Complexity Indicators and Their Impact on Algorithm Performance in Automotive Part Selection

Daniel Bischoff[1,*], Tobias Nerz[1] and Kaan Ekiz[1,2]

[1]*Mercedes-Benz AG, Leibnizstraße 6/1, 71032 Böblingen, Germany*
[2]*Aalen University – Engineering, Business and Health, Beethovenstraße 1, 73430 Aalen, Germany*

Abstract

This paper presents a comprehensive study on complexity indicators and relative scaling behavior of an algorithm designed for visualizing and changing part selection data in the automotive industry. The algorithm transforms Boolean formulas into multi-way decision diagrams within the context of a tool called POSEIDON. The case study is based on real industrial configuration data from a leading German automotive manufacturer, highlighting the challenges posed by high variant diversity and logical interdependencies.

To characterize the structural and distributional properties of the dataset, various complexity indicators are introduced and computed, including Shannon entropy and average path lengths in idealized decision diagrams. These metrics are then used to evaluate the behavior and effectiveness of the POSEIDON algorithm in generating compact, interpretable, and scalable decision diagrams. The results show that the average path length in the POSEIDON-generated diagrams scales linearly with entropy-based measures, confirming the adequacy of the algorithm in an environment where we've observed rising complexity year over year.

Keywords

Complexity Indicators, Algorithm Performance, Automotive Part Selection, Variant Configuration, Configuration Rules, Industrial Data

1. Introduction

Product configuration, particularly part selection, is a critical task in the automotive industry, based on vast and intricate datasets derived from manufacturing and engineering processes. The inherent complexity of such industrial data poses significant challenges for configuration systems. Understanding and quantifying this complexity is essential not only for benchmarking existing tools but also for improving their performance. Furthermore, evaluating how algorithms behave under realistic levels of complexity is crucial for ensuring their practical applicability.

Building on prior published work by the author(s) [1], this paper evaluates an algorithm that transforms Boolean formulas, used to encode feature-to-part relationships in automotive product lines, into multi-way decision diagrams. Unlike traditional binary decision diagrams (BDDs) [2], these structures consist of decision nodes representing feature choices with multiple discrete options (plus an explicit 'otherwise' branch encompassing remaining choices) and utilize the actual selected parts as terminal nodes, directly reflecting the configuration outcome. A key aspect distinguishing this approach is its application context: the decision diagrams, generated within the *POSEIDON* tool, serve as the primary interactive visualisation. Data engineers use them to create, analyse, and modify configuration logic, with a strong focus on comprehensibility and correctness by construction. This paper presents an in-depth case study utilizing authentic industrial data from this automotive context. Our primary contributions are twofold: first, we propose and measure complexity indicators designed to capture characteristics of industrial Bill of Materials, focusing on the combinatorial solution space described by their variation points. Second, we evaluate the performance of the decision diagram generation algorithm on our data, primarily focusing on the compactness of the resulting diagrammatic representation. For brevity, we will henceforth refer to this simply as *the algorithm*.

ConfWS'25: 27th International Workshop on Configuration, Oct 25–26, 2025, Bologna, Italy
[*]Corresponding author.
Daniel.Bischoff@Mercedes-Benz.com (D. Bischoff); Tobias.Nerz@Mercedes-Benz.com (T. Nerz); Kaan.Ekiz@Mercedes-Benz.com (K. Ekiz)

While various techniques exist for representing and visualising product configuration logic—including tabular formats, feature models, Karnaugh maps, and tree-based structures [3]—this study discusses multi-way decision diagrams.

Previous research has addressed the measurement of complexity in product configuration systems from different perspectives. Ghosh et al. [4] analyse the cognitive complexity of configurators using a UML-based metric, but their approach does not operate on the level of parts or Bills of Materials and does not compare visualisation techniques with objective complexity indicators. Similarly, Schmidt et al. [5] develop key performance indicators for managing variant diversity in complex product families, focusing on economic portfolio management rather than analysing objective complexity metrics or variant logic visualisation. Other approaches, such as Modrak and Bednar [6] and Herrera-Vidal et al. [7], apply entropy-based measures to quantify uncertainty and complexity but primarily address assembly or process-level dynamics rather than configuration rules at the Bill-of-Materials level. Additionally, Modrak and Soltysova [8] demonstrate that the mere number of possible variants is insufficient to capture true system complexity and propose information-theoretic measures to better reflect underlying dependencies.

Despite these contributions, no study has systematically connected quantitative complexity metrics with visual representations of configuration logic. This paper combines measures such as Shannon entropy and Huffman tree path lengths with the size of decision diagrams to investigate how configuration complexity affects the compactness of decision diagrams. To the best of our knowledge, no prior study has empirically investigated this relationship between data complexity and the visualisation of configuration logic.

1.1. Case Study Context and Data Source

The study is based on industrial data sourced from the product data management systems of a major automotive manufacturer, specifically focusing on part selection rules. The nature of this data reflects the challenges encountered in large-scale industrial configuration environments.

The data set, its scope and source, as well as the data processing, are described in Sect. 3.

2. Foundations

This section details the case study setup, i.e. the complexity indicators chosen to characterise the data, and the performance metrics used for the assessment of Poseidon.

2.1. Variation Points in the Bill of Materials

In product line engineering, a variation point is a bundle of alternative parts with corresponding selection criteria, often modelled simply as a list of formulas. The variation points analysed by the algorithm are documented in the so-called 150% bill of materials and are influenced by the product overview—also commonly referred to as the high-level configuration or feature model [1]. Each variation point in the bill of materials consists of at least one variant (i.e. physical part) and while there might also be several variants in one variation point, for each configured car, exactly one variant should be chosen out of each variation point. To ensure this, for every variant there is an item selection rule — based on the car configuration codes — and dedicated tools, that check, if documentation rules are satisfied. Detailed information on these tools is published in [9]. In Fig. 1 an example for a variation point containing three variants is visualised in POSEIDON.

2.2. Configuration rules

Configuration rules are documented in the product overview. There, properties (or features) of the cars — called codes — and consistency rules are defined. Following those rules, only certain combinations of codes are allowed to be configured. The resulting set of all possible car configurations is called *variation* *space*. In [1] we call this *context* and model it as a Boolean formula γ encoding the conjunction of all configuration constraints. In the example in Fig. 1 we assume that the context consists of only one rule, namely $\neg(A \wedge B \wedge C)$, leading to the $-C$ label generated on one edge.

PV	Code Rule
010	$\neg A \wedge \neg B$
020	$(\neg A \wedge B) \vee (A \wedge \neg B \wedge \neg C)$
030	$A \wedge ((\neg B \wedge C) \vee (B \wedge \neg C))$

Table 1
Example of part selection rules for a variation point containing three Variants.

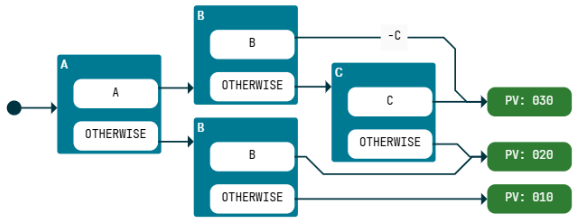


Figure 1: Variation point from Table 1 — visualised as decision diagram in Poseidon.

2.3. Algorithm Description

The core algorithm evaluated in this study transforms the input Boolean formulas into a multi-way decision diagram representation, as previously detailed in [1]. The key characteristics relevant to this evaluation are:

- Transformation: It systematically converts the propositional logic of the Boolean formulas into a directed acyclic graph (DAG).
- Multi-way Decision Nodes: Internal nodes represent decision points (corresponding to product feature alternatives). Each node can have multiple outgoing edges, representing the selection of specific alternatives for a given feature, plus a dedicated "otherwise" edge capturing the case where none of the explicitly listed options apply.
- Part Terminals: Terminal nodes (leaves) of the diagram directly represent the specific automotive part(s) selected when the conditions along the path from the root node are met.
- Application: The resulting diagram is the central data structure used in a visualiser/editor tool designed for configuration data engineers.
- Completeness: The diagram guarantees that every possible configuration path leads to a terminal node, ensuring that the representation is exhaustive and no valid variant is omitted.

2.4. Mathematical preliminaries

To make an evaluation of how efficient the graphical illustrations of POSEIDON are, we started with entropy, which is a well known concept in information theory when it comes to measuring the uncertainty of information [10]. We continued to calculate other complexity metrics; the three metrics used in this study are defined below.

For all calculations in this study consider a finite set

$$U = \{u_1, ..., u_n\} \quad \text{with} \quad |U| = n, \tag{1}$$

which in this study represents the set of all valid vehicle configurations under consideration. The assignment of vehicle configurations to a specific variant is modelled by a disjointed partition π of U, i.e. a set of sets (bundles). These bundles are technically component equivalence classes, as each bundle B_i contains all configurations that lead to the same installed part.

$$\pi = \{B_1, ..., B_m\} \quad \text{with} \quad |\pi| = m \tag{2}$$

such that the following properties hold:

$$\bigcup_{i=1}^{m} B_i = U \tag{3}$$

$$B_i \cap B_j = \emptyset, \quad \text{if } i \neq j \tag{4}$$

Visualisations of partitions can be found in Fig. 2 and Fig. 3.

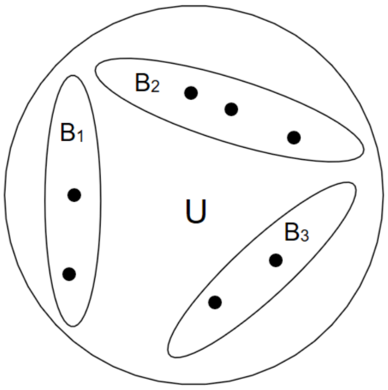


Figure 2: Visualisation of the set U associated to the example in Fig. 1 — containing seven Elements with a partition $\pi_1 = \{B_1, B_2, B_3\}$. The sets $B1$, B_2 and B_3 correspond to variants 010, 020 and 030 in Fig. 1 respectively.

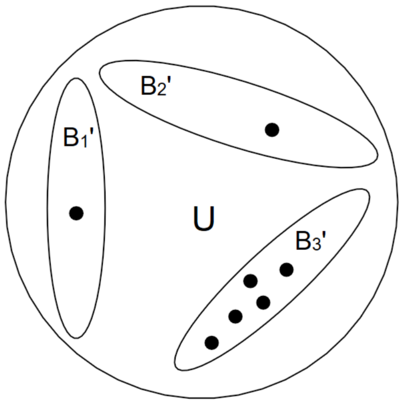


Figure 3: Visualisation of an alternative partition $\pi_2 = \{B'_1, B'_2, B'_3\}$ over the same universe U as in Fig. 2. The sets B'_1, B'_2 and B'_3 represent a different grouping of the same elements and are not to be confused with B_1, B_2, and B_3 in Fig. 2.

2.5. Complexity Indicators

2.5.1. Model Counting

An important factor for our calculations is the number of satisfying assignments (called *models*) for the Boolean formula, or relevant sub-formulas corresponding to alternative selection bundles. It reflects the total number of configurations which are valid as defined by the context. In our setting, this corresponds to the number of valid vehicle configurations represented by the set U. The variables (codes) used to describe the configurations are only those present in the item selection rules. Thus we project the context to them. This greatly reduces the effective model count by discarding all irrelevant dimensions for telling configurations apart.

In the context of our modelling, we denote:

$$|U| = \text{Number of valid configurations} \tag{5}$$

Remark 2.1. *Note that for the context of this work the variant space is projected to the variables occurring in a single variation point. For instance, in the example in Fig. 1, only the variables A, B and C are taken into consideration. Other variables might exist, but are ignored in the context of this variation point. Thus, in this example we have $|U| = 2^3 - 1$, as one possible combination of variables is excluded by context.*

Remark 2.2. *This metric provides a direct measure of the solution space size and indicates how many valid feature combinations exist. It serves as the base set for calculating derived measures such as the relative frequencies of variants in entropy-based evaluations.*

2.5.2. Shannon Entropy

We use Shannon entropy [10], a well-established metric from information theory, to quantitatively describe the complexity of a given variation point. It captures the uncertainty in the distribution of part variants across all valid vehicle configurations. A high entropy value indicates a balanced usage of many variants, while low entropy suggests the presence of dominant, frequently used options. This allows for direct comparisons between positions or even across different model series.

For some partition π, Shannon Entropy $H_S(\pi)$ is defined as:

$$H_S(\pi) = \sum_{B_i \in \pi} p_i \cdot \log_2\left(\frac{1}{p_i}\right) \quad \text{where } p_i = \frac{|B_i|}{|U|} \tag{6}$$

Definition 2.3. *Let π_1 and π_2 be two partitions of the same set U. We say that π_1 is* more complex in terms of variant distribution *than π_2 if*

$$H_S(\pi_1) > H_S(\pi_2),$$

i.e., if the Shannon entropy of π_1 is strictly greater than that of π_2.

Remark 2.4. *The Shannon entropy has the following property:*

$$0 \leq H_S(\pi) \leq \log_2(m) \tag{7}$$

Example 2.5. *To get an intuition for this complexity measure, take a look at the examples defined in Fig. 2 and Fig. 3:*

$$H_S(\pi_1) = \sum_{i=1}^{3} p_i \cdot log_2\left(\frac{1}{p_i}\right)$$

$$
\begin{aligned}
&= \frac{2}{7} \cdot log_2\left(\frac{7}{2}\right) + \frac{3}{7} \cdot log_2\left(\frac{7}{3}\right) + \frac{2}{7} \cdot log_2\left(\frac{7}{2}\right) \\
&\approx 1.557 \\
H_S(\pi_2) &= \sum_{i=1}^{3} p_i \cdot log_2\left(\frac{1}{p_i}\right) \\
&= \frac{1}{7} \cdot log_2\left(7\right) + \frac{1}{7} \cdot log_2\left(7\right) + \frac{5}{7} \cdot log_2\left(\frac{7}{5}\right) \\
&\approx 1.149
\end{aligned}
$$

We observe that π_1 exhibits a higher entropy than π_2. This results from a more uniform distribution of elements across the subsets in π_1, whereas in π_2 (cf. Fig. 3), the majority of elements are concentrated in B_3'.

2.5.3. Huffman Trees

In addition, we consider the average path length of a Huffman tree constructed using the relative frequencies of the part variants [11]. This length reflects the average number of binary decisions required to uniquely identify a variant. Thus, the Huffman tree simulates an optimal binary decision tree and serves as a theoretical benchmark for evaluating actual representations, such as those used in tools like POSEIDON. Constructing a Huffman tree can be done by implementing the following algorithm for some partition $\pi = \{B_1, ..., B_n\}$:

1. Create a working list of nodes for each B_i with value $|B_i|$.
2. Look for two nodes i and j with smallest value, i.e. $|B_k| \geq |B_i|$ and $|B_k| \geq |B_j|$ for all $k \neq i, j$.
3. Remove the nodes i and j from the list and add a node which is their (new) parent with the sum of the children's values as its value.
4. Repeat until one node is left, this is the root of the resulting Huffman tree.

Looking at this algorithm in terms of building a decision diagram, we combine the two least common endpoints in a sub tree and add up their relative frequencies. This is repeated, until the whole decision diagram is built. By evaluating this Huffman tree, we get a binary encoding — consisting of all yes-no-decisions that have to be taken to get to the regarding endpoint — for every subset B_i of the partition π. We denote the length of this binary code as l_i.

With this, we can analyse the average path length of the Huffman trees

$$\frac{1}{n}\sum_{i=1}^{n} l_i, \tag{8}$$

as well as the weighted average path lengths

$$\sum_{i=1}^{n} l_i \cdot p_i, \text{ where } p_i \text{ is the relative frequency at which a path is chosen.} \tag{9}$$

This indicators reflect how many binary decisions are needed on average to uniquely identify a variant, treating the different parts equally or taking into account their relative frequencies respectively.

2.6. Performance of POSEIDON

To evaluate the effectiveness of the algorithm and the comprehensibility of the resulting graph, several performance metrics are applied. These refer both to the structure of the generated decision diagram—particularly with regard to its suitability as a compact and interpretable representation—and to characteristics of the underlying binary structure used during its construction. In addition, the Pearson correlation coefficient is used to analyse relationships between selected complexity metrics.

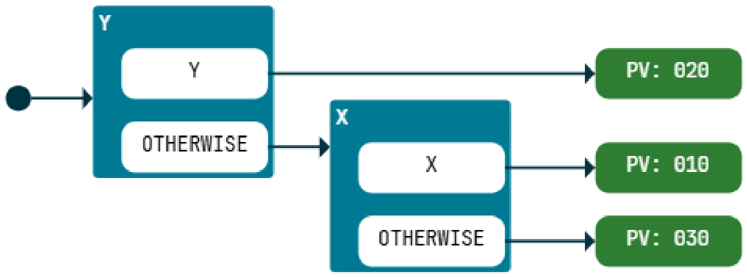


Figure 4: Huffman tree for the example mentioned in Table 1. X and Y are idealised configuration values that subdivide vehicle configuration as needed for the tree construction.

2.6.1. Average Path Length in Multi-valued Decision Diagrams (MDD)

The *average path length* in a multi-valued decision diagram (MDD) quantifies how many decision steps are required on average to reach a terminal node from the root node. A terminal node corresponds to a specific part variant.

Let n denote the number of valid paths in the MDD that lead to a concrete output. For each such path i (with $i = 1, \ldots, n$), let d_{m_i} be the number of decisions made along that path, where each decision may involve selecting one of multiple alternatives.

Then, the average path length L_{MDD} is computed as:

$$L_{\text{MDD}} = \frac{1}{n} \sum_{i=1}^{n} d_{m_i} \tag{10}$$

This metric reflects the compactness and interpretability of the diagram structure: shorter paths imply fewer decision steps to reach a result. Accordingly, a lower L_{MDD} indicates a more concise and potentially more comprehensible configuration logic.

2.6.2. Binary Average Path Length in Binary Decision Diagrams (BDD)

The *binary average path length* quantifies the expected number of binary decisions required to reach a terminal node from the root node in a binary decision diagram (BDD). In contrast to the average path length in an MDD—which may count multi-valued decisions as single steps—this metric assumes that all variation points are represented through binary decisions. That is, each possible alternative is encoded as a separate yes/no condition that must be evaluated along the path.

Let n be the number of valid paths in the BDD that lead to product variants. For each path i (with $i = 1, \ldots, n$), let d_{b_i} denote the number of binary decisions made along that path.

Then, the binary average path length L_{BDD} is computed as:

$$L_{\text{BDD}} = \frac{1}{n} \sum_{i=1}^{n} d_{b_i} \tag{11}$$

Remark 2.6. *Comparing the binary average path length to the average path length of the multi-way decision diagram generated by POSEIDON allows us to assess how efficiently the algorithm reduces decisions required.*

2.6.3. Pearson Correlation

To quantitatively assess the linear relationship between two complexity metrics, we apply the Pearson correlation coefficient r [12]. This coefficient indicates whether and to what extent two metric variables are linearly correlated. A positive value suggests a direct relationship, a negative value an inverse relationship, and a value close to 0 indicates no linear correlation.

For two variables x and y with N observations, r is defined as follows:

$$r = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{(N-1) \cdot s_x \cdot s_y} \tag{12}$$

Where:

- x_i, y_i: individual observations of the variables x and y
- $\bar{x}, \bar{y}$: arithmetic means of x and y
- s_x, s_y: empirical standard deviations of x and y

Remark 2.7. *The Pearson coefficient is a dimensionless value ranging from -1 to 1. Values close to $|1|$ indicate a strong linear relationship, while values near 0 suggest a weak or no linear correlation.*

Remark 2.8. *In this study, the coefficient is used to examine the extent to which structural metrics, such as average path length, align with information-based metrics. Each observation x_i and y_i represents the value of a specific complexity metric (e.g., entropy, path length, model count) measured at an individual BOM position. The results of these analyses are visualized in the correlation matrices shown in Fig. 5.*

While alternative complexity measures such as Kolmogorov complexity or spectral entropy were also considered, they were found to be impractical for this specific use case due to either their computational intractability or their limited interpretability in the context of real-world configuration logic.

3. Data Set — Engineering Bill of Materials

3.1. Data Source and Scope

This analysis is based on real-world data from the product configuration system of a major automotive manufacturer. The central data source is the so-called 150% Bill of Materials (BOM), a maximal BOM containing all potentially installable components and assemblies across all available product variants. In this work, we focus on the engineering stage, where the 150% BOM is represented as the Engineering Bill of Materials (E-BOM), which serves as the foundation for the systematic analysis of variant logic at the position level. Following Eigner et al. [13, p. 272], the E-BOM denotes the bill of materials during the engineering stage and is often referred to as a generic BOM [14].

In addition, we incorporate configuration rules from the context. These define which combinations of feature characteristics lead to valid — i.e., buildable — vehicle configurations. Based on this data, it is possible to determine for each variant how often each part is used, denoted as valid configurations. This distribution forms the basis for relative frequency calculations used in the complexity metrics.

The data under investigation pertains to specific model series. A model series describes a technical base model of a vehicle that is being produced in various variants—for example, with different engine types, equipment levels, or drivetrains. The selected model series have been produced since 2023 and 2024 respectively. As such, the underlying dataset is derived from real-world production data and is well-suited for in-depth analysis. Note, however, that real-world data is always evolving, e.g. our dataset might include currently unused future part variants etc. We address this further in Sect. 3.4.

For the subsequent analysis, we distinguish between two datasets, each representing a separate model series. These will be referred to as *Model Series A* and *Model Series B* throughout the remainder of this study.

3.2. Structure and Representation of Variant Logic

Each BOM position is linked to a set of code rules, expressed in the form of Boolean formulas, which define the conditions under which a particular component is installed. These rules form the central logical structure for variant management.

The internal tool POSEIDON is used for visualising these code rules. It transforms the Boolean formulas into decision diagrams represented as directed graphs. This form of representation enables a structured understanding of rule complexity and also supports the validation of rule sets with regard to uniqueness, completeness, and satisfiability [15].

3.3. Analytical Objective

The objective of this analysis is to quantitatively assess and evaluate the complexity of variant logic at selected BOM positions. For this purpose, established metrics such as Shannon entropy and the average path length in the corresponding Huffman tree are employed. These indicators were discussed in detail in Sect. 2.5 and enable a standardized assessment of variant diversity across different positions.

Based on those general measures of complexity, we aim to evaluate the performance of the POSEIDON algorithm by correlating the results obtained using POSEIDON with the underlying data complexity.

3.4. Data Cleaning

Analysis shows that some decision diagrams contain paths that end in leaf nodes without an associated component. These incomplete endpoints may result from various causes—for instance, a component may not be required for certain feature combinations, a part may not yet be maintained in the system, or inconsistencies may exist in the rule set.

To accurately capture the true structure of the variant logic, these paths are explicitly included in the calculation of the complexity metrics. In other words, whenever the data implies no part selection, we treat this as intentional and make that option explicit in the data before doing our calculations.

Furthermore, there are part selection rules that are logically contradictory or unsatisfiable within the context. These parts were completely ignored in the analysis, as they have a model count of zero and thus no influence on the output data. The item selection rule in such a case is semantically equivalent to false with regard to the valid vehicle configurations.

In addition, all outdated positions for which no feasible part remained were removed from the dataset.

This approach allows for a comprehensive assessment that considers not only the defined variants but also potential rule gaps or modelling issues.

4. Encoding and Visualisation of Variation Points

This section discusses encoding options for variation points such as truth tables, KV diagrams, Venn diagrams, and decision diagrams, while highlighting the advantages and disadvantages of these respective options, especially with regard to integrability of context information.

While various representations for Boolean configuration logic exist—such as tabular formats, Karnaugh maps, and Venn diagrams—this study focuses on multi-way decision diagrams for encoding variation points.

All these methods can be used to determine which of the variant(s) in a given variation point can be chosen for a given Boolean configuration. However, in the use case of the variation point visualisation, a considerable part of the theoretical variant space is not relevant a priori, because it is not buildable under consideration of the product overview. A key consideration for the applicability of an encoding is how effectively it conceals the non-feasible variant space.

In the following paragraphs, the methods for presenting Boolean logic, the necessary modifications for displaying mappings, and the relevant literature are discussed.

4.1. The Role of Visualisation in Data Quality and Related Work

Data visualisation can play a key role in data quality assurance by providing an intuitive and interactive way to represent and analyse data. By visualising the data, it becomes easier for data workers to identify patterns, trends, and anomalies in the data, which can help identify errors, inconsistencies, or other issues that can affect the quality of the data. Visualisation can also help to highlight the relationships and dependencies between different data elements, making it easier to understand the context in which the data is used and how it may be affected by changes.

4.2. Visualisation Option: Formulas

Encoding variation points using Boolean formulas is a common industrial practice [16]. For instance, the BOM uses Boolean formulas as item selection rules. Some companies require formulas to match certain normalizing criteria, whereas others allow any syntactically correct formula.

Formulas have the advantage of acceptable scaling, both in the case of many involved variables, as well as in the case of complex inner relations. Though in the worst-case-scenario a Boolean formula will double in size with the introduction of a new variable, in the average case it will grow significantly less. However Formulas have the disadvantage of being hard to evaluate mentally. In addition, the known normal forms for Boolean formulas are either not canonical, hard to comprehend for humans, or too large.

4.3. Visualisation Option: Truth- & Decision Tables

A common approach to visualise variability data are various forms of tables. The naive idea is to list all possible feature combinations, i.e. a truth table.

Each row of the table represents a possible combination of values for a set of Boolean inputs, and the corresponding column indicates whether the combination satisfies a given Boolean expression (output). Truth tables can be a useful tool for analysing and manipulating Boolean data, as they provide a clear representation of the possible combinations of values and the relationships between them.

However, truth tables can become unwieldy as the number of Boolean variables increases. As the number of variables of the expressions increases, the number of rows in the truth table grows exponentially, making it difficult to understand and manipulate the data.

To limit the size of the table, combinations that do not satisfy a validity constraint (in our case, the product overview) can be hidden.

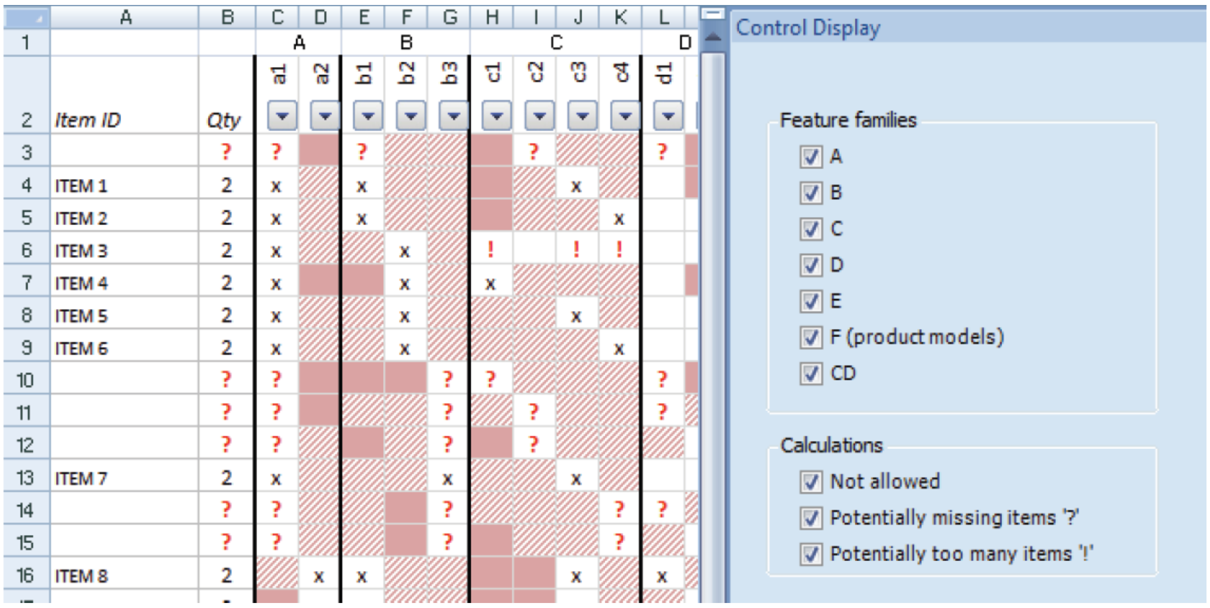
A similar table standard is used by Tidstam et al. [17] for assigning part variants to feature combinations in a configuration context. In Table 2, taken from [17], the red cells represent invalid feature selections.

One way of improving the scaling of the table is that after successful assignment, several vehicle properties/variables can often be bundled into mutually exclusive families instead of as a separate column, as in a truth value table, which enables the consolidation of many Boolean columns to multi-value columns. For instance, Table 3 doesn't show dedicated Boolean columns for *Gasoline* and *Diesel* as in the case of *Turbo, Sport* and *City*, but instead the multi-value column *Fuel* with the exclusive variants *gasoline* and *diesel*.

A further reduction is possible if only slightly different vehicle variants are assigned to the same component. For instance, the last two lines in Table 3, that only differ in their value in the city column, show the same component. For these kind of constellations, some table representations use *don't-care* symbols. In this case, a line *Volume=1.2, Turbo=no, Sport=no, City=*, Fuel=gasoline, Item(s)=E12* could have subsumed the last two lines and thus replace them.

However, this approach doesn't terminate quickly and yields its own optimisation issue, addressed by the Quine-McCluskey algorithm [18][19].

A table, that has been optimised that way, eventually contains only prime implicants for each item and in general there are multiple table rows required. Due to the potential overlap of the prime implicants, the use of all prime implicants is only necessary in the worst case.



	A	B	C	D	E	F	G	H	I	J	K	L
1			A		B			C				D
2	*Item ID*	*Qty*	a1	a2	b1	b2	b3	c1	c2	c3	c4	d1
3		?	?	🟥	?	▨	▨	🟥	?	▨	▨	?
4	ITEM 1	2	x	▨	x	▨	▨	🟥	▨	x	▨	
5	ITEM 2	2	x	▨	x	▨	▨	🟥	▨	▨	x	
6	ITEM 3	2	x	▨	▨	x	▨	!		!	!	
7	ITEM 4	2	x	🟥	🟥	x	▨	x	▨	▨	▨	
8	ITEM 5	2	x	▨	▨	x	▨	▨	▨	x	▨	
9	ITEM 6	2	x	▨	▨	x	▨	▨	▨	▨	x	
10		?	?	🟥	🟥	🟥	?	?	▨	▨	▨	?
11		?	?	🟥	▨	▨	?	▨	?	▨	▨	?
12		?	?	▨	🟥	▨	?	🟥	?	▨	▨	▨
13	ITEM 7	2	x	▨	▨	▨	x	▨	▨	x	▨	
14		?	?	▨	▨	🟥	?	▨	▨	▨	?	?
15		?	?	▨	▨	🟥	?	🟥	▨	▨	?	▨
16	ITEM 8	2	▨	x	x	▨	▨	🟥	🟥	x	▨	x

Control Display

Feature families
- ✓ A
- ✓ B
- ✓ C
- ✓ D
- ✓ E
- ✓ F (product models)
- ✓ CD

Calculations
- ✓ Not allowed
- ✓ Potentially missing items '?'
- ✓ Potentially too many items '!'

Table 2
Improved visualisaion of item selection rules as truth table, taken from Tidstam et al. [17].

Families and variants					
Volume	Turbo	Sport	City	Fuel	*Item(s)*
1.6	yes	yes	no	gasoline	*E16T*
1.6	no	no	no	diesel	*E16D*
1.2	no	no	yes	gasoline	*E12*
1.2	no	no	no	gasoline	*E12*

Table 3
Tabular assignment of vehicle variants to component variants as displayed in Voronov et al. [16].

In terms of human readability, the table has the considerable advantage that it is understood intuitively and that there are good tools for editing tables. In contrast, the scaling properties of tables are no longer sufficient for many practical data constellations.

For these reasons, truth tables may not be well-suited for visualising complex Boolean data for data workers. The scaling properties make them undesirable or outright impossible to apply in some real-world scenarios. In such cases, other visualisation methods, such as decision diagrams or graphical models, may be more effective for helping data workers to understand and analyse the data.

4.4. Visualisation Option: Decision Diagrams

Decision Diagrams have been selected as the visualisation method for POSEIDON. Our algorithm utilizes Binary Decision Diagrams (BDDs) to transform data into multiple decision diagrams (MDDs).

By leveraging BDDs, we obtain a data structure that meets numerous requirements. The exclusion of unbuildable spaces is straightforward, achieved by removing the corresponding subgraphs. Configurable variable ordering, along with ordering heuristics to minimize graph size, provides customizable perspectives on the data. Reduced decision diagrams enhance efficiency by reusing equivalent subtrees, resulting in more concise representations

In [20] the authors encode vehicle configuration at Renault into a Constraint Satisfaction Problem (CSP). Whereas Amilhastre et al. [21] encode context validity in a CSP, compile it into an automaton and represent the automaton graphically. The automaton accepts valid feature selections, but, in contrast to our approach, does not compute which part selection follows.

4.5. Other Visualization Options

Karnaugh Maps [22], also known as KV diagrams, are a graphical representation of Boolean expressions used to simplify and minimise the expressions. They are based on the idea of grouping adjacent terms in a truth table that represent the same logical function. KV diagrams appear to be unsuitable for the visualisation of variation points, since they cannot simply hide a cell because the relative positions of cells to each other are of vital importance, and their meaning is defined via their geometric position.

Similar issues can be observed for Venn Diagrams. They also become notoriously difficult to work with, even for a small number of variables.

5. Experimental Results and Evaluation

5.1. Quantifying BOM Complexity

In addition to the number of valid configurations (Model Count), the evaluation incorporates indicators of distributional characteristics of the solution space. These indicators include entropy-based measures and average path lengths derived from Huffman Trees, Multi-valued Decision Diagrams (MDDs), and Binary Decision Diagrams (BDDs).

For each metric, key descriptive statistics were calculated, including minimum, maximum, arithmetic mean, median, and standard deviation. The results are detailed in Table 4 and Table 5.

Table 4
Statistical Key Figures – Model Series A

Metric	Min	Max	Mean	Med.	Std. Dev.
Model Count	2.00	2520.00	18.34	5.00	74.66
Shannon Entropy	0.00	4.82	0.96	0.97	0.66
Huffman Weighted Avg. Path	1.00	5.90	2.03	2.00	0.67
Huffman Avg. Path	1.00	7.82	2.19	2.00	0.95
MDD Avg. Path	2.00	9.93	2.93	2.67	1.10
BDD Avg. Path	2.00	70.98	5.39	4.00	3.60

Table 5
Statistical Key Figures – Model Series B

Metric	Min	Max	Mean	Med.	Std. Dev.
Model Count	2.00	1760.00	17.39	6.00	49.64
Shannon Entropy	0.00	5.29	0.96	0.97	0.67
Huffman Weighted Avg. Path	1.00	6.34	2.04	2.00	0.68
Huffman Avg. Path	1.00	7.74	2.23	2.00	0.97
MDD Avg. Path	2.00	8.80	3.06	2.75	1.09
BDD Avg. Path	2.00	68.98	6.00	5.00	3.65

Model Count. The mean is 18.34 (A) and 17.39 (B), while the median is considerably lower at 5.00 and 6.00, respectively. This indicates a right-skewed distribution—most items have few valid variants, while a few show high combinatorial complexity, as also reflected by the high standard deviations (74.66 and 49.64).

Shannon Entropy. With a mean of 0.96 in both datasets, this suggests low combinatorial uncertainty—i.e., in most cases, only a few equally probable variants exist.

Huffman Tree Weighted Avg. Path length. Mean values of 2.03 (A) and 2.04 (B) indicate a more differentiated distribution, where rare but valid configurations contribute significantly to the total information content.

Huffman Tree Avg. Path length. Values of 2.19 (A) and 2.23 (B) suggest that, on average, about two binary decisions are required to uniquely identify a specific variant. Note: This is under the assumtion that such binary dimensions already exist and their interrelationship with other variables is modelled elsewhere. This is why the Huffman Tree represents a lower bound for real world BDDs.

MDD Avg. Path Length. The mean is 2.93 (A) and 3.06 (B). This metric reflects the average number of decision steps required to reach a terminal node in the multi-valued decision diagram generated by Poseidon and thus characterizes the structural depth of the rule logic.

BDD Avg. Path Length. The mean is 5.39 (A) and 6.00 (B). A purely binary encoding requires roughly twice as many steps as the MDD, highlighting the structural compactness of the multi-valued representation.

5.2. Evaluating MDD generation with regards to BOM Complexity

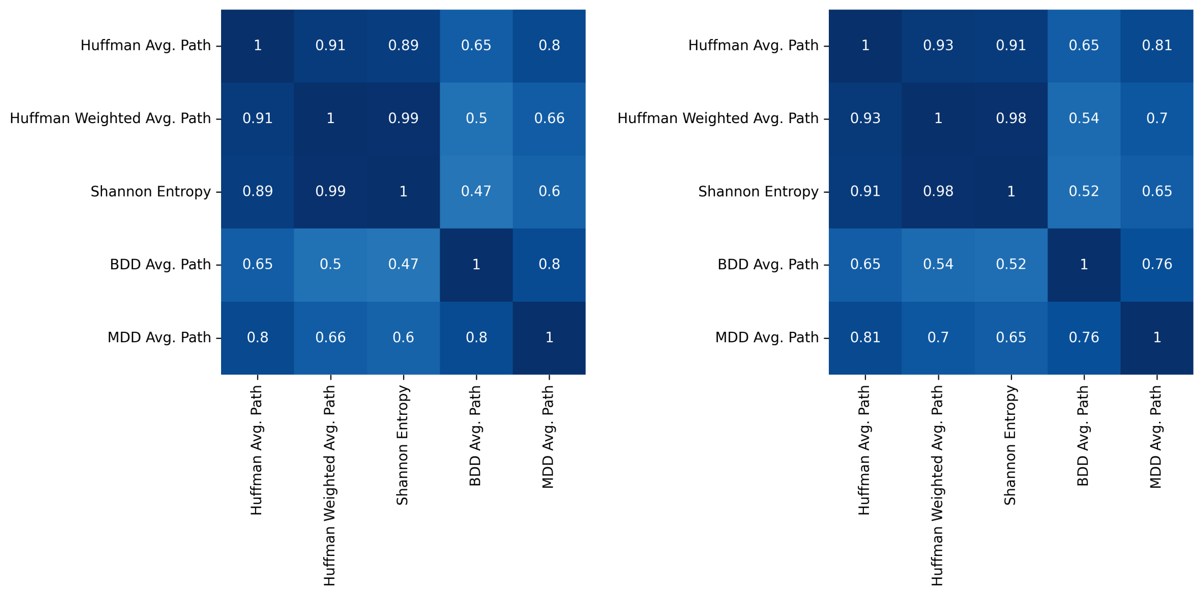


Figure 5: Comparison of the correlation matrices of the analysed metrics for variant complexity. Model Series A shown on the left, Model Series B on the right.

This section aims to investigate to what extent the decision diagrams generated by POSEIDON reflect the underlying variant structure. As reference measures, both information-theoretic and structural metrics are considered, including Shannon entropy and (weighted) Huffman Tree average path lengths.

5.2.1. Correlation Analysis of Complexity Metrics

To evaluate the relationships between the selected complexity metrics, the Pearson correlation coefficient r was calculated (cf.Sect. 2.5). The correlation matrices in Fig. 5 display the dependencies among entropy-based measures, the (weighted) average path length in the Huffman Tree, and the path lengths in decision diagrams based on MDDs and BDDs.

MDD Structure The average path lengths in the MDDs also show a positive correlation with entropy-based measures, albeit to a lesser extent. For Model Series A, correlation values range between $r = 0.65$ and 0.70, and for Model Series B, between $r = 0.60$ and 0.66. These correlations indicate that the structure of the decision diagrams reflects key aspects of the information-theoretic complexity of the configuration logic. This suggests a non-exponential scaling behaviour of the MDD algorithm with respect to increasing variant diversity — even though the underlying algorithm does not explicitly optimize for entropy.

Comparison with BDDs In contrast, the BDDs (Binary Decision Diagrams) exhibit significantly lower correlation with entropy-based metrics. In particular, the value for Shannon entropy in Model Series B drops to $r = 0.47$, indicating that BDDs are less sensitive to the actual distribution of configuration options. This reflects the binary nature of BDDs, where path lengths are primarily determined by the number of Boolean decisions rather than actual occurrence probabilities.

Model Comparison The comparison of the two model series reveals overall consistent correlation patterns. In Model Series A, the correlations are slightly stronger throughout, which may point to a more homogeneous variant structure or a lower degree of extreme configuration cases.

Additional observations Unsurprisingly, in both model series, a very high correlation is observed between Huffman Tree weighted Avg. Path length and Huffman Tree path lengths, as well as BDD and MDD path lengths. This is due to the fact that both Huffman Tree Path lengths are obtained from the same tree, while the MDDs are derived from the BDDs.

Conclusion Overall, the findings suggest that the structure of the MDD scales linearly with increasing variant complexity. This supports the conclusion that the logical complexity embedded in the configuration data is represented in a comprehensible and traceable manner.

5.2.2. MDD compared to Complexity Metrics

In Fig. 6, we investigate how structural and information-theoretic complexity metrics relate to the average path length in the MDD representation, based on all configuration positions.

Across all three metrics, a clear and approximately linear correlation is observed: Positions with greater entropy or more uneven variant distributions tend to produce deeper MDD structures. While the correlation strength differs slightly between metrics, the overall pattern indicates that the MDD depth increases in line with the underlying distributional complexity.

These results suggest that the MDD structure captures distributional complexity linearly, indicating that the MDD generation will continue to scale with increasing complexity.

5.2.3. Comparison betweeen MDD and BDD Average Path Lengths

In Fig. 7 we present the quotient between the Avg. Path Lengths of the BDD and the MDD for each variation point in Model Series A (top) and B (bottom) are visualised in relation to the Shannon Entropy. It appears that, independent of the underlying entropy, the MDD generation reduces the path length of the BDD by a factor of about 1.7.

5.3. Implications for Practical Applicability

The results presented in this chapter demonstrate that the algorithm used to transform configuration logic into decision diagrams performs robustly even as variant complexity increases. Despite increasing entropy and growing structural depth, the generated diagrams remain compact and interpretable.

This is particularly important for integration into the POSEIDON tool: since the generated structures are directly edited and reviewed by data engineers, a clear and concise representation is essential for

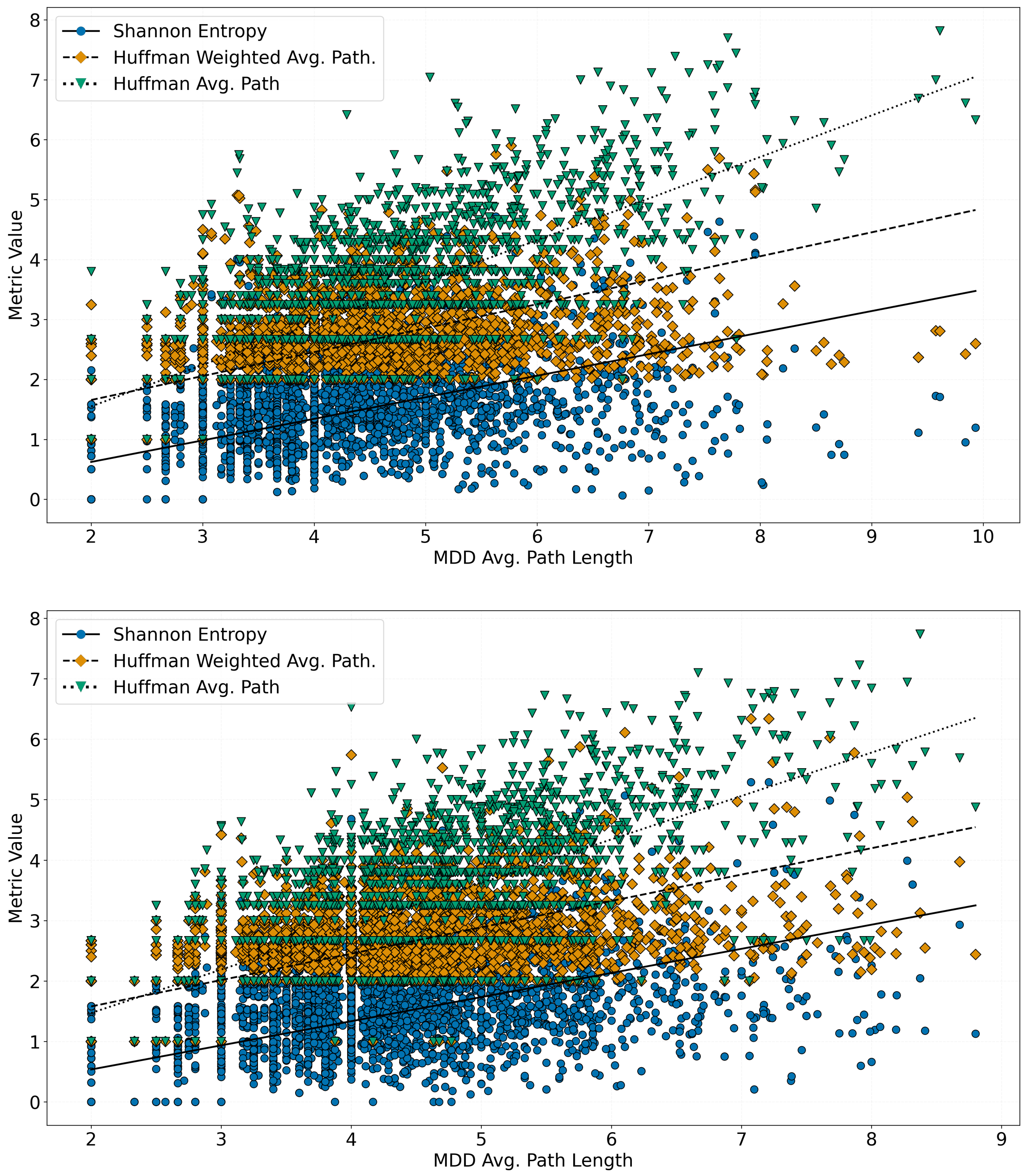
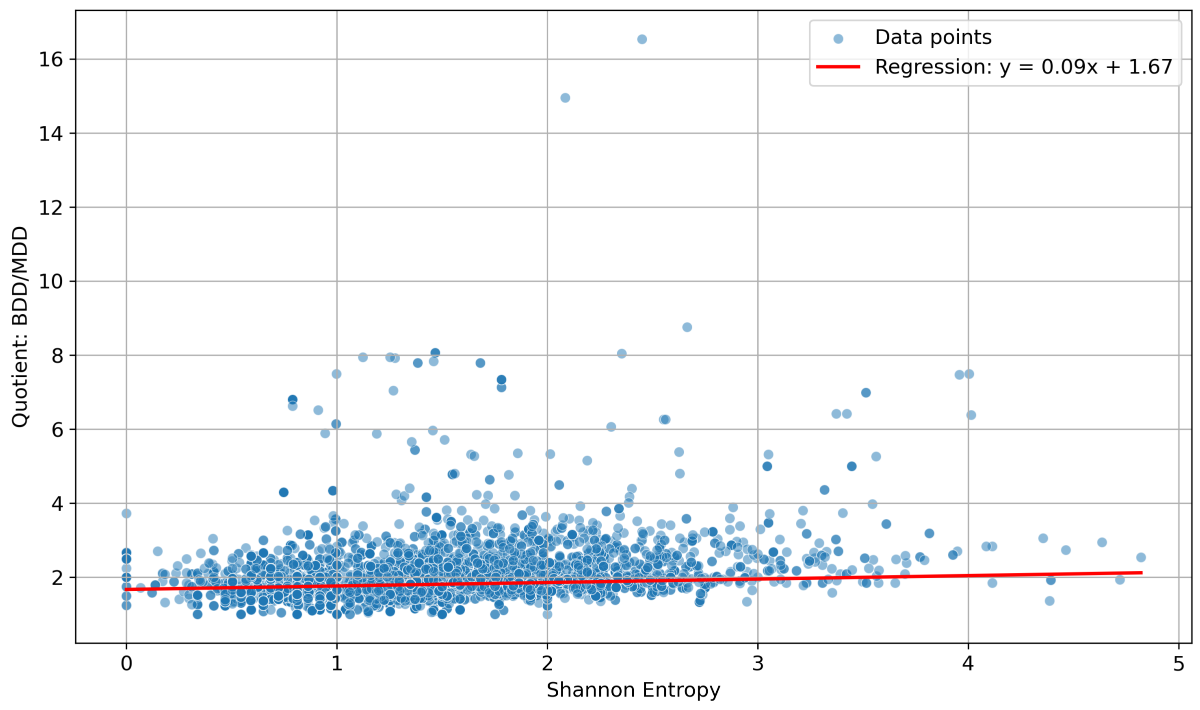


Figure 6: Comparison of the MDD Avg. Path Length and the complexity metrics per configuration position for Model Series A (top) and B (bottom). The average path length in the MDD is plotted against Shannon entropy (circle), Huffman Tree Weighted Avg. Path Length (diamond), and Huffman Tree Avg. Path Length (triangle), including corresponding trend lines.

maintainability, error prevention, and efficiency in handling complex product logic. The algorithm's ability to suppress non-configurable regions of the solution space while clearly structuring relevant branches offers a distinct advantage over alternative visualisation techniques.

Overall, the analysis shows that the measured complexity indicators are not merely theoretical constructs but have tangible implications for the usability and scalability of the tool in real-world application scenarios.



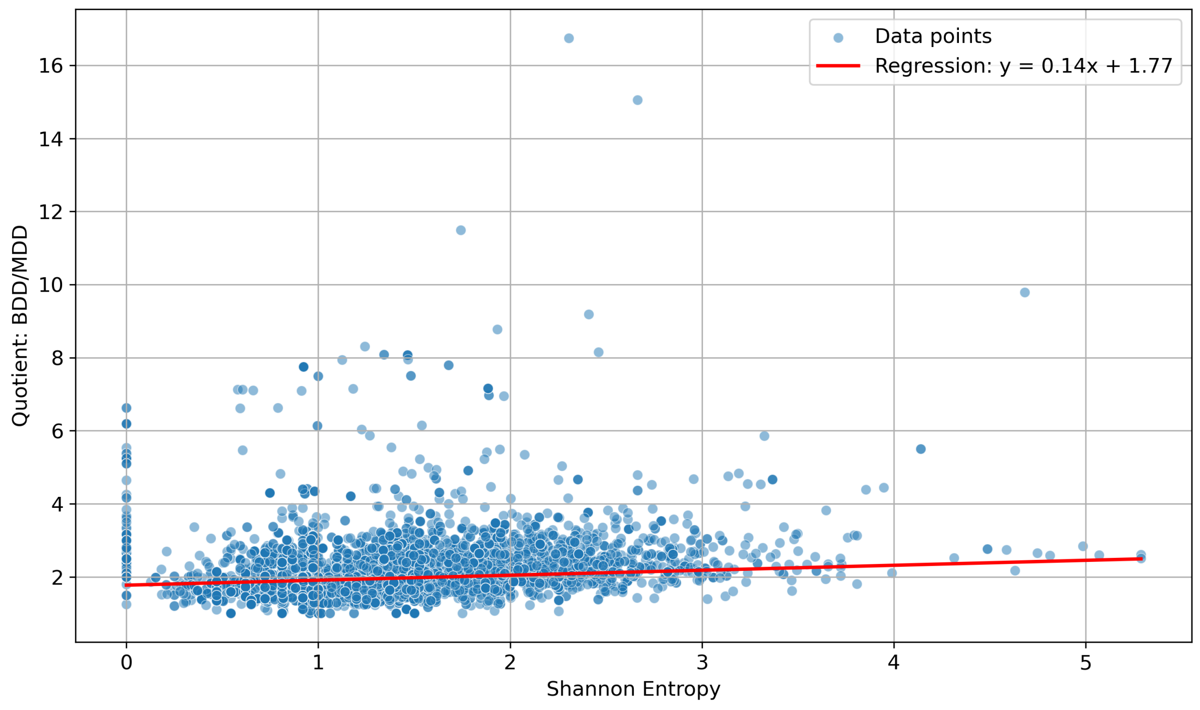


Figure 7: Relationship between Shannon Entropy and the quotient of average path lengths in BDDs and MDDs for Model Series A (top) and B (bottom). Each point represents a variation point.

6. Conclusion

This paper investigated the relationship between complexity indicators and the scalability of the POSEIDON algorithm for visualizing and manipulating automotive configuration data. This study demonstrated that the generation of POSEIDON's multi-way decision diagrams (MDDs) exhibit a linear scaling behavior with increasing variant complexity, as measured by Shannon entropy and Huffman tree path lengths. Compared to binary decision diagrams (BDDs), POSEIDON offers a distinct advantage by suppressing non-configurable regions of the solution space and providing a more compact and interpretable representation of the configuration logic, as demonstrated by the lower average path lengths observed in Tables 4 and 5.

The findings, derived from real-world configuration data, have significant implications for the practical applicability of POSEIDON. The linear scaling of MDD complexity ensures that the generated diagrams remain compact and interpretable even as variant diversity and logical interdependencies increase. This is crucial for data engineers who directly edit and review these structures, as it supports maintainability, error prevention, and efficient handling of complex product logic. The study confirms that the measured complexity indicators are not merely theoretical constructs but have tangible implications for the usability and scalability of the tool in real-world application scenarios.

This work contributes a comprehensive evaluation, combining theoretical indicators with empirical results, to demonstrate POSEIDON's effectiveness in managing the rising complexity observed in automotive product configuration.

While the evaluation focused on automotive data, future research should investigate POSEIDON's applicability to non-automotive domains and more heterogeneous data sources, for instance by analysing codebases and rule structures from other industries.

Additionally, further research could explore optimizing the MDD generation algorithm, especially for product lines with extremely high variant diversity. Another promising direction is to investigate alternative visualisation algorithms that maintain linear scalability while achieving a lower growth rate of the visualisation, enabling even more compact and efficient diagram representations.

Declaration on Generative AI

During the preparation of this work, the authors used an internal GenAI tool (based on GPT-4o) and Thesify for grammar, spelling checks, and literature proposals. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] D. Bischoff, W. Küchlin, O. Kopp, Poseidon: A graphical editor for item selection rules within feature combination rule contexts, in: F. Noël, F. Nyffenegger, L. Rivest, A. Bouras (Eds.), Product Lifecycle Management. PLM in Transition Times: The Place of Humans and Transformative Technologies, volume 667 of *IFIP Advances in Information and Communication Technology*, Springer Nature Switzerland, Cham, 2023, pp. 3–14. doi:10.1007/978-3-031-25182-5_1.

[2] Akers, Binary decision diagrams, IEEE Transactions on computers 100 (1978) 509–516.

[3] S. Shafiee, K. Kristjansdottir, L. Hvam, A. Felfernig, A. Myrodia, Analysis of visual representation techniques for product configuration systems in industrial companies, in: 2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), IEEE, 2016, pp. 793–797. doi:10.1109/ieem.2016.7797985.

[4] A. Ghosh, K. Kristjandottir, L. Hvam, E. Vareilles, Measuring the complexity of product configuration systems, in: Proceedings of the 20th International Configuration Workshop, volume 2220, CEUR Workshop Proceedings, 2018, pp. 61–68.

[5] M. Schmidt, J. Schwöbel, M. Lienkamp, Developing key performance indicators for variant management of complex product families, in: Proceedings of NordDesign 2018, Technical University of Munich, Linköping, Sweden, 2018, pp. 1–12.

[6] V. Modrak, S. Bednar, Entropy based versus combinatorial product configuration complexity in mass customized manufacturing, Procedia CIRP 41 (2016) 183–188. doi:10.1016/j.procir.2015.12.100.

[7] G. Herrera-Vidal, J. R. Coronado-Hernandez, I. Derpich, B. Paredes, G. Gatica, Measuring com-

plexity in manufacturing: Integrating entropic methods, programming and simulation, Entropy 27 (2025) 50. doi:10.3390/e27010050.

[8] V. Modrak, Z. Soltysova, Assessment of product variety complexity, Entropy 25 (2023) 119. doi:10.3390/e25010119.

[9] C. Sinz, A. Kaiser, W. Küchlin, Formal methods for the validation of automotive product configuration data, Ai Edam 17 (2003) 75–97.

[10] C. E. Shannon, A mathematical theory of communication, The Bell System Technical Journal 27 (1948) 379–423. doi:10.1002/j.1538-7305.1948.tb01338.x.

[11] D. A. Huffman, A method for the construction of minimum-redundancy codes, Proceedings of the IRE 40 (1952) 1098–1101.

[12] K. Pearson, Mathematical contributions to the theory of evolution. iii. regression, heredity, and panmixia, Philosophical Transactions of the Royal Society of London. Series A 187 (1896) 253–318. doi:10.1098/rsta.1896.0007.

[13] M. Eigner, D. Roubanov, R. Zafirov (Eds.), Modellbasierte virtuelle Produktentwicklung, 1 ed., Springer Vieweg Berlin, Heidelberg, 2014. doi:10.1007/978-3-662-43816-9.

[14] H. Hegge, J. Wortmann, Generic bill-of-material: a new product model, International Journal of Production Economics 23 (1991) 117–128. doi:https://doi.org/10.1016/0925-5273(91)90055-X.

[15] D. Bischoff, W. Küchlin, Adapting binary decision diagrams for visualizing product configuration data, in: INFORMATIK 2017, Gesellschaft für Informatik, Bonn, 2017, pp. 1499–1509. doi:10.18420/in2017_149.

[16] A. Voronov, A. Tidstam, K. Åkesson, J. Malmqvist, Verification of item usage rules in product configuration, in: IFIP International Conference on Product Lifecycle Management, Springer, 2012, pp. 182–191.

[17] A. Tidstam, L.-O. Bligård, F. Ekstedt, A. Voronov, K. Åkesson, J. Malmqvist, Development of industrial visualization tools for validation of vehicle configuration rules, in: Proceedings of the Tools and Methods of Competitive Engineering (TMCE), Organizing Committee of TMCE 2012, Karlsruhe, Germany, 2012, pp. 305–318.

[18] W. V. Quine, The problem of simplifying truth functions, The American mathematical monthly 59 (1952) 521–531.

[19] E. J. McCluskey, Minimization of boolean functions, The Bell System Technical Journal 35 (1956) 1417–1444.

[20] J. Astesana, L. Cosserat, H. Fargier, Constraint-based modeling and exploitation of a vehicle range at renault's: new requests for the csp formalism, in: International Conference on Tools with Artificial Intelligence (ICTAI), IEEE Computer Society, Arras, France, 2010, pp. 68–75. doi:10.1109/ICTAI.2010.19.

[21] J. Amilhastre, H. Fargier, P. Marquis, Consistency restoration and explanations in dynamic csps—application to configuration, Artificial Intelligence 135 (2002) 199–234.

[22] M. Karnaugh, The map method for synthesis of combinational logic circuits, Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics 72 (1953) 593–599.