L_2, Fig. 16. In some cases, L_1 and L_2 may be the same. We call these H-LL simulations, and we further investigate the results with them to assess the reliability of the predicted similarity.

Table 6 shows a summary of link prediction for the foI load-case during the primary stage. First, we consider all the simulations, Table 6a. Each table row shows the predicted similarity of H-LL simulations while increasing the number of parts. For this data set, the graph is disconnected if we consider less than six most energetic parts. An additional observation is that the HL similarity drops after including at least 15 parts in the analysis. Accordingly, the 15 most energetic parts will have the highest similarity range, and afterward, the H-LL pairs are stable.

In a further step, we remove the outlier simulations from cluster (a) in Fig. 14. We identify the H-LL simulations while increasing the number of parts included in the bipartite graph, Table 6b. Increasing the number of parts keeps the similarities range shrinking constantly. Generally, the selection of parts for the similarity assessment is a hyperparameter for the user; however, the H-LL similarities support revealing the differences.

Additionally, we consider a plot using the energy features for a couple of H-LL simulations. With the so-called energy scatter plot [19], one compares the energy features of the most energetic parts of the simulations, which enables us to visually assess the similarity of the simulations' parts in energy absorption.

We start with the H-LL_1, (354)-(387)-(237), from Table 6a that s^{++}_{trgt} that predicted simulation (237) as least similar. Figure 17a visualizes the energy features of 20 parts for each of these three simulations. The plot indicates that simulation (237) has enormous energy in one part, and the simulation ended earlier, causing it to differ noticeably in comparison with the other two simulations. We expected the least similar simulation to be an outlier, cluster (a) in Fig. 14, and this comparison confirms it.

An initial remark from comparing Table 6a and b is that the minimum number of required parts in the dataset that avoids having a disconnected graph decreases from six to

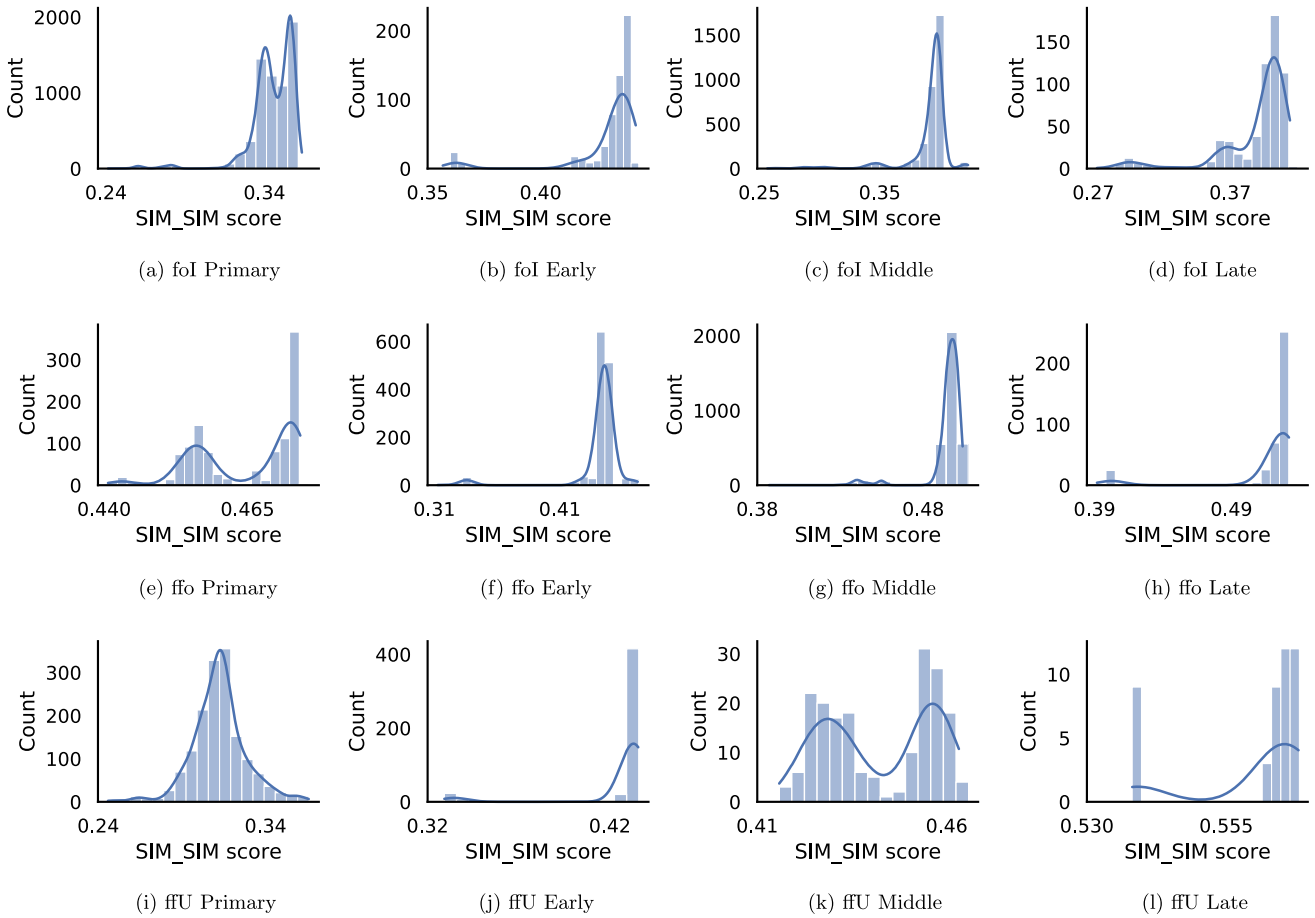


Fig. 15 KDE plot for s^{++}_{trgt} prediction with 20 most energetic parts in three different load-cases full front overload (ffo), front oblique overlap (foI), and front small overlap (ffU) in four development stages, primary, early, middle and late that reflects the sequence of the stage, for more information look in Section 3 of [19]. The simulations with a maximal similarity less than 0.2, and therefore outliers, are removed.

two, comparing Table 6a and b respectively. This drop is an additional cross-check for verifying the effect of filtering the outliers.

Next, we investigate H-LL simulations from Table 6a excluding the outlier simulations. Here, we expect to find the clusters (b) and (c) in the KDE plot, Fig. 14. Further comparison of Table 6a and b emphasizes that filtered simulations have the two most energetic parts in common, but there is a flip in its order that we can not decrease it to one part and still have a connected graph. Consequently, there is a bifurcation in the behavior of the simulations.

In Table 6b, increasing the number of parts does not settle in one H-LL simulations group, whereas in Table 6a occurs after 15 parts. However, in Table 6b, a trend is observed in H-LL simulations. These groups are marked in Table 6b as H-LL$_2$, H-LL$_3$, H-LL$_4$, they include simulations (354)-(387)-(17), (4)-(7)-(287), and (4)-(7)-(354), respectively. The first and second sets H-LL$_2$ and H-LL$_3$ do not overlap in simulations; however, H-LL$_4$ contains HH simulations of H-LL$_2$ and H-LL$_3$. This observation underlines that two behavior trends are dominant and have a detectable distance from each other.

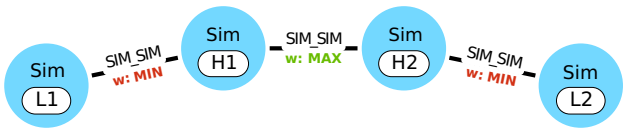


Fig. 16 H-LL simulation schema selected for each load-case in a development stage based on similarity prediction score.

We look into the energy features scatterplots of these simulations to verify our assumptions. Figure 17b, c, and d plot energy scatterplats of these simulations for H-LL$_2$, H-LL$_3$, and H-LL$_4$ respectively. In Figure 17b and c, we see that parts of HH simulations are nearby (blue and green markers), whereas L simulation points are further away (red marker). Furthermore, Fig. 17d proves the distance between HH of H-LL$_2$ and H-LL$_3$. Here we can see that the difference is not as noticeable as H-LL$_2$ and H-LL$_3$; parts with lower energy become dominant in similarity prediction if we increase the number of parts.

Table 6 The H-LL simulations id and their similarities, s^{++}_{trgt} $C = 0.95$, prediction score with varying parts included in the bipartite graph, including all the foI load-case simulations in the primary stage, CEVT data

No. Parts	H_1	H_2	L_1	L_2	H_1H_2	HL_1	HL_2
6	008	018	386	386	0.486	0.295	0.291
10	004	007	090	090	0.438	0.287	0.294
14	353	354	090	090	0.409	0.266	0.253
15	354	387	237	237	0.401	0.002	0.002
16	354	387	237	237	0.388	0.002	0.002
***20**	354	387	237	237	0.364	0.003	0.003

(a) All simulations, cluster (a), (b), and (c) in Figure 14

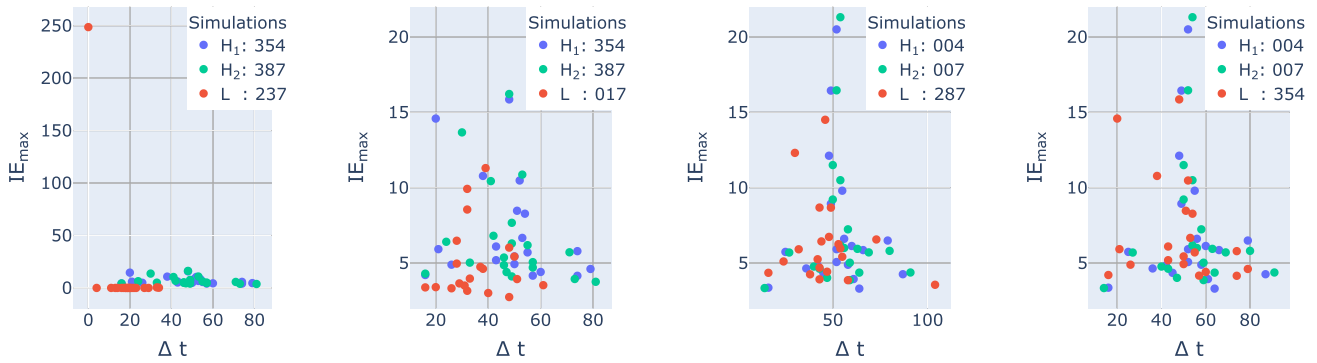
No. Parts	H_1	H_2	L_1	L_2	H_1H_2	HL_1	HL_2
2	195	197	354	354	0.477	0.113	0.112
4	354	387	197	197	0.455	0.108	0.123
5	135	197	021	021	0.485	0.236	0.237
6	135	190	287	287	0.492	0.304	0.307
8	354	385	028	028	0.468	0.322	0.346
10	354	357	028	017	0.453	0.340	0.358
12	004	007	287	287	0.442	0.369	0.370
***14**	004	007	354	354	0.427	0.375	0.376
16	354	387	017	021	0.405	0.344	0.350
**18	004	007	287	287	0.389	0.349	0.349
*20	354	387	017	017	0.377	0.321	0.331

* H-LL_1, Fig. 17a
* H-LL_2, Fig. 17b
** H-LL_3, Fig. 17d
*** H-LL_4, Fig. 17c

(b) Excluding outliers, cluster (b), and (c) in Figure 14

8 Application perspective

In this work, we introduced searchability to the CAE domain, improving the engineering workflow by enabling efficient discovery and comparison of simulation data. Figure 18 illustrates the workflow as a discovery platform for engineers who wish to use searchability in their projects. In that platform, engineers can select a specific part of a vehicle or analysis to obtain a ranked list of simulations based on their simi-



(a) H-LL_1 (b) H-LL_2 (c) H-LL_3 (d) H-LL_4

Fig. 17 Analysis of s^{++}_{trgt} prediction, energy features for 20 most energetic parts in H-LL simulations for primary development stage and FoI load-case. In all the plots blue-green markers refer to high similarity pairs of simulations and red is the low similarity to them. (Δt $[ms]$,IE_{max} $[kNmm]$).

larities. This capability provides several practical benefits to their analyses, including:

- **Find the same performance with a different design by investigating highly similar simulations.** Engineers can identify alternative designs that achieve similar performance metrics. This enables more innovative solutions and potentially more cost-effective alternatives. For example, if an engineer is optimizing a car door design, they can find alternative designs that meet the same safety standards but may offer other benefits such as reduced weight or cost.
- **Evaluate the robustness of a design by considering the spread of simulation similarity scores.** By analyzing the variation in similarity scores, engineers can assess how consistent the performance of a design is under different conditions or assumptions, thereby evaluating its robustness. For example, when analyzing the crashworthiness of a vehicle, engineers can assess how minor design changes affect overall performance. This ensures that the design remains robust under different conditions.
- **Identify unreliable simulation analysis due to a low simulation similarity score.** Simulations that deviate significantly from others can be flagged for further investigation, helping to identify potential errors or unreliable data.
- **Cluster simulations based on the distribution of their similarity scores.** Clustering similar simulations can reveal patterns and relationships within the data, helping to understand how different factors influence performance. This clustering can help engineers understand the relationships between different designs and identify trends or common characteristics among high performing designs.

In summary, controlling the variation of structural properties is a fundamental task in the design of lightweight structures. This is especially critical when applying deterministic structural optimization methods, such as topology optimization, to individual components or entire structures. The presented searchability method not only supports the discovery of similar simulations, but also helps to manage these variations. Our approach ultimately contributes to more effective and efficient structural design and optimization processes.

In addition, we have introduced the use of KDE and the H-LL to visualize and summarize the differences across a large number of simulations. These visualizations play a critical role in the practical CAE process by providing an intuitive and interactive discovery platform. Engineers can actively select input simulations from the KDE distribution and validate these clusters against the H-LL simulations. This interactive approach allows for more detailed and focused analysis.

For a practical CAE process, the discovery platform allows engineers to adjust hyper-parameters such as the number of parts included or which specific parts are included in the similarity evaluations. This flexibility allows users to customize the focus of the comparison to suit their specific interests and needs. In addition, the platform supports the integration of

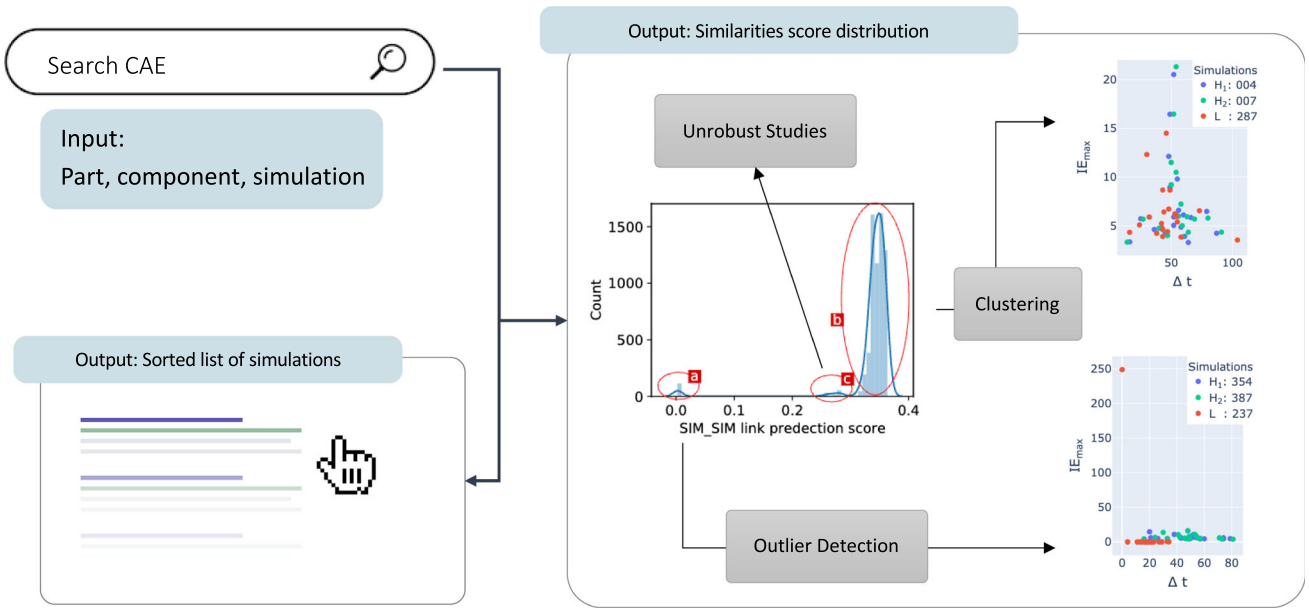


Fig. 18 User workflow for simulations searchability. Engineers can select specific vehicle parts or analyses to obtain ranked lists of simulations based on similarity scores, aiding in design comparison, robustness evaluation, detection of unreliable simulations, and trend identification, crucial for optimizing structures.

human feedback, which is an essential requirement for the improvement of data labelling for further enhancement of method development.

Given the novelty of predicting simulation similarity and the lack of labeled data, these types of visualizations are essential for evaluating the similarity results. They facilitate the integration of the proposed method into the CAE workflow, enabling the collection of valuable feedback from engineers. The integration of these methods into CAE processes is a critical requirement for future improvements using graph analysis. This holistic approach ensures that the methodology evolves with practical insights and real-world applications, driving continuous improvement in the field of CAE.

9 Conclusion and Outlook

Today, the searchability of the web is an obvious benefit for everyone. However, enhanced searchability still needs to be realized in the CAE domain, where its advantages need to be demonstrated to the engineers. For example, it enables multidisciplinary collaboration by easing the finding of data within the team and across the company, which increases efficient problem-solving. Moreover, it allows different interconnecting solutions for the same problem and highlights unexplored solutions.

In the context of crash simulation searchability, we focused on predicting the similarity of simulations and a corresponding ranking. Regardless, enhanced searchability will also bring benefit to other CAE domains and other semantics of a domain, e.g., design features, cause and effect analyses, modeling techniques, discipline requirements, and project decisions.

In this work, the case study is in front crash simulations. Other load-cases and CAE domains are beyond the scope of this work. For other crash load-cases, the same semantics are recommended; however, introducing new illustrative data for assessment of the method is required. Additionally, other domains require introducing the semantics that characterize the analysis's mechanical properties, e.g., high strain elements for fatigue analysis and eigenmodes structural frequencies for noise-vibration-harshness analysis.

Our graph modeling results in a heterogeneous graph and we need to consider unsupervised learning. To be able to use SimRank-style methods, the only method we found suitable in our scenario, we extracted a bipartite-weighted graph. Additionally, we introduced an alternative weight normalization for SimRank++. We could show that this works better for our addressed application of predicting the similarity of crash simulations and the corresponding ranking of simulations, shown on a constructed illustrative example with relative labels.

Furthermore, we compared similarity predictions using part-based and component-based approaches. Here we introduced an automatic approach to group the parts and combine the features. While the overall outcome for part-based and component-based similarity is close, the component-based similarity provides a more stable prediction, whereas the part-based similarity is a more sensitive technique. Consequently, we recommend component-based similarity in partial comparison of simulations and part-based for complete model comparison.

On industrial data, we could show that our methods scale up to real data scenarios. Overall, we could verify the simulation similarity prediction, while the data representations showed promise for outlier detection and clustering of simulations based on similarity score distribution.

In order to improve the graph-based search capabilities within the CAE domain, future work should focus on the extension of the graph model. This can be achieved by the incorporation of quantified physical parameters, which are relevant for each specific CAE domain, as new weighted features of the edges. This is similar, for example, to the incorporation of energy features in crash simulation.

Furthermore, one can consider quantifying the input designs and including more outputs of the simulations. Design features support further link prediction tasks such as the similarity of the FE-models' inputs or cause-effect relations that interconnect the input designs and output features. However, including more simulation outcomes will also require a feature embedding to combine the features or aim for a SimRank formulation with multi-edge weights.

Additionally, extending applications of graph analytics methods for domain-specific demands will assist cross-domain solutions. Including the multidisciplinary requirements as the search object will support ranking the cross-discipline solutions. Another step to enhance searchability is to improve the data labeling to leverage result assessment. Enhanced graph models with larger numbers of simulations stored would also allow applying graph neural networks in this domain.

One approach for improving data labeling is the integration of the current method in companies as a dynamic report platform to collect feedback from engineers, e.g., CAEWebVis[12] introduced in [19]. Only with feedback on the predicted similarities from engineers can one envision a transferring of the unlabeled CAE data to labeled data. Such labeling will open up new possibilities to empower further ML solutions, e.g., graph neural networks. To allow further investigation of graph analytics for the CAE process, we release our illustra-

[12] Accessible at CAEWebVis.scai.fraunhofer.de/.

tive example, the databases and the source code with a user tutorial[13].

The s^{++}_{trgt} method presented has significant potential, which would benefit from a benchmark study to evaluate its performance in different applications. Unfortunately, our current work has focused specifically on crash simulation similarity, so such an evaluation is beyond the scope of this study. Nevertheless, we recommend applying the s^{++}_{trgt} method to any bipartite weighted graph where the relationships between nodes represent a ○— PART_OF —○ concept. In these scenarios, the weight distribution between entities in different groups is more relevant for similarity judgments than the distributions within groups. The reason for recommending the use of s^{++}_{trgt} is that its semantic comparison in normalization preserves minor changes in each semantic during similarity evaluation. Simultaneously, s^{++}_{trgt}'s evidence factor accounts for each semantic's importance in the overall similarity calculation. To our knowledge, the theoretical investigation focussed so far on SimRank, in particular in view of approximation for large scale application. The presented matrix formulation could allow further investigation on SimRank++, also in view of the random walk interpretation of SimRank [2, 12, 25].

This method could be particularly useful for other automotive analyses beyond safety, such as noise, vibration and harshness (NVH), durability and computational fluid dynamics (CFD). In NVH studies, for example, the method can be used to compare the vibrational behavior of vehicle components across different design iterations, enabling engineers to identify and minimize sources of unwanted noise and vibration early in the design process. Similarly, in durability analysis, the method can be used to compare the long-term performance and reliability of different design versions, helping to develop more robust and resilient automotive parts. In CFD, it can help compare the aerodynamic and thermal performance of different design configurations, optimizing for better fuel efficiency and thermal management.

In addition, we suggest exploring the use of s^{++}_{trgt} in other domains in view of its applicability. For example, it could be applied to document comparison, such as comparing sections of different document versions (e.g. (section) — PART_OF — (document)), which facilitates tracking changes and improvements in technical documentation. In software engineering, it can be used to compare code modules within packages (e.g. (module) — PART_OF — (package)), aiding version control and ensuring consistency and compatibility between software updates. These use cases demonstrate how the method can be generalized as a powerful tool for version control, effectively conveying the same semantics across different versions - be they sections in documents, modules in software packages, or parts in vehicles. This cross-disciplinary application highlights the potential of the method for process streamlining and accuracy improvement in a wide range of fields.

[13] github.com/Fraunhofer-SCAI/GAE-vehicle-safety.

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Competing interests The authors have no competing interests, as defined by Springer, or other interests that might be perceived to influence the results and discussion reported in this paper. This work was partly funded by the German Federal Ministry for Education and Research (BMBF) within the project ViPrIA, FKZ 01IS19014C.

References

1. Ackermann S, Gaul L, Hanss M, et al (2008) Principal Component Analysis for Detection of Globally Important Input Parameters in Nonlinear Finite Element Analysis. In: Weimar optimization and stochastic days 5.0
2. Antonellis I, Garcia-Molina H, Chang CC (2007) Simrank++: Query rewriting through link analysis of the click graph. In: Proceedings of the 17th international conference on World Wide Web, pp 1177–1178
3. Berkhin P (2005) A survey on PageRank computing. Int Math 2(1):73–120
4. Center for Collision Safety and Analysis (2015) Toyota Yaris finite element model validation detail mesh. https://www.ccsa.gmu.edu/models/2009-toyota-yaris/
5. Du X, Zhu F (2018) A new data-driven design methodology for mechanical systems with high dimensional design variables. Adv Eng Softw 117:18–28. https://doi.org/10.1016/j.advengsoft.2017.12.006
6. Du X, Zhu F (2019) A novel principal components analysis (PCA) method for energy absorbing structural design enhanced by data mining. Adv Eng Softw 127:17–2. https://doi.org/10.1016/j.advengsoft.2018.10.005
7. Du Bois P, Chou CC, Fileta BB, et al (2004) Vehicle crashworthiness and occupant protection. Am Iron Stell Inst pp 27–280, 304–330
8. Fan H, Zhang F, Wei Y et al (2022) Heterogeneous hypergraph variational autoencoder for link prediction. IEEE Trans Pattern Anal Mach Intell 44(8):4125–413. https://doi.org/10.1109/TPAMI.2021.3059313

9. Garcke J, Pathare M, Prabakaran N (2017) ModelCompare. In: Scientific computing and algorithms in industrial simulations. Springer, pp 199–205
10. Garcke J, Iza-Teran R, Pathare M, et al (2022) Identifying similarities and exceptions in deformations and mesh functions. In: SIMVEC 2022, pp 277–288. https://doi.org/10.51202/9783181024072-277
11. Iza-Teran R, Garcke J (2019) A geometrical method for low-dimensional representations of simulations. SIAM-ASA J Uncertain Quantif 7(2):472–49. https://doi.org/10.1137/17M1154205
12. Jeh G, Widom J (2002) SimRank: a measure of structural-context similarity. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp 538–543
13. Kim YD, Choi S (2009) Weighted nonnegative matrix factorization. In: 2009 IEEE international conference on acoustics, speech and signal processing, pp 1541–154. https://doi.org/10.1109/ICASSP.2009.4959890
14. Kracker D, Dhanasekaran RK, Schumacher A, et al (2022) Method for automated detection of outliers in crash simulations. Int J Crashworthines 1–12
15. Lee JA, Verleysen M (2007) Nonlinear Dimensionality Reduction. Springer
16. Liu TY (2011) Learning to Rank for Information Retrieval. Springer
17. Mei L, Thole CA (2008) Data analysis for parallel car-crash simulation results and model optimization. Simul Model Pract Theory 16(3):329–337. https://doi.org/10.1016/j.simpat.2007.11.018
18. Pakiman A, Garcke J (2022) Graph modeling in computer assisted automotive development. In: 2022 IEEE international conference on knowledge graph (ICKG), pp 203–21. https://doi.org/10.1109/ICKG55886.2022.00033
19. Pakiman A, Garcke J, Schumacher A (2023) Knowledge discovery assistants for crash simulations with graph algorithms and energy absorption features. Appl Intell 53:19,217-19,23. https://doi.org/10.1007/s10489-022-04371-w
20. Pan S, Hu R, Long G, et al (2018) Adversarially regularized graph autoencoder for graph embedding. In: Proceedings of the 27th international joint conference on artificial intelligence. AAAI Press, IJCAI'18, pp 2609–2615
21. Patel A, Jain S (2021) Present and future of semantic web technologies: a research statement. Int J Comput Appl 43(5):413–422
22. Schembera B, Durán JM (2020) Dark data as the new challenge for big data science and the introduction of the scientific data officer. Philos Technol 33(1):93–115
23. Schwanitz P (2022) Towards AI based recommendations for design improvement (AI-B-REDI), presentation at SIMVEC, Baden-Baden November 2022
24. Schwanitz P, Greve L, Moldering F (2022) Towards AI based recommendations for design improvement (AI-B-REDI). In: SIMVEC 2022, pp 299–314. https://doi.org/10.51202/9783181024072-299
25. Wang H, Wei Z, Liu Y et al (2021) ExactSim: benchmarking single-source SimRank algorithms with high-precision ground truths. VLDB J 30(6):989–1015
26. Wang Y, Wang Z, Zhao Z et al (2020) Effective similarity search on heterogeneous networks: A meta-path free approach. IEEE Trans Knowl Data Eng 34:3225–3240