Applied Intelligence (2023) 53:19217–19236
https://doi.org/10.1007/s10489-022-04371-w

Knowledge discovery assistants for crash simulations with graph algorithms and energy absorption features

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

Accepted: 28 November 2022 / Published online: 27 February 2023

Abstract

We propose the representation of data from finite element car crash simulations in a graph database to empower analysis approaches. The industrial perspective of this work is to narrow the gap between the uptake of modern machine learning methods and the current computer-aided engineering-based vehicle development workflow. The main goals for the graph representation are to achieve searchability and to enable pattern and trend investigations in the product development history. In this context, we introduce features for car crash simulations to enrich the graph and to provide a summary overview of the development stages. These features are based on the energy output of the finite element solver and, for example, enable filtering of the input data by identifying essential components of the vehicle. Additionally, based on these features, we propose fingerprints for simulation studies that assist in summarizing the exploration of the design space and facilitate cross-platform as well as load-case comparisons. Furthermore, we combine the graph representation with energy features and use a weighted heterogeneous graph visualization to identify outliers and cluster simulations according to their similarities. We present results on data from the real-life development stages of an automotive company.

Keywords FE analysis · Automotive · Semantic data · Outlier detection · CAE knowledge · Knowledge graphs · Graph database · Heterogeneous graph

1 Introduction

In the past 30 years, the reliability of the finite element (FE) method for predicting the crash behavior of vehicles has steadily improved. FE modeling improvements resulted in more and more detailed simulations with continuously intensifying complexity of the data. Moreover, the growth of computing power has increased the number of simulations. Due to this data and complexity growth on the one hand and limited availability of engineering time on the other hand, simulation result data is often unexplored.

Furthermore, the complexity and size of the simulation data also prevent the direct application of most machine learning (ML) methods on the simulation data, e.g., to capture and detect patterns and trends. We also observe that determining a simulation data representation based on engineering principles, which in particular helps to quantify crash behavior, is a largely unexplored research area.

Therefore, we propose a data representation approach called vehicle knowledge graph (KG), car-graph in short, which allows the discovery of patterns and trends in simulation results. The visionary goal of the car-graph ansatz is to extract a crash identity of the vehicle. A short-term outcome is an assistance tool for engineers to efficiently explore and evaluate data, e.g., from previous simulation studies in different projects. The assistance functionality shall encompass searchability and the capturing of patterns and trends, while the aim is to support the prediction of outcomes or the recommendation of solutions. To achieve this, we represent crash simulation data in a graph database, including suitable physical properties, to generate a KG and to empower graph-based ML algorithms.

✉ Anahita Pakiman
anahita.pakiman@scai.fraunhofer.de

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

There are various options available for the selection of physical properties from the different computer-aided engineering (CAE) outputs. Most of these depend on the simulation setups and are too costly for comparing many simulations. Hence, we propose the usage of a few scalar features stemming from the internal energy (IE) of components, both in our graph representation of simulation data and for the analysis of simulation results. The fundamental physics of a crash accident is to absorb the energy of the impact through structural deformations, causing an increase in the IE. The close physical connection of the IE properties to the crash problem and their ease of use, with no need for further simulation pre-processing, makes them a compelling candidate for further investigations. First, we demonstrate the capability of the IE features to capture the differences between simulations with minor changes. Later we show that the chosen IE features are characteristic enough for any given simulation (with 10-12 million elements) to enable graph algorithms to perform well with the comparatively small number of simulation data (200-300).

Combined, we build the car-graph, focusing on the internal energy in the CAE outputs. We introduce features based on the IE, which allow visualizations that enable the engineer to extract additional knowledge from and gain insight into simulations. For example, one can study part similarity, summarize development stages, or analyze crash behavior. Finally, we integrate the proposed energy features into a heterogeneous weighted graph, which allows the identification of outliers and absorption trends, as well as a visual clustering with force-directed graph algorithms [19].

Figure 1 summarizes our approach, where we introduce new visualizations for knowledge discovery with the support of graph databases. Next, we recapture related work in Section 2, followed by a description of the investigated industrial data in Section 3. Then, we introduce features for energy curves in Section 4. Afterwards, we present a graph database structure and investigate the ranking of the design components during development stages in Section 5. Later, we explore energy features for identifying similarities and introduce design of experience (DOE) fingerprints in Section 6. Further, we use graph visualizations in Section 7. A conclusion and outlook are presented in Section 8. Figure 1 summarizes the approach in this study in introducing new visualization for knowledge discovery with the support of graph databases.

2 Related work

The proposed car-graph is inspired by information retrieval and mining trends that have transferred from document-centric to entity-centric. Since IBM Watson won Jeopardy in 2011, knowledge graphs have gained increasing research interest due to their capability of storing knowledge, structured or unstructured, elicited from heterogeneous domains, and their further querying to realize question answering [26]. A survey in domain-specific knowledge graphs [1] summarizes available knowledge graphs in engineering.

The most relevant engineering domain for our research is manufacturing. However, ongoing research focuses more on production and manufacturing than product development. Moreover, there are mostly text-based knowledge graphs, and only limited work exists on 3D shapes as the center of the product description [3]. Example applications of knowledge graphs include digital twin models for industrial production, industry 4.0, and computer-aided manufacturing (CAM). In [5], an overview of available

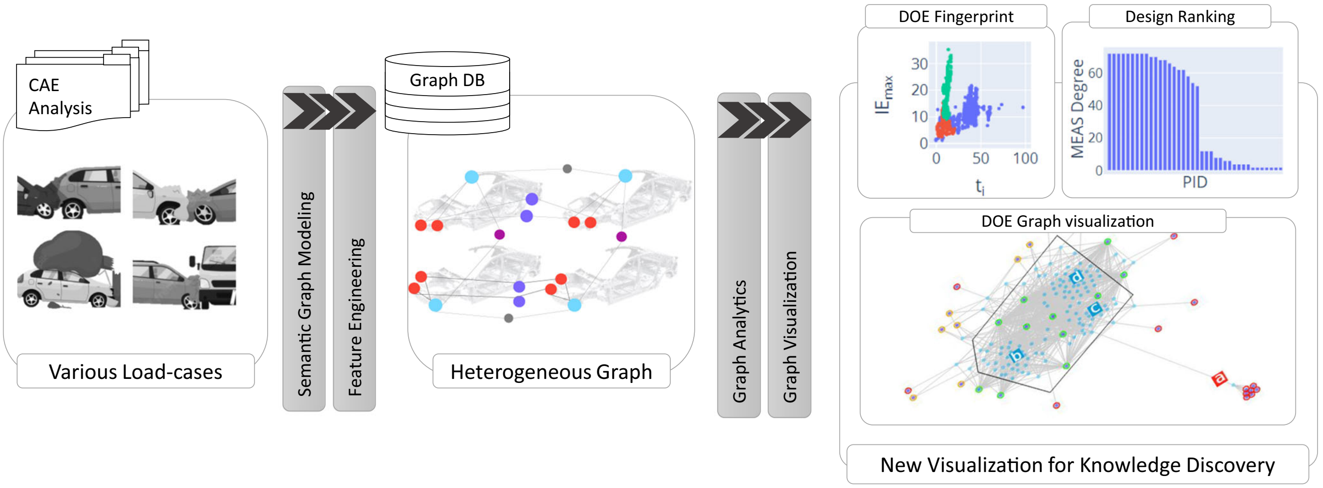


Fig. 1 The graph-based analysis and visualization workflow involves loading crash simulations to a graph database, including the computation of engineering features and a data visualization for knowledge discovery. The node coloring in the heterogeneous graph reflects different node types and follows the schema in Fig. 6, fingerprints, degree distributions, or force-directed graph layouts are examples of visualization approaches

research in manufacturing is provided. Currently, there is no knowledge graph available for the CAE/FE domain nor specifically crash, which is the focus of this paper.

However, there are investigations on ontologies for CAD and CAE integration [4], FE simulation [23], and crash analysis [12, 13]. In the context of computing, an ontology is a concrete, formal representation of what terms mean within the scope in which they are used (e.g., a given domain). Similar to other conventions, the usefulness of an ontology depends on the level of agreement on what that ontology defines, how detailed it is, and how broadly and consistently it is adopted. Adopting an ontology by the parties involved in one knowledge graph may lead to consistent use of terms and modeling in that knowledge graph [16].

According to [23], several studies have already applied a knowledge-based ontology system to provide simulation knowledge to FE users. These studies disregard extracting new relationships among the data or answering analytical questions of an engineer. In some, the focus was on automating the generation of the FE simulation [23] or retrieving simulation solutions from existing simulation [24]. However, the case studies are simpler [23, 24] than a full crash simulation. [31] characterized the CAE domain and identified unsolved challenges for tailored data and metadata management as a graph. [12, 13] looked explicitly at a crash simulation ontology and investigated the reasoning structure of engineers, particularly regarding report generation. Overall, [12, 13, 31] have a knowledge management system orientation to understand data structure and procedures in the company, while the simulation data itself has not been studied.

To summarize, previous articles have a product management perspective representing the crash development process structure. CAM, computer-aided design (CAD), and CAE are different product data in the development phases of vehicles. There are more research studies on knowledge graphs in CAD and CAM compared to CAE. Additionally, automated CAD-CAE model integration generated ontology models and geometrical feature extractions. These studies are of interest to connect CAD knowledge to CAE. Kirkwood et al. [25] uses design change vectors to enable sustained integration of FE mesh and CAD models. Feng et al. [14] represents an automatic approach to generate simplified and idealized geometry models for CAE simulation.

Parametrized CAD models, knowledge management, and knowledge-based engineering (KBE) systems have for decades strived to capture, digitize, and automate the application of this kind of knowledge within product and production development [21]. Using the product design knowledge graph, [3] demonstrated the effectiveness of 3D shape retrieval using an approximate nearest neighbor search. They illustrate using the KG for design reuse of co-occurring components, rule-based inference for assembly similarity, and collaborative filtering for a multi-modal search of manufacturing process conditions. However, KG should still expand to include downstream data within product manufacturing and towards improved reasoning and methods to provide actionable suggestions for design bot assistants and manufacturing automation. Additionally, [17] showed that in a design context, a cognitive assistant enables the participants to select applicable design rules more precisely, allowing them to spend more time on the CAD modeling activity. However, there is still a need for a test protocol to confirm the preliminary results presented in this paper.

In view of the general application of ML for crash data and knowledge graphs, it is not broadly used in current CAE workflows compared to typical machine learning domains. There are two main applications of ML in crash analysis. First, it predicts the crash behavior to replace/support the FE simulation; see [2]. Second, using dimensionality reduction on the vehicle components' data during crash deformation to explore FE simulations, e.g., by identifying clusters [18]. Here, an engineer must usually specify the critical components in advance. While considering all parts together is time-consuming and inefficient, it may also fail to highlight the bifurcation behavior. This limitation emphasizes the importance of auto-detecting and filtering the essential components.

Regarding energy absorption characteristics for crash simulation, studies show that energy absorption characteristics enable quantifying component performance for the design of experiments (DOE) feedback in optimization studies [9, 10]. However, to our knowledge, there is no research on using energy curve features to calculate the similarity of simulations or summarize development stage results.

3 OEM data from CAE development stages

We evaluate the proposed data representation and the resulting data exploration approaches on industrial data stemming from a vehicle development project undertaken at CEVT. In particular, we consider four development stages and three load-cases for front impact analysis. The development stages are so-called primary, early, middle, and late development stages, where the names reflect the sequence of the stages. The considered development window covers roughly one-third to two-thirds of the complete R&D development phase (before the first real crash test). Table 1 summarizes the three load-cases and the number of included simulations.

In particular, we aim to assess the scalability and feasibility of the introduced energy features and graph algorithms. The focus is on data visualization to summarize the behavior and trends. Note that data confidentiality hinders illustrating the developed vehicle platform or giving

Table 1 Properties of the investigated CEVT data. The total number of all simulations from four development stages in each load-case. The deviation of the KE is due to the changes in vehicle mass during development

Load-case	Name	No. Sim	KE_i* $[kNmm]$ range	Velocity $[km/h]$
ffo**	full front overload	215	328.6 − 389.4	64.0
foU	front oblique overlap, new US-NCAP	121	346.4 − 436.2	90.0
foI	front small overlap, IIHS	275	778.0 − 890.4	66.9 †

*Initial kinetic energy

**Internal load-case

†Over loaded speed, requirement is 40 $[mph]$

details about the FE model. However, we can discuss the crash behavior using the component name. So the general knowledge of crashworthiness helps to interpret and evaluate the results. Tables 3 and 4 summarize the components referred to in this paper.

Generally, the positions relative to the direction of the barrier are essential in the analysis of crash behavior with regard to geometry. Therefore, one can divide the components into the early, middle, and late energy absorbent components, i.e., bumper beam, crash-box, and side-member, respectively. Additionally, the vehicle's vertical axis positioning includes middle and lower load paths in the absorption, i.e., crash-box and lower load path component, respectively.

The crash-box and the side-member are thin-walled structures with well-designed cross-section shapes and crumple points, e.g., ditches and crash beads. They may collapse in a particular pattern to absorb energy efficiently. A side-member is longer and stiffer in comparison to crash boxes. Further, the deformation modes of longitudinal beams include folding, tearing, and bending. Here, reinforcing components strengthen the beams and optimize the absorption of energy. However, the lower load path component is a thin-walled structure positioned vertically lower than the crash-box. It distributes the load in the subframe. Finally, the subframe is a structural component with a discrete structure that supports the axle, suspension, and powertrain. This component has minor absorption crashworthiness design aspects among the mentioned components due to the required durable performance.

So far, we have introduced the components of the ffo load-case that are studied in-depth in Section 6.3. Additional essential components for the foU and foI load-cases are the A-pillar, cowl, front fascia, wheel arch, and wheel rim. A-pillar is the most forward vertical support of the vehicle (among A, B, C, and D pillars). The cowl separates the front compartment from the passenger cabin between two A-pillars. The rest of the components are none structural. For further background on crashworthiness, see e.g. [11].

4 Energy features

The deformation structures of the vehicle shall absorb the kinetic energy so that the occupants and pedestrians have the least possible injuries. In the case of shell structures, this is done by means of suitable deformation patterns. The main underlying physics of the crash analysis problem is the energy dissipation performance. CAE engineers often determine the crash behavior only by analyzing the intrusion, the acceleration, the force, the deformation, and the failure. These parameters are assessed and correlated with reality visually and quantitatively. They do not take into account the energy dissipation performance. However, the energy is a solver output available for all parts in a simulation, whereas section forces require specific FE model preparation.

The FE solver outputs energy per part over time, a so-called energy curve. Parts in the CAE model preferably refer to each vehicle component. Despite this, CAE modeling techniques require arranging vehicle components into several properties, for example, due to changes in the thickness and material. Consequently, CAE models have many parts (1500-4000). The number of parts confronts CAE engineers with a practical assessment of energy curves. Therefore, it limits the use of energy curves in the workflow to, e.g., stability investigations (checks the simulation's total energy) and outlier entity identifications (e.g., parts with negative energy).

We claim that energy curves hold information to characterize the simulation crash behavior. Data analysis on energy curves will simplify data processing to represent the crash behavior based on a few features. Figure 2a shows the energy curve for the most energetic part of a complete vehicle simulation in a front overload load-case,

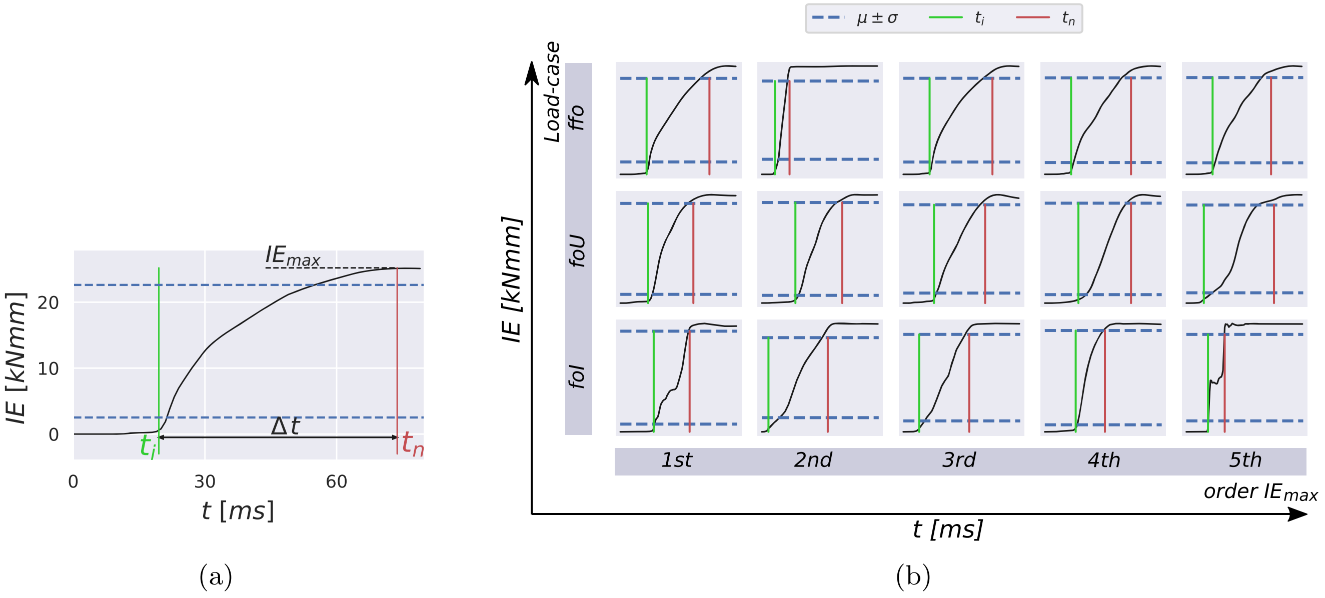


Fig. 2 (a) The internal energy output of the solver over time for a single property with the three features t_i, Δt, t_n that characterize the energyabsorption. The dashed blue lines that intersect with the energy curve visualize the standard deviation approach. (b) Diversity of internal energy output over time for the five most energetic parts, Table 3, in three load-cases, Table 1, with calculated initial and end of the absorption time and standard deviations. x and y axes are normalized

ffo, with a total initial kinetic energy of 453 [$kNmm$] (initial velocity of 64 [km/h]). The shape of internal energy over time is approximately a sigmoid curve, except for parts with negative energy due to numerical error. From the crash analysis perspective, measures with the potential to analyze the crash behavior from this curve are initial absorption time, absorption end-time/period, and absorbed energy, Table 2. These features indicate three abstract characteristics of the energy curve. We define absorption time with Δt and t_n as relative (to initial time) and absolute measuring, respectively. For now, we keep both features and study the functionality of each in different applications.

Figure 2b shows representative examples of internal energy curves and the features extraction over several simulations and parts. These curves belong to three simulations from three different front impact load-cases (Table 1) that we selected randomly. For each simulation, the five most energetic parts are plotted (part names in Table 3 and part definition in Section 3). In these examples, the shape of the curve during the absorption time (Δt) is nonlinear for some parts (first and fifth part in foI load-case). This nonlinearity indicates the probable necessity of additional features or more complex methods for characterizing the absorption interval.

Table 2 Introduced scalar features representing an energy curve

Energy Features		
t_i	initial absorption time	The initial time that the energy absorption starts.
$IE_{\max}$	absorbed energy	The max. internal energy absorbed by the part.
t_n	absorption time	The time in which $IE_{\max}$ is reached.
Δt		$t_n - t_i$

We define $IE_{\max}$ as the max of the internal energy curve, and in the following, we describe the time extraction features t_i, t_n, and Δt. Here, the preciseness of the timings depends on the solver's time interval output. We investigate three approaches to estimate the features based on the IE behavior: thresholding, derivative change, and standard deviation spread. With thresholding, one considers the time when the IE crosses a pre-defined threshold value. The derivative-based method calculates the internal energy derivative ($\dot{IE}$) and determines a significant change. However, using the spread $\mu \pm \sigma$, we consider upper and lower thresholds depending on the mean μ and standard deviation σ of the IE for the part over time.

Figure 2 shows the methods for t_i and t_n versus the time standard deviation. To summarize our observations, the derivative method is more suitable for t_i due to its sensitivity in capturing trigger time. However, thresholding performs more desirable for t_n since IE growth is saturating at the

Table 3 Part name for Fig. 2b, load-case information Table 1

load-case	1st	2nd	3rd	4th	5th
ffo	side-member RHS-I	crash-box LHS-V	side-member LHS-I	subframe LHS-U	subframe RHS-U
foU	side-member RHS-I	crash-box LHS-V	side-member LHS-I	subframe LHS-U	subframe RHS-U
foI	A-pillar LHS-I-L	front fascia	wheel Arch -F	cowl -L	wheel rim LHS-F

RHS: right-hand side, LHS: left-hand side

-U: upper, -V: vertical, -L: lower, -I: inner, -F: front

end. Furthermore, the standard deviation approach fails for the parts with a long absorption time or negative IE in the initialization. In the following sections, we will compare thresholding and derivative-based methods for t_i and t_n in more detail. For this, we use visual engineering judgment. We consider both methods on random samples from three complete vehicle front load-cases in four development stages, Table 1. For each simulation, we consider the 20 most energetic parts.

As mentioned, the features are not continuous values. Their resolution depends on the solver settings for the timestep output, which can vary from 1 $[ms]$ to 0.001 $[ms]$. Consequently, the features binning is according to the timestep definition. We consider a further and detailed investigation into binning and resolution as out of the scope of our initial study into energy features. Further, we focus on these three features, although considering other features during the absorption may contain more component characteristics during a crash simulation. However, extracting more features is out of the scope of this work, where our focus is to investigate the potential of features from energy curves, but not to find the best approach to achieve this. The described features were selected because of their simplicity to use and interpret.

4.1 Initial time

The initial time t_i for each part reflects when a part begins to absorb the impact energy. In crash simulations, there is a gap between the start time and the time when a sp ecific structural part gets affected by the crash, which makes finding the exact time imprecise. Here, the extracted t_i from the thresholding and derivative-based methods are close for most of the studied parts. Further, we investigated parts with significant differences in the two calculated t_i for visual comparison. Figure 3 presents an example of such a part with significant differences, together with a part where both approaches give similar t_i. From visual engineering judgment, part two starts to absorb energy earlier than part one, whereas the thresholding method extracts the same time ($t_i \approx 65$) for both parts. However, the derivative-based method computes for part two an earlier time ($t_i \approx 40$) than the thresholding method, which is preferred from an engineering perspective. Consequently, we perform further investigations with the derivative-based method.

The derivative-based method requires curve filtering due to the non-smoothness of the curve. We investigate filtering methods from SciPy.signal. From lfilt[1], filtfilt[2] and sosfilt[3] we select the FIR filter (lfilt, sample number n=75, b=1/n a=1), which smoothens the curve without any time shift, Fig. 4a. With this filter and a min-max normalization of the IE derivative, t_i is extracted as the time when the derivative is above 0.005. Both methods' lower limits are selected based on visual engineering judgment and visual tuning of the explored data. Figure 4b shows the result for a selected part.

4.2 Absorption time

We define the absorption time interval Δt as the time interval from when the part's internal energy increases until it stabilizes to its maximum. The start time is t_i from the last subsection, and the time at the end of absorption is t_n. To extract t_n, we compare the outcome of thresholding of $IE_{\max}$ and consider the derivative. For thresholding, we introduce a factor y for $IE_{\max}$ to exclude the gradual energy increase at the end of the simulation:

$$t_n = \max_t \{t \mid IE(t) \leq y \times IE_{\max}\}, \tag{1}$$

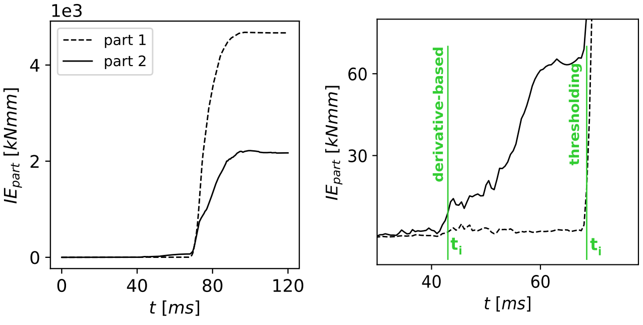
$$\Delta t = t_n - t_i. \tag{2}$$

The factor y is set to 0.95 based on a visual judgment from an engineering perspective on randomly selected simulations. For the method using derivatives, we consider the second derivative of the IE equal to zero as the time for t_n. This calculation requires filtering, where we evaluate the same filters as for t_i. However, here a time shift of the filtering is inevitable; see Fig. 4b. Consequently, the second derivative does not allow the reasonable extraction of the absorption time. Therefore, the maximum percentage time provides the best result for t_n.

[1] Filter data along one-dimension with an IIR or FIR filter

[2] A digital filter forward and backward to a signal.

[3] Filter data along one dimension using cascaded second-order sections



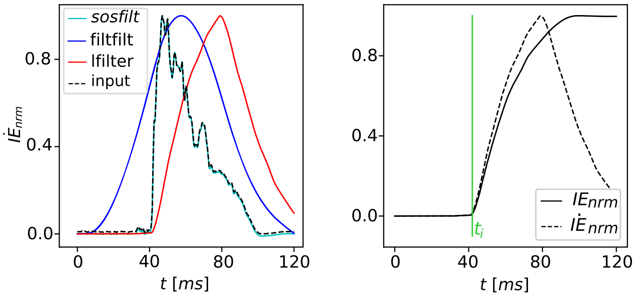
(a) (b)

Fig. 3 (a) IE curves for two different parts. (b) zoomed view of (a), the derivative-based method for part two (left marker) gives an earlier, and preferred, t_i than thresholding (right marker)

4.3 Discussion

Lastly, the standard deviation of IE is calculated for the parts, Fig. 4b. The crossing of IE with $\mu \pm \sigma$ refers to t_i and t_n. However, there is a deviation in the result for different parts compared to the other method. Two extreme examples, not shown in the figure, show undesired results are curves with long absorption time and the components with spring-back FE modeling, i.e., negative IE in the initialization. In both situations, the standard deviation is relatively tiny and causes a higher value for t_i and a smaller value for t_n, respectively.

Note that parts with common t_i and t_n in one simulation are simultaneously involved during the crash. Identifying such simultaneous parts can be used to identify parts for grouping as one absorption block. But, such a grouping is out of the scope of this paper and part of future work. Furthermore, we can filter out the parts that behave similarly from the energy absorption perspective by considering parts that share all three features for several simulations.



(a) (b)

Fig. 4 (a) Several filters are applied to the derivative of the IE over time, where filtfilt[2] and sosfilt[3] miss the ramp-up time. (b) Normalized IE and normalized $\dot{IE}$ that is filtered with lfilter[1], t_i extracted based on $\dot{IE}$

5 Graph database

Knowledge graphs are typically created using a top-down approach, which involves the creation of an ontology that is then populated with data to create the KG. However, an ontology-based approach is time-consuming to initialize and update. Automating the acquisition, processing, and use of knowledge has high value when dealing with large amounts of diverse domain data [3]. Moreover, it is essential to consider the questions one wants to address with the data. We aim to introduce a data representation that allows us to answer CAE questions and works for complete crash FE models. These observations guided us to work with an evolving schema developed in a feedback loop.

In Fig. 5, we summarize the workflow for loading data into the graph database and exploring graph mining methods. When evaluating the workflow components, we aim to consider the reliability of the methods, particularly for a full-scale CAE crash FE model, as well as their computational efficiency. A primary concern is that the workflow components can detect slight and significant crash behavior deviations in view of the different changes engineers perform during the development stages.

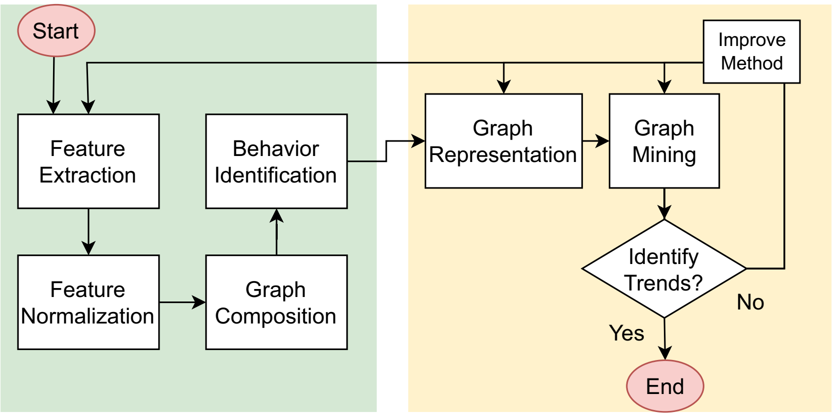
In this work, we build a first graph schema for car crash simulations to gain insight into their usage in the CAE-based development process. We consider a knowledge graph, as defined in [16], as a way to accumulate and convey knowledge of the real world. Its purpose is searchability and analysis of the data, which shall provide additional insight for the user obtained from the data. The current graph modeling addresses mainly two outputs, an intuitive graph analysis of CAE data, reflecting semantics or behavior, and a graph representation of the data that enables ML methods. Our workflow has two stages: first, we process the data and store it in a graph database, and second, we extract the computational graph from the database for visualization and ML studies. In this section, we describe the data schema.

5.1 Database schema

We now introduce our database schema, with building blocks illustrated in Fig. 6, where we aim to build a graph representation that allows the application of graph analytics and ML methods.

A (Sim) node represents a FE simulation outcome, where its properties stem from global features of the simulation, e.g., total mass or kinetic energy. The vehicle development and crash test protocols separate each simulation into a vehicle and a barrier/impactor. Observing that, we introduce nodes for the FE model (Model), barrier (Barr), and impactor (Imp) and consequently the relationships ○— SIM_MODEL —○ and ○— SIM_BARR/IMP —○, Fig. 6a. With this setup, different crash scenarios (load-cases) share

Fig. 5 Feedback process during car-graph development with a focus on the identification of trends in the simulations. Green zone: Stages of the construction and population of the graph representation. Orange zone: Usage of the graph representation and feedback loops to earlier stages



the same FE model from the vehicle design. Further, the focus is on the vehicle input independent of the load-case, i.e., barrier or impactor. Besides addressing different load-cases this choice also enables multidisciplinary design.

In addition, each part in a FE model and simulation is modeled as a Part node individually and connected to a Sim node with ○— INCL_PART —○ relations, Fig. 6b. A Part node contains information from its simulation states or the FE modeling level. Here, simulation states can consist of the energy absorption features that we defined in Section 4, while the FE modeling info consists of properties ID (PID), box center, material (name and ID), the center of gravity, or, when it applies, thickness (average and distribution). Geometrical features are a part's length, width, and height, along with the coordinate system of the FE model (L-x, W-y, H-z).

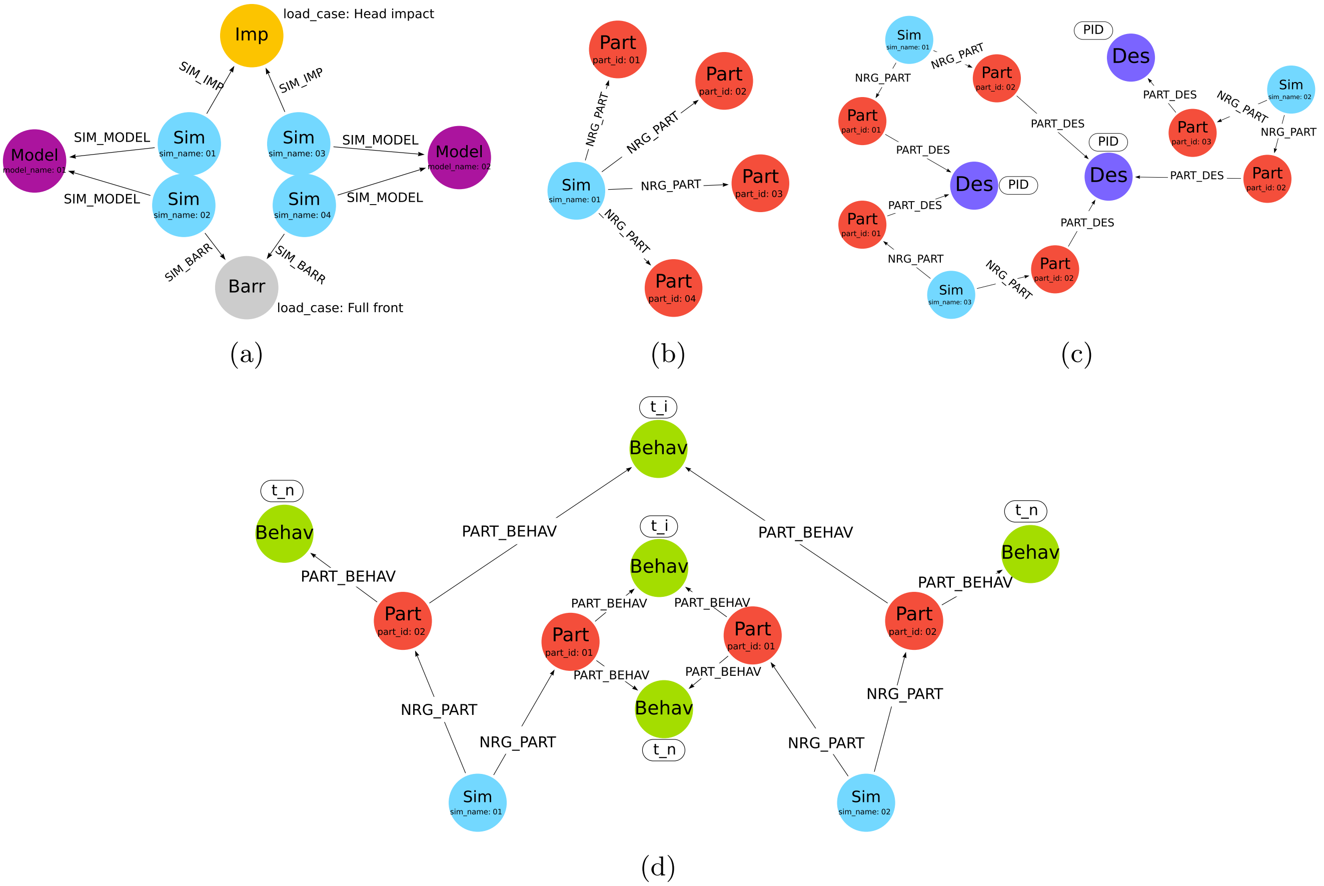


Fig. 6 Illustration of the building blocks of our graph modeling. (a) A high-level simulation consists of one model and one impactor/barrier depending on the crash scenario (b). More detailed components of a simulation are the parts that are filtered as energetic parts (c), while design and (d) behavior nodes connect parts of simulations based on the similarity of the design or the crash behavior, respectively

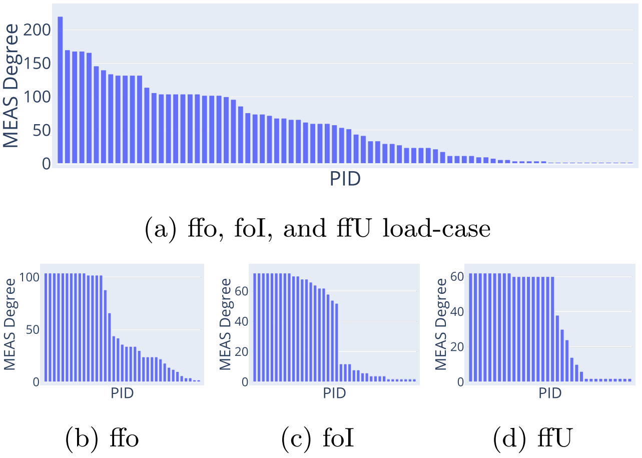
The basic schema so far is independent of any data analysis. Design Des and behavior Behav nodes contain outcomes from specific analysis steps, such as feature extraction or dimensionality reduction. Moreover, Des and Behav nodes are connected to Part nodes with ○— PART_DES —○, Fig. 6c, and ○— PART_BEHAV —○, Fig. 6d, edges, respectively. Here, a Des node collects parts that are similar at the FE modeling level, where in this work, the input for a ○— PART_DES —○ connection is the similarity of PIDs in one development stage. In contrast, a Behav node collects parts that show similar behavior during the simulation, where energy features, introduced in the next section, are used in this work for ○— PART_BEHAV —○ connections.

Moreover, we add two types of edges between simulations to the FE data model. Firstly, one usually knows the predecessor of a simulation configuration model based on the so-called development tree used in the CAE development process; we connect these two simulations with a ○— MODEL_REF —○ edge. Secondly, as a weighted edge, we introduce ○— SIM_SIM —○, where the weight refers to a similarity prediction between the simulations and is the outcome of a graph analytics algorithm, for example, SimRank [20]. Note, we study such similarity predictions and their usage as edge weights in a companion paper [29]. What we presented here is part of a more extensive graph modeling for automotive CAE data. We give further details of our graph modeling for CAE data, with a passive safety focus, in [28].

5.2 Query database

Graph analysis methods allow simple data explorations, discover non-trivial patterns in the data, and reveal behaviors. One of the used properties is the node's degree, and one can rank the nodes accordingly. A ranking of Des nodes extracts common parts in a development stage and reflects fundamental parts. Further, low degree Des nodes reflect components that are outliers or are in an essentially unexplored design space. A degree ordering of Behav nodes can, for example, extract common timings of behavior in a development stage, e.g., using the introduced energy features, which reflects essential times during the energy absorption. Such selection procedures allow automated post-processing scripts to support the CAE-ML workflow instead of requiring interactive user selections. High-ranked parts in a development stage for a load-case identify required parts in energy absorption. High-ranked parts are more reliable than just filtering the most energetic part in a simulation since there are outlier parts with high energy due to FE modeling errors.

We conjecture that a ranking of Des nodes, i.e., according to PID, can summarize information relevant to the engineering process for problem-solving. While the node degrees in real-world, large-scale networks often follow a power-law distribution, i.e., a fast decline in the degrees [27], we do not observe this for the Des nodes that reflect PIDs, Fig. 7. Here, the distributions for each load-case show some parts with high degrees, some in the middle, and the remainder with small degrees. Generally, high and low-ranked parts correspond to essential and outlier parts, respectively. Additionally, a significant degree drop can help identify the number of essential parts in a load-case. Additionally, the middle-degree parts are the ones that are not dominant in all the simulations and are neither outliers. Consequently, these parts are interesting structural components that potentially change the crash behavior and summarize the simulation design scenarios. Such middle-degree parts can be valuable input for inexperienced CAE engineers to identify parts that affect the crash behavior. For example, the fast transition of the degree in the foI load-case, Fig. 7c, compared to the rest, indicates that the number of parts affecting the load-case is limited, or that the engineer has performed a limited exploration of the design space.



(b) ffo (c) foI (d) ffU

Fig. 7 Degree distribution for the Des nodes for the CEVT data from the early stages

6 Scatter visualization

Data visualization is a key component in a typical data analytics project. The main aim of data visualization is to identify patterns and trends that are hidden behind the data. An explorative visualization of the data rather than descriptive analytics, which describes the data in a summarized way, provides a way for generating insights from the data [30].

Here, we propose several data visualization techniques for better data exploration of crash simulation data. In particular, we consider energy features from Section 4 as a data representation for each part, where we use a scatter plot for visualization. Each point in the scatter plot refers to one part (PID) of the simulation, and its coordinates are the part's energy features. This visualization allows for assessing the similarity of the underlying energy curves, identifying outlier parts, finding the similarity in component-wise crash behavior, and generating a visual DOE fingerprint for numerous simulations.

6.1 Curve similarity

The energy features were selected to extract the main properties of the energy curves. Therefore, they enable the assessment of the similarities of energy curves. For example, Fig. 8 shows a scatter plot for three pairs of parts from two different simulations and the corresponding energy curves. This figure indicates that the similarities of energy features of parts are related to the similarity of the corresponding curves. Consequently, the scatter plot of the parts' energy features facilitates visualizing clusters with similar behavior from an absorption perspective.

Note that for a three-dimensional visualization, it is more illustrative to have independent variables, which facilitates the separate investigation of each feature. Therefore, in the three-dimensional scatter plot, we employ Δt as the energy absorption time because t_n includes the effect of t_i, whereas Δt is independent of t_i. However, if one wants to include t_i information, using t_n in two-dimensional visualizations can be advantageous.

Generally, the weighted sum of the energy features can be used to measure the curve similarity. Here an open question is the normalization and weighting of the energy features, which likely also depends on the analysis goal in the application. For simplicity, we concentrate on visual exploration and individual energy features.

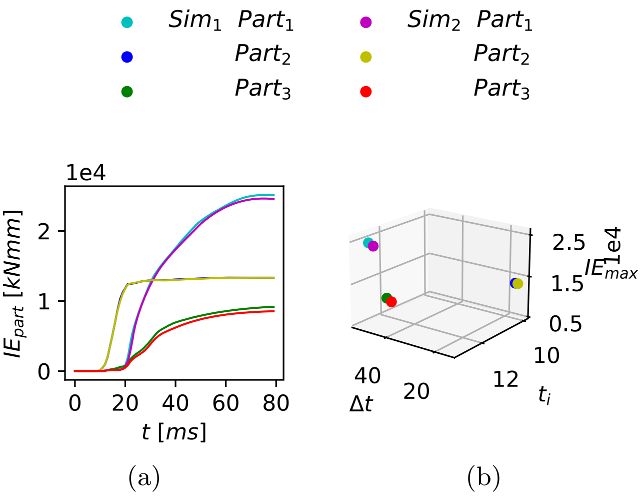


Fig. 8 IE curves similarity with scatter visualization (a) IE curves from three identical components in two simulations with the same load-case and (b) the scatter plot for their energy features

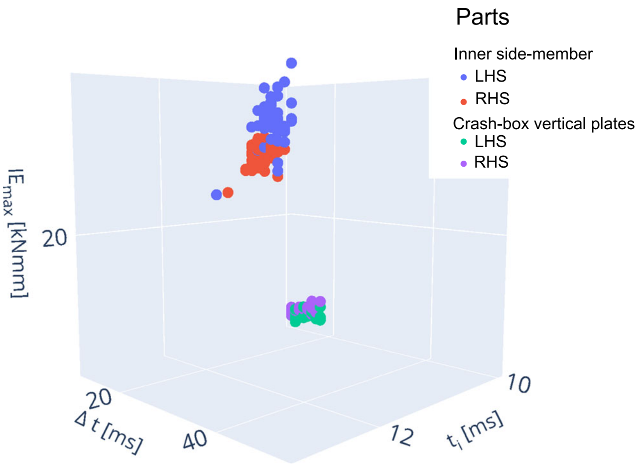


Fig. 9 Energy features for symmetric components, side-member, and crash-box plates. The coloring of the points is based on the semantics of the parts

6.2 Part similarity

Here we investigate the detection of geometrically corresponding components with energy features. We say that components are geometrically correspondent if they are located symmetrically in the vehicle, their undeformed geometries mainly overlap symmetrically, and their deformations are symmetrical. One straightforward use case is capturing similar energy absorption by symmetric parts of the vehicle structure in a full-frontal impact. The similarity is due to the almost symmetrical design of the vehicle on the left-hand side (LHS) and right-hand side (RHS). Moreover, the full-frontal load-case affects the LHS and RHS of the vehicle structure symmetrically.

Figure 9 illustrates this use case. It contains the four most energetic parts of 50 simulations of a full-front load-case in one development stage. A similar PID of the thereby selected parts implies that their geometries are more relevant than the remaining parts in the vehicles[4]. This data overview shows that the four most energetic parts generate two distinct point clusters. Each cluster holds two parts, and each pair consists of the RHS and LHS of the corresponding geometrical part.

As a final result, we observe that energy features detect symmetrical behavior in crash simulations. While here we imposed constraints on the data set, i.e., considered only one load-case and development stage, this holds more generally.

[4] Assuming the PID remains fixed during one development stage.

An example, which we further discuss in Section 6.3.1, is for distinct point clusters, where if the PID changes for a component, one can now connect components between different development stages.

6.3 DOE fingerprint

Summarizing the behavior of a DOE with many simulations is an additional application of energy features. We introduce a DOE fingerprint as a data visualization, which is the scatter distribution of the energy features. The scatter plot contains energy features from energetic parts of many simulations in one or several development stages. A DOE fingerprint of a group of simulations assists in assessing the vehicle's development process. We study four color schemes that visually group the data points differently during the exploration. The color schemes are according to PID, $IE_{\max}$ order, development stage, and load-case, respectively.

The color schemes reflect different use cases for the data exploration. The PID color scheme visualizes the design space for each part. Nonetheless, due to possible PID variations between load-case or development stages, the PID color scheme is limited to simulations in one development stage and one load-case. The second color scheme uses the $IE_{\max}$ order in a simulation, which visualizes the parts order in the energy absorption for each simulation. This visualization is informative if coupled with the PID color scheme to highlight the permutation of parts in absorption behavior. Additionally, the fingerprint with the development stage color scheme emphasizes load-cases in one/several development stages. Finally, the load-case color scheme demonstrates the evolvement of the platform in several/single development stages independent of PID change between several load-cases.

We now show examples of data visualization by DOE fingerprints for the real-life development stages from CEVT. These examples show the types of engineering information that a DOE fingerprint can visualize. To better demonstrate a three-dimensional plot in a two-dimensional figure, we present the DOE fingerprint as a matrix scatter plot; see Fig. 10. In matrix scatter plots, we use two features for absorption time (Δt and t_n) since the coupling between t_i and Δt is lost in a two-dimensional visualization. Additionally, the range of end-time or absorption period difference remains identifiable, i.e., when comparing the spread shape between different platform structures, by just considering t_n or Δt.

Note that an interactive application is the most helpful visualization for exploring the data using DOE fingerprints. For example, the application can enrich the data by connecting each point in the scatter plot to additional information such as pictures, deformation videos, or metadata of the part and simulation.

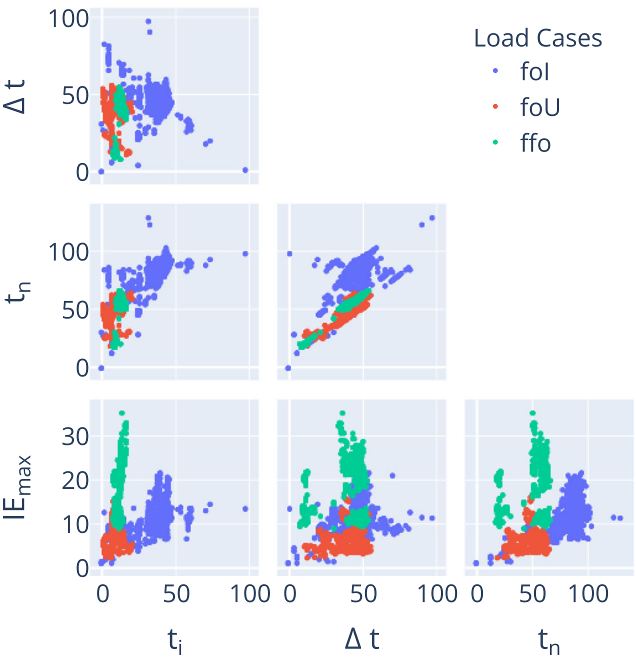


Fig. 10 DOE fingerprint load-case scheme, five most energetic components in simulations from four development stages. Simulation specification in Table 1 (t_i $[ms]$, Δt $[ms]$, $IE_{\max}$ $[kNmm]$)

6.3.1 PID scheme

The DOE fingerprint in each plot is an imprint of the distribution of the energy features independent of the PID. Consequently, the pattern shown by the PID color scheme conveys the parts between development stages even though the PID has changed. Figure 11 uses the PID scheme for an early and a middle development stage in a two-dimensional $IE_{\max} - t_n$ fingerprint. This visualization shows that even though the part numbering differs in these two development stages, the shape of the scatter plot and absorption order identify the pairwise components that correspond in energy absorption, see the point clouds (a) and (b) in Fig. 11 and Table 4. Here, cloud (a) consists of the inner plate of the side-member. For both stages, the cloud includes only 2 PIDs referring to the LHS and RHS parts. However, an offset along the y-axis shows a decrease in the mean of $IE_{\max}$.

Likewise, cloud (b) contains two components. The upper points belong to the subframe, and the lower ones to the outer wall of the side-member. However, this cloud holds many different PIDs. The variation of the PID for the subframe highlights the critical components studied in the CAE-based analyses.

Additionally, the cloud distribution shapes a pattern where it addresses the difference between development stages, e.g., a change in the FE modeling technique or a change in the vehicle concept. In this example, the vertical and horizontal plates of the crash-box have separate PIDs for RHS and LHS. However, these are modeled as one in the

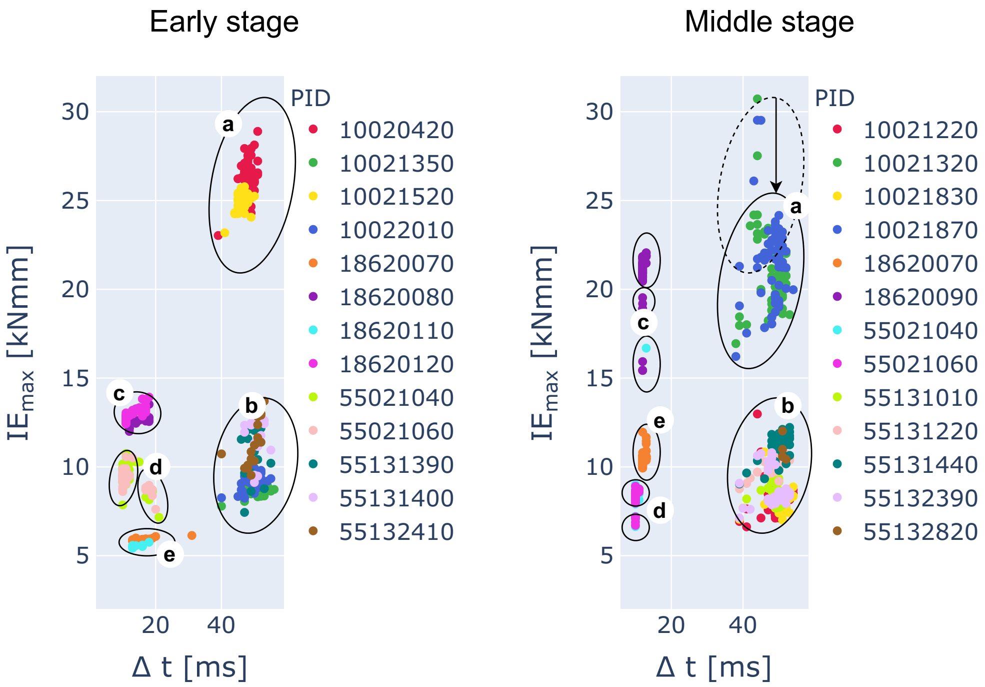


Fig. 11 DOE fingerprint with PID color scheme, CEVT data ffo load-case, Part name in Table 4

mid-stage. Consequently, the absorption has doubled; see clouds (c) and (e) in Fig. 11.

Finally, point cloud (d) belongs to the lower load path components RHS and LHS. It keeps its dual behavior, but this visualization summarizes that the absorption interval is more stable in the later stage.

6.3.2 Order scheme

The ordering scheme visualizes the $IE_{\max}$ order for each part in a simulation. The ordering scheme visualizes the point cloud for the energy absorption order combined with the PID scheme. Figure 12 compares the ffo

Table 4 PID part name in two development stages for Fig. 11

Early stage	Middle stage	Part Name (Cloud Label)	
10020420	10021870	LHS-I	(a)
10021520	10021320	RHS-I	side-member
10022010	10021830	LHS-O	
10021350	10021220	RHS-O	(b)
18620080	18620090	LHS-V	(c)
18620120		RHS-V	crash-box
18620070	18620070	LHS-H	
18620110		RHS-H	(e)
55021040		LHS	lower load path (d)
55021060		RHS	
55131390, 55132410	55132390, 55131220	LHS	subframe (b)
55131400	55131440, 55132820, 55131010	RHS	

RHS: right-hand side, LHS: left-hand side

-U: upper, -V: vertical, -H: horizontal, I: inner, O: outer

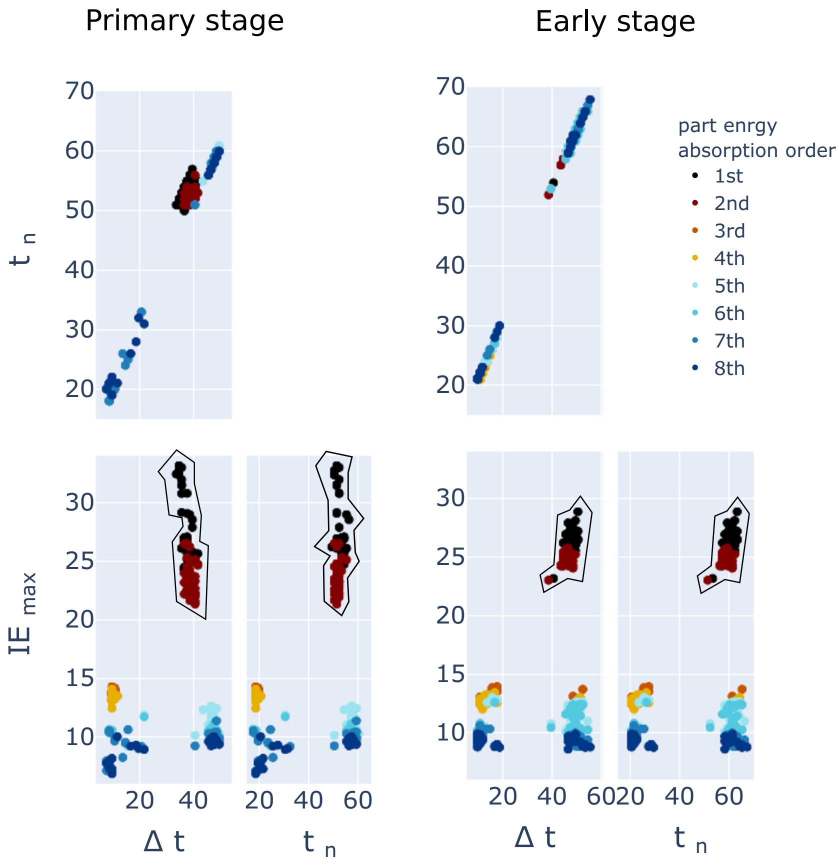


Fig. 12 DOE fingerprint with order scheme, ffo load-case considering eight most energetic parts for each simulation. Primary and early development stages 50 and 53 simulations, respectively. (t_n $[ms]$, Δt $[ms]$, $IE_{\max}$ $[kNmm]$)

load-case in two development stages using the $IE_{\max}$ order scheme for each simulation's eight most energetic parts. The number of point clouds for each placement captures the number of scenarios for evaluating the permutation of the energy absorption (e.g., for third-order, it is one and two, respectively, in the primary and early-stage). In the primary stage, bifurcation exists for the sixth, seventh, and eighth-ordered components; however, in the early stage, bifurcation starts right after the second part. Besides the number of scenarios, the density of the point clouds can reflect outlier simulations or an unexplored design space. For example, a few simulations in the early stage have the fifth and sixth parts in the left point cloud.

So far we looked at $IE_{\max}$, t_n, and Δt features. Additionally, the t_i fingerprint achieves a different knowledge summarization. Figure 13 shows the initial time for the same development stages as Fig. 12. Here we see that the two most energetic parts, the side-members, have noticeable

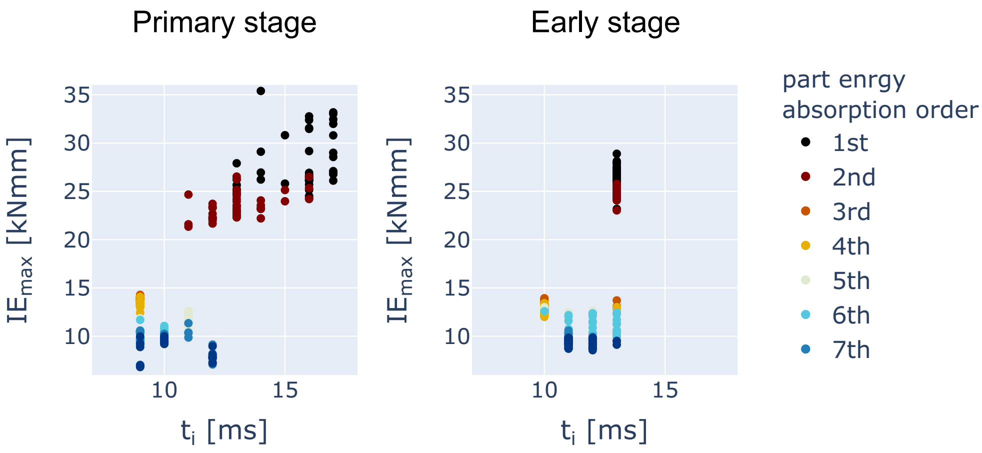


Fig. 13 Initial time distribution for Fig. 12, (t_i $[ms]$, $IE_{\max}$ $[kNmm]$)

differences in the t_i spread. The deviation is also captured in the $t_n - \Delta t$ plot, Fig. 12. The early development stage is more stable in trigger time than the primary development stage and limits the DOE. Consequently, the t_n and Δt relation becomes more linear. Therefore, $IE_{\max} - \Delta t$ and $IE_{\max} - t_n$ provide similar DOE fingerprints in the early stage. However, in the primary stage, the relation of t_n and Δt is non-linear for the side-member. Consequently, the point cloud shape of $IE_{\max} - \Delta t$ and $IE_{\max} - t_n$ differs in the primary development stage.

6.3.3 Development stage scheme

This coloring scheme is beneficial for summarizing the trends of the development stages. In this visualization, t_n is preferable to Δt since an absolute value is better for comparing development stages. Figure 14 shows the pair-wise comparison of four development stages with the development stage scheme coloring. In summary, remarkable detections are:

a) The initial time absorption span has been the smallest for the early development stage, and absorption initialization varies a lot for the rest.
b) The inner side-member part with the highest $IE_{\max}$ has been declining in the maximum absorbed energy during the development.
c) The 2d visualization overlays point clouds in initial absorption time.
d) The inner side-member stays almost steady in absorption time spread.
e) The spread of absorption time declines as the development stages evolve for the rest of the parts.

6.3.4 Load-case scheme

This visualization enables the comparison of DOEs between load-cases, which supports detecting multi-disciplinary development challenges with different crash requirements. Figure 10 is a matrix scatter plot for three load-cases of the front crash in four development stages of the CEVT data with a load-case scheme, Table 1. It includes 611 simulations with five parts with high-ranked $IE_{\max}$. The visualization indicates that the ffo load-case has discontinuous absorption compared to the other two. This gap exists for t_n values that make two clusters: early ($\approx$ 10 ms) and late absorption ($\approx$ 60 ms).

7 Graph visualization

We now investigate graph visualization techniques for knowledge discovery in simulation studies, where we use energy features as weights in these graphs. Visual exploration in an interactive way allows one to apprehend the underlying graph and thereby gain insight. Visual representations of graphs can be classified into three major categories: node-link diagrams, matrix representations, and hybrid methods. Here we focus on the application of node-link diagrams. Among node-link diagram methods, the most widely used are force-directed layout algorithms [6]. They

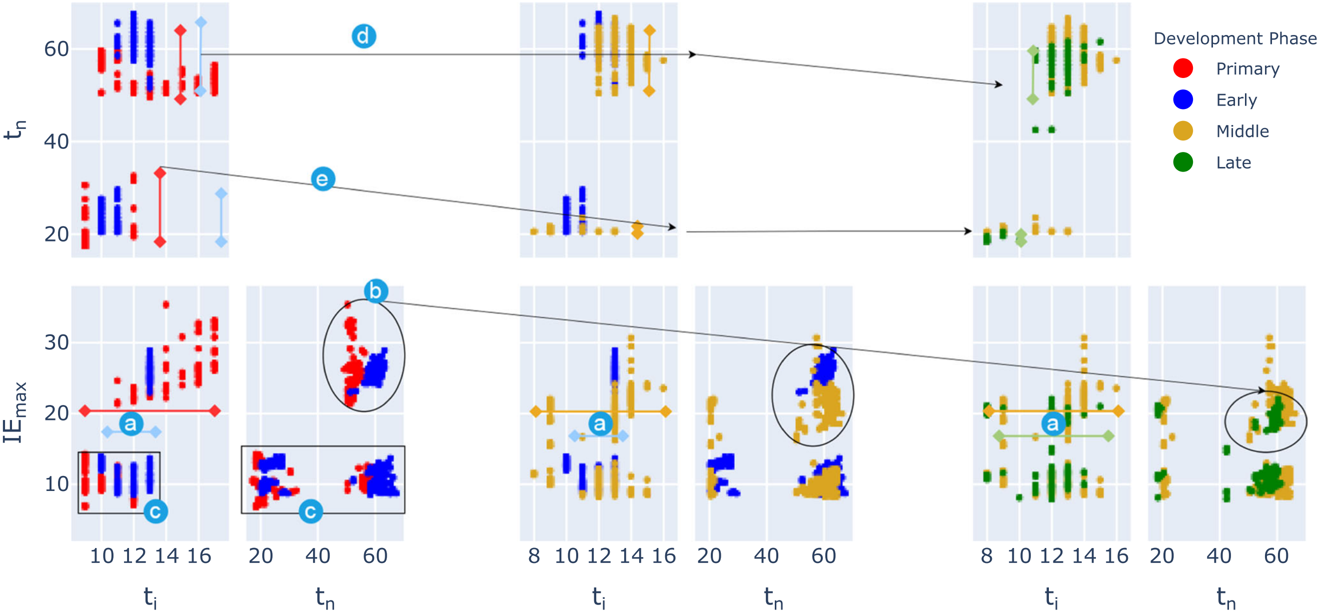
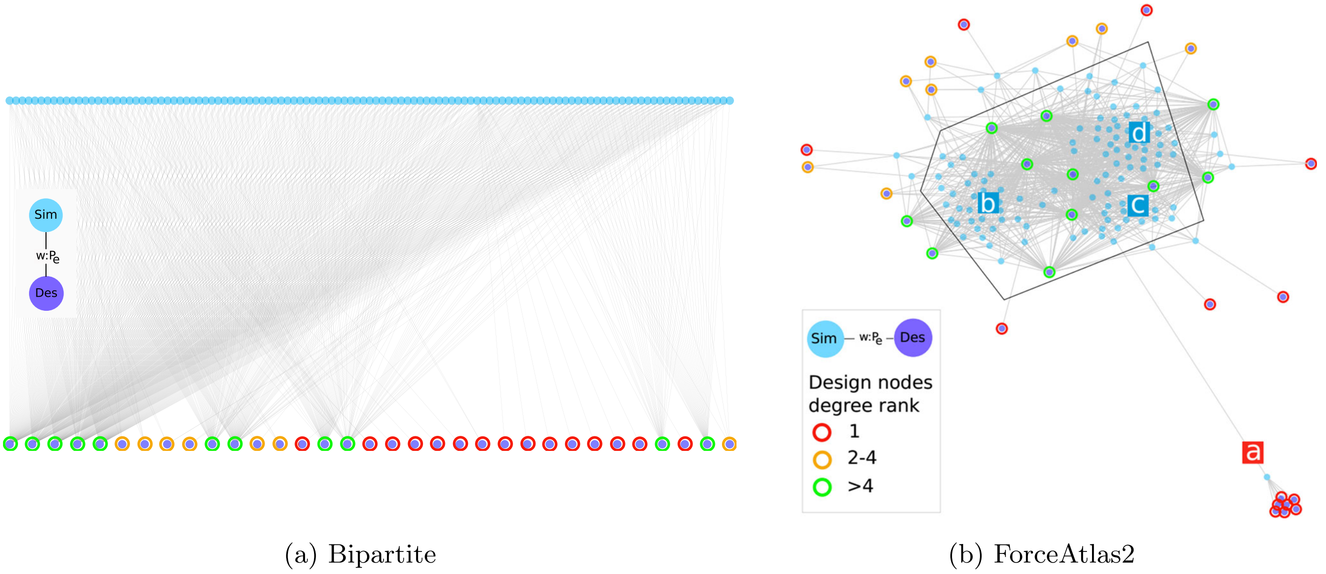


Fig. 14 DOE fingerprint, development stage scheme, ffo load-case CEVT data. Pair-wise comparison for four different development stages. Primary, early stage respectively 50, 53 simulations, Midd-stage, 82 simulations, Late stage, 28 simulations, for 7th most energetic parts(t_i [ms], Δt [ms], $IE_{\max}$ [$kNmm$])



(a) Bipartite (b) ForceAtlas2

Fig. 15 Information extraction with graph visualization, foI load-case in the primary stage, 115 Sim simulation nodes with eight Des nodes for each simulation and weighted with P_e. Design nodes' edge colors are based on the nodes' degree

have often been preferred over other algorithms since the 1980s. Force-directed algorithms can be divided into classical and hybrid algorithms according to their characteristics and computational modeling. Classical force-directed algorithms are usually based on physical laws, specifically in ways that simulate a spring system. For large and complex networks, hybrid force-directed algorithms are designed, which use heuristics to improve the performance of classical force-directed algorithms [8]. Classical algorithms are still suitable in our case due to the relatively small size of the graph.

Following the survey [8] the three methods of Fruchterman–Reingold [15], Kamada–Kawai [22], and ForceAtlas2 [19] are suitable for our purposes. We investigated the three methods on our data, where ForceAtlas2 showed the best results[5]. In general, more successful force-directed techniques are those that have avoided certain principles to show off other structural properties of the graph, such as ForceAtlas2 [19]. The method is still following the idea of a physical system, but the principle the authors have tried to optimize is one of clustering rather than being concerned with edge lengths or uniform node distributions, for example. In the following, we summarize the use of such graph visualization methods.

Here, we extract a sub-graph by using nodes with Sim — SIM_DES — Des edges. The result is a bipartite graph consisting of two types of objects, namely Des and Sim nodes. The edges of the Sim — SIM_DES — Des bipartite graph are weighted by the energy power absorption ($P_e = IE/\Delta t$), which can be seen as an aggregation of energy features.

Due to the widespread energy power absorption, specifically in the networks that include outlier simulations, it is challenging to get an interpretable view of the network. Consequently, from the available options to improve the graph visualizations, we deactivate the gravity option to simplify the equibalance of the forces. Instead, we study two options of this method, scaling ratio and edge weight influence. The scaling ratio R refers to the repulsion required and is claimed to result in a more sparse graph. Furthermore, the edge weight influence e_{inf} scales from zero, for no weight influence, to one as normal.

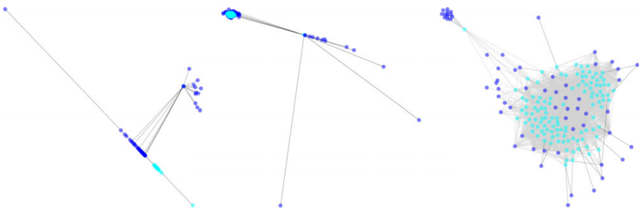
Note that our graph is relatively small compared to graphs in many other domains, with less than 26000 nodes considering all the types. Consequently, the primary computational time is loading the data to the graph database, which is done offline in a pre-processing step. The ForceAtlace2 calculation depends on the number of included nodes and the needed iterations. For our data, both only take a couple of seconds. The timings for the rest are less than a second, which overall makes it easy to explore the data interactively.

The visualizations presented in the following are for three scenarios:

- one load-case in one development stage,
- different load-cases in one development stage,
- one load-case in several development stages.

All approaches mentioned in the following are practical options for an interactive user interface to assist engineers in data cleaning and knowledge discovery.

[5] We use the ForceAtlace2 implementation available at https://github.com/bhargavchippada/forceatlas2.



(a) $e_{inf} = 2$ (b) $e_{inf} = 0.5$ (c) $e_{inf} = 0.02$

Fig. 16 Improving visualization by varying edge weight influence, e_{inf}, from 2 to 0.02. Case one, foI load-case in the primary development stage

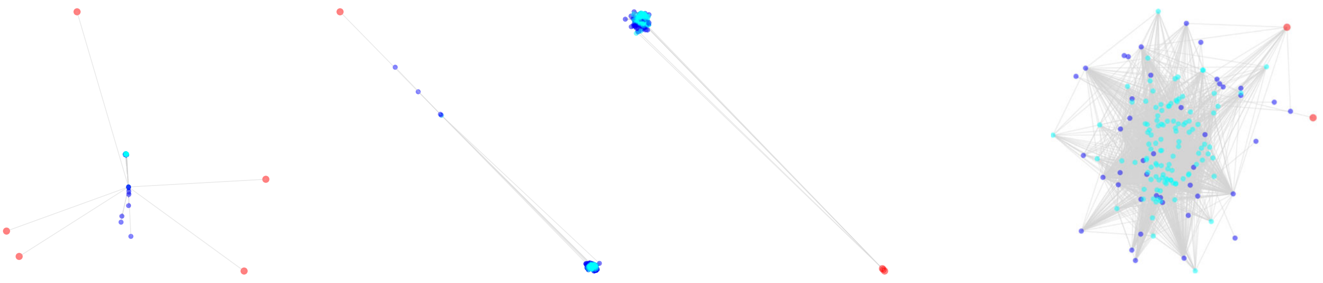
In the first case study, we consider the eight most energetic parts for 115 simulations in a primary development stage and foI load-case. We visualize the bipartite graph in Fig. 15a. The graph has 115 Sim and 33 Des nodes. The number of design nodes is more than eight due to differences in the most energetic parts of the simulations. This basic visualization can only distinguish the density of the degree of Des nodes.

However, the ForceAtlas2 method reveals more information about this network, Fig. 15b. By positioning them off-centered, this visualization emphasizes the outlier Des and Sim nodes. Most of the Des outliers are related to the connection modeling, which is very sensitive to the modeling. Therefore, the solver tries to rectify it, which causes high internal energy in the corresponding connection part. However, connections are not the study object of these FE simulations. As a result, these are unreliable simulations and designs.

Additionally, the Sim nodes are distributed based on their similar structural connection to Des nodes, which shapes clusters of simulations, zones (b), (c), and (d) in Fig. 15b. From a simulations clustering perspective, ForceAtlas2 has an outstanding result. The other two methods, Fruchterman–Reingold and Kamada–Kawai, separate only the outlier Des nodes. The Des nodes located in the center of simulations are the Des nodes with the highest degree. These nodes are essential parts of most of the simulations. This visualization highlights that Des nodes cause the split of the simulations point cloud. In this example, each cluster has several design nodes positioned outwards and with high degrees, Fig. 15b green nodes. Additionally, there are simulations further away from the central simulation clouds. This can indicate less explored design space, Fig. 15b orange nodes.

The initial visualization of ForceAtlas2 has the edge influence set to zero, which means we see mostly the structural effect of the network, Fig. 17. There are several ways to get a good visualization, including the weights. One is to take advantage of the ForceAtlas2 method options. The other approach is deactivating the distant nodes, which are outliers. For initial visualization, we look at only eight designs. Next, we increased the designs to 20 for each simulation, ending with 62 designs. First, we consider two options of e_{inf} and R from ForceAtlas2 to improve the visualization. Figure 16 summarizes the effect of edge weight influence for the network above. This figure visualizes that by decreasing the edge weight influence, we can keep the whole DOE graph and have the resolution in the graph's structure. Consequently, we still see outliers and DOE clusters, similar to Fig. 17. Note changing the scale factor does not seem to have a noticeable influence on the resulting visualization of our data.

The next option is to remove outliers, where we take an iterative approach to identify and remove distant nodes using the ForceAtlas2 algorithm. In each iteration, we calculate all edge lengths based on the ForceAtlas2 positioning of the nodes. Afterwards, we remove the edges with a length higher than a specified threshold. We remove the disconnected nodes before we recalculate the positions for the reduced network in the next iteration. Figure 17 visualize several iteration steps in which we removed the distant nodes with a thresholding factor of 0.8. Here, the red nodes are the identified outliers, and we remove them before the next iteration. Thus, we can quickly identify the outliers and clean the data. In this method, there is no consideration



(a) itr 1 (b) itr 3 (c) itr 5 (d) itr 7

Fig. 17 Odd stages when iteratively removing distant nodes from the foI load-case in a primary stage, 115 Sim simulation nodes with 20 Des nodes for each simulation and weighted with P_e, The red nodes are the outliers identified to be removed. Case one, foI load-case in the primary development stage, the threshold of 0.8, $e_{inf} = 1$, $R = 1$

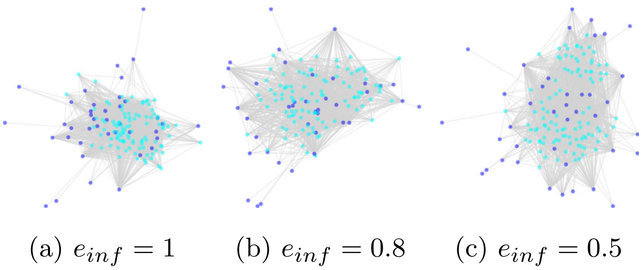
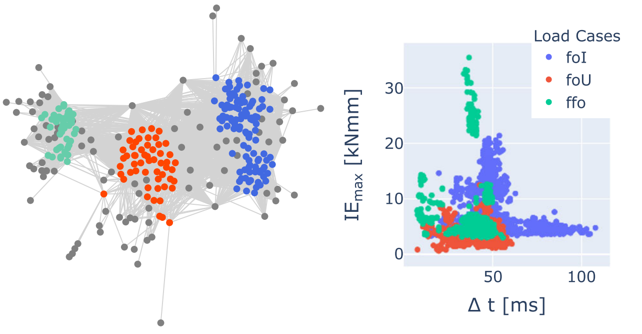

(a) $e_{inf} = 1$ (b) $e_{inf} = 0.8$ (c) $e_{inf} = 0.5$

Fig. 18 Improving visualization for network without outlier simulations with varying edge weight influence, e_{inf}, from 1 to 0.5. Case one, foI load-case in the primary development stage

of the labeling of the nodes, so the identified nodes are either Des or Sim.

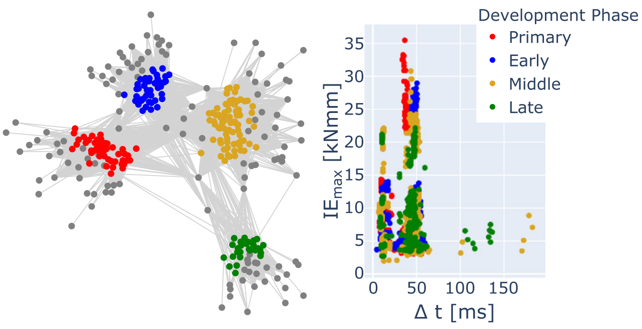
As the final investigation for this case, we consider the visualization with ForceAtlas2, where we exclude the outliers simulations from the graph, Fig. 18. The difference between this visualization to Fig. 17 is that in removing the nodes, we consider the node labeling, and we remove the Sim and its corresponding edges and, consequently, the disconnected nodes. In this visualization, keeping the edge influence smaller than 0.5 allows us to detect the clusters. This factor may differ depending on the intensity of the graph and its node numbers. Next, we look into two other study cases of graph visualization.

The example for different load-cases in one development stage includes the primary development stage of three load-case of foI, foU, and ffo, Fig. 19a. This graph contains 20 Des nodes for each Sim node, resulting in 204 simulations and 94 designs. We exclude outlier simulations and corresponding design nodes from this network. Figure 19 compares the ForceAtlas2 visualization with the DOE fingerprint. Similar to Fig. 10, this visualization indicates that the ffo load-case contains a limited design of experiments. An additional discovery based on comparing to the DOE fingerprint is the relation of these three load-cases with each other: foI and foU have more in common than the ffo load-case. This can be visualized with several design nodes between blue and red point clouds vs. green and red. Furthermore, the Des nodes between each load-case cluster identify the essential parts in common.



(a) ForceAtlas2 (b) DOE fingerprint

Fig. 19 Case two, several load-cases in the primary development stage. (a) Des nodes are colored in gray and Sim nodes are following the scatter plot coloring (b). $R = 1$ and $e_{inf} = 0.5$



(a) ForceAtlas2 (b) DOE fingerprint

Fig. 20 Case three, ffo load-case in several development stages. (a) Des nodes are colored in gray and Sim nodes are following the scatter plot coloring (b). $R = 1$ and $e_{inf} = 0.5$

The last case study is one load-case (ffo) in several development stages, Fig. 20a. This graph also considers 20 Des for each Sim, including 196 simulations and 123 designs. Besides the scarcity of each DOE compared to others, it is possible to discover the essential parts in common between different development stages. The ones in the center are in common for several stages. Moreover, the late development stage, green cloud, differs more from the other three. The DOE fingerprint in Fig. 20b visualizes this difference with the lowest absorption feature of that development stage. An important note is that the commonality of Des nodes between different development stages is uncertain. However, the DOE fingerprint is independent of this assumption for comparing parts. We present this network as a potential for knowledge discovery and note that better semantics for Des nodes are required than the used PID.

In the last two use cases, we set $e_{inf} = 0.5$ to visualize the clustering of the parts. However, due to the increase in the graph constraint by having several DOEs included, this value needs to be higher to highlight the additional outlier. A good example is the parts in the scatter plot in Fig. 20b whose absorption time is more than 100[ms]. These parts are not as noticeable as in Fig. 20a. However, increasing e_{inf} makes these parts more outstanding.

8 Conclusion and outlook

The complexity of the raw data from simulations and the lack of semantics in the current vehicle development

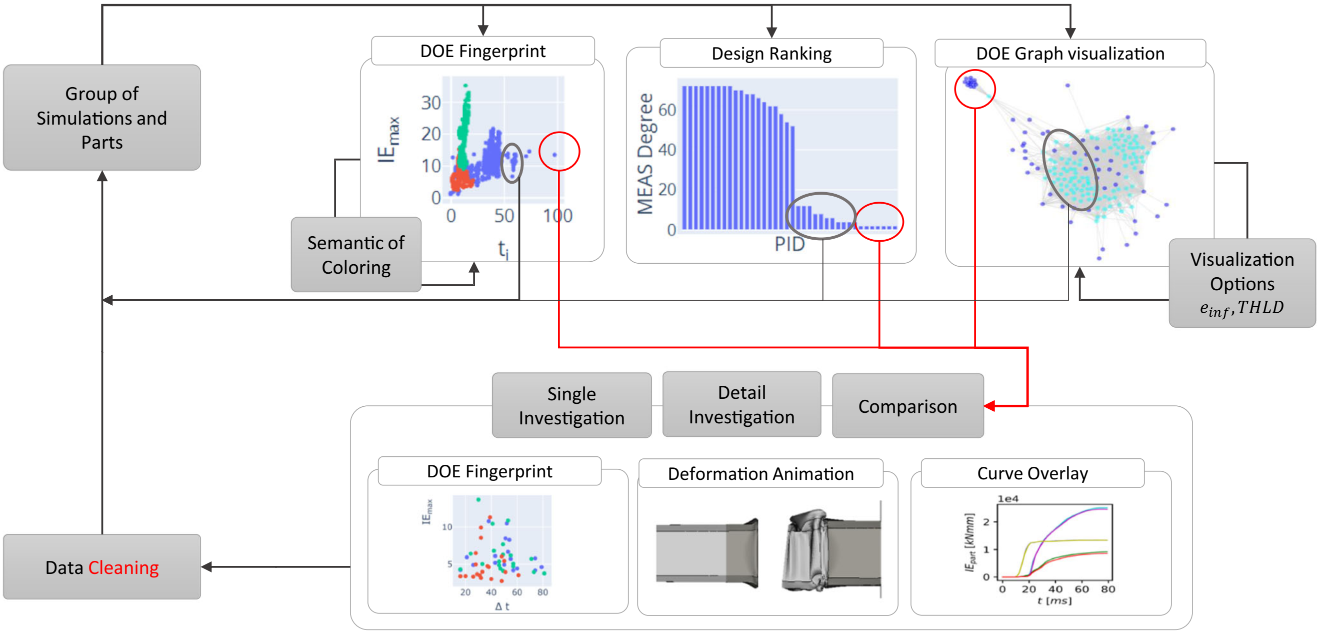


Fig. 21 The workflow of the knowledge discovery assistant for dynamic reporting

workflow causes design engineers and attribute leaders to rely on the reporting from the CAE engineers working with them. However, such static reporting restricts an independent exploration of the data. The lack of semantics in CAE data makes the data disconnected and hinders multi-disciplinary collaboration, which degrades efficient problem-solving. Disconnected data in an OEM, and even more between OEMs, is one of the obstacles we aim to address with the car-graph vision for an efficient data exploration that exploits semantics.

Our research aim for this work was to introduce semantics for crash simulations, which enables searchability or filtering of FE crash simulations. Based on graph representations of the data, we proposed energy features and used them for data visualization while leveraging them as weights in the data graph to empower knowledge discovery. We showed the sensitivity of energy features for differentiating FE crash simulations during development stages. Moreover, it introduces a simple way of filtering the necessary parts to be studied in ML deformation-based workflows. Besides, applying ForceAtlas2 visualization further empowered outlier detection, data cleaning, and the clustering of the parts and simulations. This visualization allows vehicle DOE knowledge discovery, e.g., by assessing a single load-case in one development stage and comparing different load-cases and development stages.

Overall, DOE fingerprint, design ranking, and graph visualization are three new visualization concepts for CAE data and allow further knowledge discovery[6]. In a broader view, we envision a web-based platform to enable semantic reporting for CAE[7] as a practical tool, which targets CAE attribute leaders, CAE engineers, design engineers, and data analysts in automotive R&D. It should enable project members from different teams to access the CAE results, understand the design performance limitations, compare simulations, and use algorithms on the car-graph. For example, we can support a data exploration with two- and three-dimensional views of DOE fingerprints (Section 6.3). Interpreting a DOE fingerprint involves further investigation, where a dynamic interaction and filtering facilitate the data exploration. For example, each scatter point can link to the corresponding energy curve, metadata, pictures, and deformation videos of the simulations/parts.

Figure 21 summarizes the interactions of such a workflow. Here design ranking and DOE graph visualization was the use case of trend and outlier detection at a high level. In comparison, the DOE fingerprint can find some extreme outliers and is best used for summarizing the exploration and more detailed investigations.

However, it is still an early stage for research on a vehicle knowledge graph, and additional data should be

[6] The databases example and a user tutorial are at https://github.com/Fraunhofer-SCAI/GAE-vehicle-safety

[7] Accessible at https://CAEWebVis.scai.fraunhofer.de/.

loaded into the graph database to enrich it. For example, in the model level as the input, it will be material, geometrical semantics, grouping the parts as functional components. Moreover, examples of extra data in the simulation output are accelerometers, cross-section forces, deformations, and safety requirements. Moreover, there is a necessity for grouping parts and features, where, for example, a higher level of grouping may enable load-path detection. Furthermore, link prediction and similarity assessment will support engineers in exploring a simulation database to search the data. We give examples for similarity assessment or part grouping in a companion paper [29].

For graph visualization, additional improvements can still be made. One additional study can be on extending the types of nodes and relations included in the network. For example, including development tree connections, impactor/barrier nodes, and simulation similarity predictions. The edge bundling method can also reduce the visual clutter caused by edge overlaps. It can provide a global overview of complex connection graphs while providing information on the primary connection relationships in the graph by the thickness and color of the edges [7].

A mid-term goal is to predict simulations for unexplored design spaces and recommend solutions to the engineer. Likewise, predicting cause-effect relations between the model and simulations will further enrich the data. Finally, a long-term target is to enable the evaluation of performance robustness. Correspondingly, the car-graph shall allow an extension of the safety evaluation from regulated tests, which are just examples of real crash scenarios, to more diverse crash scenarios. Finally, the handling of unlabeled data still remains a big challenge in this domain. In our case, we needed to verify the overall discoveries with an inefficient process of manual engineering feedback that limited ML applications. Consequently, empowering the semantics using web technology to increase data labeling will further support the uptake of analysis approaches in this application domain.

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Conflict of 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.

References

1. Abu-Salih B (2021) Domain-specific knowledge graphs: a survey. J Netw Comput Appl 185:103–076
2. Belaid MK, Rabus M, Krestel R (2021) Crashnet: an encoder–decoder architecture to predict crash test outcomes. Data Min Knowl Disc 35(4):1688,1709
3. Bharadwaj AG, Starly B (2022) Knowledge graph construction for product designs from large CAD model repositories. Adv Eng Inform 53:101,680
4. Boussuge F, Tierney CM, Vilmart H et al (2019) Capturing simulation intent in an ontology: CAD and CAE integration application. J Eng Des 30(10-12):688–725
5. Buchgeher G, Gabauer D, Martinez-Gil J et al (2021) Knowledge graphs in manufacturing and production: a systematic literature review. IEEE Access 9:55,537–55,554
6. Chen W, Guo F, Han D et al (2018) Structure-based suggestive exploration: a new approach for effective exploration of large networks. IEEE Trans Visual Comput Graph 25(1):555–565
7. Chen Y, Guan Z, Zhang R et al (2019) A survey on visualization approaches for exploring association relationships in graph data. J Vis 22(3):625–639
8. Cheong SH, Si YW (2020) Force-directed algorithms for schematic drawings and placement: a survey. Inf Vis 19(1):65–91
9. Du X, Zhu F (2018) A new data-driven design methodology for mechanical systems with high dimensional design variables. Adv Eng Software 117:18–28. https://doi.org/10.1016/j.advengsoft.2017.12.006
10. Du X, Zhu F (2019) A novel principal components analysis (PCA) method for energy absorbing structural design enhanced by data mining. Adv Eng Software 127:17–27. https://doi.org/10.1016/j.advengsoft.2018.10.00 , using pca to generate simulation
11. Du Bois P, Chou CC, Fileta BB et al (2004) Vehicle crashworthiness and occupant protection. Am Iron Stell Inst:27–280, 304–330
12. Fatfouta N, Stal-Le Cardinal J (2020) Towards a framework for integrated and collaborative knowledge management for engineering design – a case study. Proc Des Soc Des Conf:559–568, https://doi.org/10.1017/dsd.2020.136
13. Fatfouta N, Stal-Le Cardinal J, Royer C (2019) Empirical study of car crash simulation analysis within the development phase. Proc Int Conf Eng Des ICED:2843–2852, https://doi.org/10.1017/dsi.2019.291
14. Feng Q, Zhou X, Li J (2020) A hybrid and automated approach to adapt geometry model for CAD/CAE integration. Eng Comput 36:543–563. https://doi.org/10.1007/s00366-019-00713-4
15. Fruchterman TMJ, Reingold EM (1991) Graph drawing by force-directed placement. Force-Directed Placement. Softw-Pract Exper 21(11):1129–1164
16. Hogan A, Blomqvist E, Cochez M et al (2021) Knowledge graphs. ACM Comput Surveys (CSUR) 54(4):1–37
17. Huet A, Pinquié R, Véron P et al (2021) Cacda: a knowledge graph for a context-aware cognitive design assistant. Comput Industry 125:103–377

18. Iza-Teran R, Garcke J (2019) A geometrical method for low-dimensional representations of simulations. SIAM-ASA J Uncert Quantif 7(2):472–496. https://doi.org/10.1137/17M1154205
19. Jacomy M, Venturini T, Heymann S et al (2014) Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software. PloS one 9(6):e98–679
20. 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
21. Johansson J, Contero M, Company P et al (2018) Supporting connectivism in knowledge based engineering with graph theory, filtering techniques and model quality assurance. Adv Eng Inf 38:252–263. https://doi.org/10.1016/j.aei.2018.07.005
22. Kamada T, Kawai S (1989) An algorithm for drawing general undirected graphs tomihisa kamada and satoru kawai. Inf Process Lett 31(1):7–15
23. Kestel P, Kügler P, Zirngibl C et al (2019a) Ontology-based approach for the provision of simulation knowledge acquired by data and text mining processes. Adv Eng Inf 39:292–305. https://doi.org/10.1016/j.aei.2019.02.001
24. Kestel P, Kügler P, Zirngibl C et al (2019b) Ontology-based approach for the provision of simulation knowledge acquired by data and text mining processes. Adv Eng Inform 39:292–305
25. Kirkwood R, Sherwood JA (2018) Sustained CAD/CAE integration: integrating with successive versions of step or iges files Eng Comput:34, https://doi.org/10.1007/s00366-017-0516-z
26. Li X, Lyu M, Wang Z et al (2021) Exploiting knowledge graphs in industrial products and services: a survey of key aspects, challenges, and future perspectives. Comput Indust 129:103–449
27. Newman ME (2005) Power laws, Pareto distributions andZipf's law. Contemp Phys 46(5):323–351
28. Pakiman A, Garcke J (2022) Graph modeling in computer assisted automotive development. 2022 IEEE international conference on knowledge graph (ICKG), ArXiv:2209.14910
29. Pakiman A, Garcke J, Schumacher A (2022) SimRank-based prediction of crash simulation similarities. Institut für Numerische Simulation, Universität Bonn
30. Srinivasa K, Siddesh G, Srinidhi H (2018) Introduction to data visualization. In: Network data analytics. Springer, pp 321–331
31. Ziegler J, Reimann P, Keller F et al (2020) A graph-based approach to manage CAE data in a data lake. Procedia CIRP 93:496–501