Check for updates

Proceedings of the ASME 2025
International Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
IDETC-CIE2025
August 17-20, 2025, Anaheim, CA

DETC2025-165946

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89220/V03AT03A008/7557535/v03at03a008-detc2025-165946.pdf by guest on 12 February 2026

AN AUTOMATED FUNCTIONAL DECOMPOSITION METHOD BASED ON LARGE LANGUAGE MODELS

Zebin Cai[1,2], Ruocong Liu[2], Qianzhi Jing[1,2], Haoyu Zuo[2], Lingyun Sun[1], Peter Childs[2], Liuqing Chen[1,*]

[1]College of Computer Science and Technology, Zhejiang University, Hangzhou, China
[2]Dyson School of Design Engineering, Imperial College London, London, United Kingdom

ABSTRACT

Functional decomposition in conceptual design aims to break down the overall function of a complex product or system into smaller, implementable subfunctions. This process enables designers to systematically explore solution spaces, improve problem structuring, and manage design complexity. However, traditional functional decomposition presents substantial cognitive challenges, as designers often struggle to identify functionalities at suitable abstraction levels, adhere to structured syntax format requirements, and maintain solution neutrality. Such difficulties may result in inconsistencies in decomposition quality and inefficiencies throughout the design process. To mitigate these challenges, this research proposes an automated functional decomposition method leveraging a Large Language Model (LLM). A comprehensive function basis is constructed from a large-scale dataset of award-winning product and concept designs. This function basis serves as the vocabulary base for the automated decomposition process. The proposed method iteratively employs an LLM-powered functional decomposer to generate subfunctions while integrating a filtering mechanism to screen outputs and determine whether decomposition is complete. A comparative experiment evaluating designer-generated versus LLM-generated function trees demonstrates that the proposed method significantly enhances functional ideation breadth and improves decomposition quality and efficiency. The findings suggest that LLM-assisted decomposition can effectively support conceptual design, offering a scalable and systematic alternative to traditional manual approaches.

Keywords: Functional Decomposition, Large Language Model, Conceptual Design

1. INTRODUCTION

In the field of conceptual design, systematically understanding and defining a product's functionality is crucial for both innovation and problem-solving [1]. Functional decomposition is a widely used approach to achieve this, the method allows designers to break down complex design problems into simpler, manageable subfunctions, enabling systematic exploration of the design space [2]. Function trees are produced in the process of functional decomposition, offering a modular and hierarchical breakdown of functionality [3]. By structuring functions in this way, designers can explore more diverse solutions and reduce design fixation, fostering unbiased and innovative ideation [4].

Despite its advantages, functional decomposition remains a cognitively demanding task. Designers often struggle with identifying functions at an appropriate level of abstraction, as they naturally think in terms of physical components rather than abstract functional concepts [5]. Moreover, maintaining structured syntax rules (e.g., Verb-Noun format) and ensuring solution neutrality present further challenges, particularly for novice designers [3] (as will be discussed in Section 2.1). While the existing function basis [6, 7] provides a structured vocabulary to standardize function descriptions, it has practical limitations. Its predefined function sets are often rigid and abstract, making it difficult to capture the functional intent of innovative products. Designers may struggle to align real-world product functionalities with the constrained vocabulary, limiting the adaptability of function basis to complex design tasks. Automated functional decomposition methods also face challenges. Some existing approaches are tailored to specific functional categories, such as morphological changes of material flow [8]. This specialization reduces their generalizability across diverse product domains. This is because function identification often relies on expert knowledge, which is inherently subjective and inconsistent, making it difficult to formalize a universally applicable automated approach. These limitations highlight the need for a scalable and knowledge-driven functional decomposition method.

This study aims to address these challenges by introducing a Large Language Model (LLM)-based approach to automate functional decomposition. Traditional methods rely heavily on expert

*Corresponding author: chenlq@zju.edu.cn

Copyright © 2025 by ASME; reuse license CC-BY 4.0

knowledge. In contrast, the approach presented here incorporates a comprehensive function basis derived from a large-scale dataset of award-winning product designs. This serves as a structured knowledge base for the LLM. The decomposing process operates as an iterative loop, consisting of a functional decomposer and a function filter. Guided by design requirements and the constructed function basis, an LLM-powered decomposer systematically breaks down a parent function into multiple subfunctions. A filtering mechanism is then applied to assess the decomposed subfunctions and determine whether further decomposition is required, ensuring completeness and structural coherence.

The contributions of this research are as follows:

- A new function basis is constructed from award-winning product and concept design documents (e.g., Red Dot Design Award), making it more suitable for universal and innovative product development.
- An LLM-based decomposing method is proposed that systematically decomposes a main function into diverse subfunctions based on design requirements, broadening the functional ideation space.
- Structured top-down function trees are automatically generated to ensure compliance with syntax formats and solution neutrality principles. This significantly improves the efficiency and quality of functional decomposition.

2. RELATED WORK

2.1. Functional decomposition in design

Functional decomposition is a systematic design approach to assist designers in identifying essential functions across various design tasks, including concept generation, product dissection, and reverse engineering [5]. More generally, it serves as a problem-solving strategy that breaks down complex problems or customer requirements into simpler functions and subfunctions [1]. This enables designers to ideate solutions for each subfunction individually, rather than addressing the entire complexity at once. Functional decomposition methods are generally categorized into three types: top-down, energy-flow, and enumeration [5]. Top-down starts with identifying a system's overall function and progressively decomposes it into subfunctions until they reach an implementable level. Energy-flow maps the transformations of material, energy, and information flows within a product or system, linking functions to these changes. Enumeration is defined as generating functions through functional brainstorming, listing all possible functions without a structured approach or order. The output of functional decomposition can be visualized through various representations, including hierarchical function trees, flow diagrams, networks, and lists. Among these, function trees are the most commonly used. In a function tree, each function node represents an operation performed by an artifact, typically expressed in verb-object format (e.g., 'Control Torque') [2]. The verb describes the action performed, while the noun represents the flows (material, energy, or information) being acted upon. To enhance design flexibility and mitigate fixation, solution neutrality is a key requirement, meaning functions should be described independently of specific implementations. This approach helps reduce design fixation compared to other idea representation methods [4].

Prior studies have developed various methodologies and ontologies, such as FBS (Function-Behavior-Structure) and SAPPhIRE [9], to support functional decomposition. Despite its advantages, functional decomposition presents several challenges. Currently, functional decomposition is predominantly performed manually. A major difficulty lies in identifying functions, which is one of the most cognitively demanding tasks in the design process [5]. Studies have shown that novice designers, in particular, struggle to define functions independently of specific components and often find it difficult to conceptualize functions at an abstract level [10]. This challenge is further compounded by the requirement for solution neutrality, which is often perceived as counterintuitive by engineers who naturally think in terms of concrete solutions. Strict adherence to verb-noun syntax, though recommended to maintain function neutrality, has also been found to cause frustration, as designers may struggle to distinguish functions from physical components [2].

To support function identification and standardization, researchers have introduced several methodologies. The functional basis, proposed by Stone and Wood [6], provides a structured design language using a verb-object format, which was later refined by Bohm et al. [7] to integrate vocabulary from academia and industry. However, empirical studies indicate that designers rarely adhere strictly to the provided verb-object list, as the vocabulary remains too limited and abstract to fully capture the functional needs of products. More recently, automated approaches using graph-based functional representation models have been explored to support functional decomposition in an enumeration-based manner [11]. Although these methods show promise, they are typically designed for specific tasks, such as morphological changes of flows, and lack the generalizability needed for universal product design applications. Given these limitations, there is a strong need for a generalizable, automated functional decomposition method that can effectively support function identification while maintaining format requirements. Such an approach would enhance function modeling, reduce cognitive barriers, and improve applicability across diverse design scenarios.

2.2. AI for functional design

As Artificial Intelligence (AI) achieves remarkable progress in natural language processing and generation, researchers have introduced AI techniques to develop automated functional design methods. These methods focus on tasks such as function extraction, function modeling, function recombination, and function reasoning to assist designers in rapidly retrieving function information from extensive knowledge repositories, including patents, scientific publications, and encyclopedias, to stimulate design inspiration. Function extraction [12, 13] employs natural language processing techniques such as syntactic parsing and TF-IDF to identify function keywords from raw documents. Then these keywords can be structured into functional knowledge graphs using methods such as co-occurrence frequency analysis and dependency parsing to model statistical relationships between functions and their associated structures [14, 15]. Lin et al. [16] leverage language models like Word2Vec to construct functional vector

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89220/V03AT03A008/7557535/v03at03a008-detc2025-165946.pdf by guest on 12 February 2026

 Copyright © 2025 by ASME; reuse license CC-BY 4.0

spaces, where semantic similarity is represented by computing vector distances. This enables function recombination tasks such as function expansion, function trimming, and function substitution. Liu et al. [17] utilize semi-supervised classification algorithms to categorize functional keywords based on functional basis. Once these structured functional knowledge bases are constructed, they can be leveraged for function-based reasoning in subsequent design tasks.

Function-based reasoning aims to automatically map functional requirements to corresponding product behaviors and structure solutions, providing designers with valuable references and inspiration. Existing research has primarily focused on training machine learning models that take the given functions as input and generate relevant product behaviors and structures as output. On the one hand, Lou et al. [18] introduce a memory-inspired reinforcement learning framework to learn knowledge from historical design cases and guide the designer in selecting feasible behaviors for the specific function requirement. On the other hand, recent studies [19, 20] have introduced LLMs to follow the FBS reasoning framework and generate textual structure solutions based on the functional requirements, significantly enhancing both design efficiency and solution quality.

In summary, current AI-driven functional design research primarily focuses on function extraction, representation, and reasoning, yet overlooks functional decomposition. This is likely because functional decomposition not only requires AI to possess advanced cognitive reasoning capabilities but also demands strict adherence to formal constraints (e.g., syntax format and solution neutrality), making it inherently challenging to generate well-structured decomposition results. To address this gap, this study establishes a standardized function basis as a database and, for the first time, introduces LLMs into functional decomposition, addressing this critical limitation.

3. METHOD

3.1. Functional Decomposition Strategy

Fig.1 shows an overview of the proposed LLM-based functional decomposition approach, which decomposes functions iteratively. It mainly comprises four modules: function basis, functional decomposer, function filter, and the constructed function tree. To address the limitations of the existing function basis, we extracted functional vocabularies from recent award-winning design documents to construct a comprehensive new function basis. Details are elaborated in Section 3.2. The functional decomposer decomposes a parent function into multiple subfunctions according to design requirements and the function basis (detailed in Section 3.3). The function filter then evaluates these subfunctions, deciding whether further decomposition is needed (detailed in Section 3.4). We adopted the top-down functional decomposition method because its hierarchical structure facilitates step-by-step decomposition from abstract to specific functions. Compared to energy-flow or enumeration methods, this approach simplifies decomposition management and reduces complexity.

As shown in Fig.1, the decomposition begins by inputting the function basis, design requirements, and the main function into the functional decomposer, generating initial subfunctions ("F_1, F_2 ...F_n"). After decomposing all parent functions at each layer, the function filter evaluates these subfunctions and discards those failing to meet criteria (e.g., "F_3" in Fig.1). Approved subfunctions are added to the next layer of the function tree and are recursively input into the decomposer for further decomposition. When no new subfunctions pass the filter (N_{pass}=0), decomposition terminates, and the complete function tree is outputted.

3.2. Function Basis

To construct a more practical function basis, we extract functional phrases from product design descriptions. Award-winning design, patents, and product manuals are potential data sources that contain functional information. Patents are too domain-specific and highly technical, which does not align with conceptual design. Product manuals, on the other hand, focus on specifications and components rather than explicit functional descriptions. Instead, award-winning design descriptions provide clearer functional insights. Among them, Red Dot award[1] descriptions are particularly detailed compared to alternatives (such as the iF Award[2]), often explaining how product functions meet user needs. Therefore, we selected Red Dot data as our primary source.

The construction process is illustrated in Fig.2. First, we use a web crawler to extract "Design Description" text from Red Dot award-winning product pages. Then, an LLM extracts functional phrases based on a structured prompt, leveraging its natural language understanding capabilities. Prior research has demonstrated the effectiveness of LLMs in extracting function-related information [21]. The third step ensures that extracted function phrases conform to syntax rules and maintain solution neutrality. We reference prior function basis studies, which categorize verbs (operations) into eight types and nouns (flows) into three categories, as summarized in Table 1. To validate extracted function phrases, we first check whether they follow a verb-noun structure. If it does, we then verify that the verb belongs to one of the operation categories and the noun to a flow category. Only phrases that meet all criteria are retained. The final step eliminates redundant synonyms. Function phrases with high semantic similarity (e.g., "reduce energy consumption" vs. "reduce energy usage") are merged using SBERT-based embeddings [22]. To efficiently process large-scale data, FAISS (Facebook AI Similarity Search) [23] is employed for fast similarity-based clustering. Phrases with a similarity score above 0.9 are grouped, and one representative phrase from each cluster is retained in the final function basis.

3.3. Automated Functional Decomposer

The Functional Decomposer, powered by an LLM, is designed for decomposing a single parent function into its immediate subfunctions. We use GPT-4o, as it is one of the most capable LLMs available, demonstrating strong natural language reasoning abilities. Additionally, its large token capacity allows it to process the entire function basis in a single inference, ensuring consistency across decomposition results. To enforce a top-down hierarchical decomposition, we employ prompt engineering to guide the LLM effectively. The decomposition prompt

[1] https://www.red-dot.org/
[2] https://ifdesign.com/en/

Copyright © 2025 by ASME; reuse license CC-BY 4.0

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89220/V03AT03A008/7557535/v03at03a008-detc2025-165946.pdf by guest on 12 February 2026

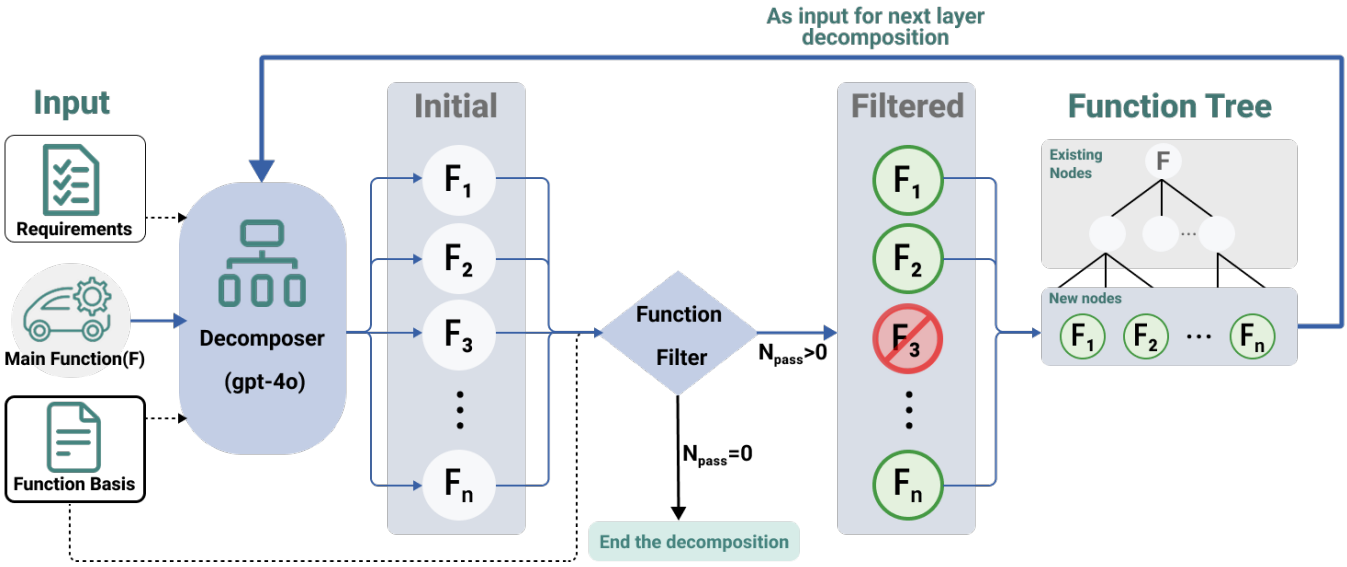


FIGURE 1: FUNCTIONAL DECOMPOSITION PROCESS

TABLE 1: CATEGORIZATION OF FUNCTION BASIS: OPERATION AND FLOW TYPES

Type	Class	B asic
Operation	Branch	Separate, Refine, Distribute
	Channel	Import, Export, transfer, Guide
	Connect	Couple, Mix
	Control Magnitude	Actuat, Regulate, Change
	Convert	Convert
	Provision	Store, Supply, Extract
	Signal	Sense, Indicate, Display, Measure
	Support	Stop, Stabilize, Secure, Position
Flow	Material	Human, Gas, Liquid, Solid
	Informatin	Status, Control
	Energy	Human, Acoustic, Biological, Chemical, Electrical, Electromagnetic, Hydraulic, Magnetic, Mechanical, Pneumatic, Radioactive, Thermal

is structured as shown in Table 2. The system_prompt defines the decomposition task, while the user_prompt provides input data, including the parent function, design requirements, and function basis.

The prompt design enforces structured decomposition while maintaining consistency and logical constraints. First, all generated subfunctions must adhere to the provided function basis to ensure standardization. The decomposition process also considers requirement awareness, aligning subfunctions with the provided design constraints to prevent arbitrary generation. To maintain the top-down structure, only direct subfunctions are generated, preventing unnecessary hierarchical depth. Additionally, logical independence ensures subfunctions are mutually exclusive and collectively exhaustive, reducing redundancy. Flow preservation guarantees that the material, energy, and information flows remain consistent across decomposition levels. To avoid over-decomposition, we limit the number of subfunctions to five, with the option to return an empty list when no meaningful decomposition is possible. Finally, the output strictly adheres to a JSON format, enabling seamless integration into downstream functional modeling.

3.4. Filtering Mechanism in Functional Decomposition

After the functional decomposer generates subfunctions for each node at a given layer, the function filter evaluates and refines them before proceeding to the next layer. The process removes empty functions and ensures all generated subfunctions exist within the constructed function basis. It also checks for redundancy by eliminating functions that have already appeared in previous layers. Additionally, if identical subfunctions are produced from different parent nodes, their semantic similarity (S-BERT) with their respective parents is computed. Among these, only the one (subfunction) with the highest similarity is retained, as we assume that a higher similarity indicates a stronger functional relationship.

4. EXPERIMENT AND EVALUATION

4.1. Construction of Function basis

We collected design description data from the "Industrial Design & Robotics" category under "Product Design" on the Red Dot Award website (Step 1 in Fig.2). This category was chosen because its products typically feature detailed functional descriptions and are commonly used in functional decomposition research [24]. In total, 1,645 product descriptions were obtained. The data was then processed using GPT-4o in two stages. First, functional phrase extraction was performed, generating 12,282 candidate phrases. Second, filtering was applied based on syntax rules (Verb-Noun structure) and solution neutrality, reducing the set to 6,142 valid functional phrases. Finally, the FAISS clustering algorithm was employed to merge synonymous phrases, resulting in 1,730 unique functional phrases.

Copyright © 2025 by ASME; reuse license CC-BY 4.0

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89220/V03AT03A008/7557535/v03at03a008-detc2025-165946.pdf by guest on 12 February 2026

TABLE 2: STRUCTURED PROMPT FOR FUNCTIONAL DECOMPOSITION

Type	Prompt
system_prompt	### Task Description You are an AI expert in conceptual design and functional decomposition. Your task is to decompose a given parent function into its immediate subfunctions, ensuring they align with the design requirements—but **only if** further decomposition is truly needed. ### Input Parameters - Parent Function: The function that needs decomposition. - Design Requirement: A natural language description of the design requirements. - Function Basis: A predefined list of allowed functions. ### Rules and Constraints 1. **Follow the Function Basis vocabulary**: Use only subfunctions that exist in the provided function basis list. 2. **Requirement Awareness**: Ensure that each generated subfunction aligns with the Requirement Description. If the requirement suggests a specific action or flow, reflect it in the subfunctions. 3. **Single-layer decomposition**: Each generated subfunction must be a **direct** child of the Parent Function. **Do not generate grandchildren** or deeper-layer functions. 4. **Logical Independence**: Subfunctions must be mutually exclusive and collectively exhaustive. A subfunction should not already contain another subfunction as part of its meaning. 5. **Flow Preservation (Energy-Material-Information Flow)**: Subfunctions must collectively achieve the Parent Function without exceeding its scope. Ensure the flow types (material, energy, signal) of the subfunctions are **covered** by the Parent Function's flow. 6. **Concise and Structured Selection**: Generate only the necessary number of subfunctions. **Not more than 5**. If you judge the Parent Function to be already atomic (i.e., cannot be meaningfully broken down further **within the design requirements**), then you may return **an empty JSON list** '[]' or **only one subfunction** if a partial decomposition is still meaningful. 7. **Output Format**: Return only a JSON list containing subfunction names. Do not include any additional text, code fences, or formatting outside the JSON list.
user_prompt	Parent Function: {parent_function}, Design Requirement: {requirement}, Function Basis:{function_basis}

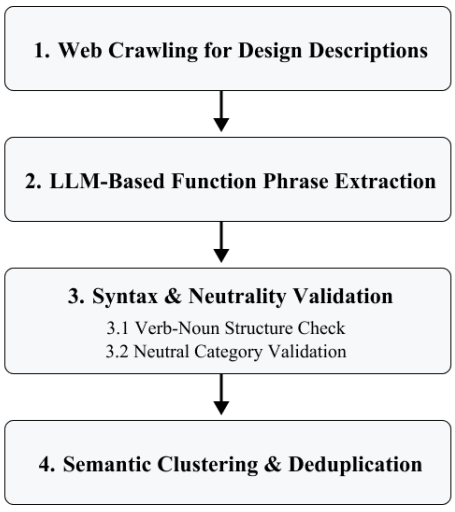


FIGURE 2: WORKFLOW FOR CONSTRUCTING THE FUNCTION BASIS

Compared to existing Function Bases, this approach significantly expands functional vocabulary coverage. For example, Stone and Wood [6] proposed a function basis with only 32 primary functions, while Bohm et al. [7] defined 21 secondary functions based on the former. Additionally, since our data is sourced from innovative award-winning products, the extracted phases may be more intuitive and better suited for modern product design.

4.2. Comparative experiment

We define the challenge of building a function tree as strict adherence to formal constraints and broadening the functional ideation space, so in this experiment, we obtained two groups to evaluate the performance of the LLM-based functional decomposition method we proposed against the human-generated function tree. The human group included 15 participants, between the ages of 22 to 31 (Mean=25.87, SD =2.88), including 10 females and 5 males, while all had engineering design-related work or study experience to ensure that the participants could complete the function decomposition tasks individually. Eight out of fifteen participants have experience in product decomposition tasks, while seven of them have experience in function decomposition tasks. Two decomposition tasks were provided to the partici-

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89220/V03AT03A008/7557535/v03at03a008-detc2025-165946.pdf by guest on 12 February 2026

Copyright © 2025 by ASME; reuse license CC-BY 4.0

pants to avoid bias and affect the objectivity of findings caused by user perception of a single product. A power screwdriver and a capsule coffee machine were selected as case products because these two products maintain the balance of familiarity and complexity for participants. Both of the products provide design space for participants, neither of them is too complex or too easy to decompose, and they could be finished within a limited time by participants. In the meantime, these two products have different operations, structures, and usage environments, which helps avoid bias caused by the need for participants with different domain knowledge.

Before the decomposition tasks, each participant will be provided with an experiment introduction: A page of definitions of function trees and functions, including an example of a fully decomposed top-down function tree and a list of some common functions as references from the prior function basis [6]; a specific introduction of decomposition principles with examples, including the definition of the top-down function tree structure, syntax format, and solution neutrality. During the tasks, participants will first review the information provided, including the main functions of the decomposition tasks, design requirements, and a brief scenario to help participants understand the basic function of these two products. After that, they were asked to construct a function tree, the specific guidelines were provided: (1) To decompose the main function as comprehensively as possible while ensuring that the function meets the design requirements; (2) ensure following the correct syntax format of functions writing on the function tree; (3) All functions should be considered solution neutral. Participants were asked to finish each task in around 30 minutes. After completing the decomposition tasks, all participants were asked to complete a post-study survey to gain a comprehensive understanding of the benefits and challenges encountered during functional decomposition and to identify opportunities for future research.

For the machine group, 30 results were generated just like the human group, which mitigated bias caused by objective differences in outlier outputs. The machine group employed the automated LLM-based functional decomposition method we proposed and carried out the same decomposition tasks with identical main functions and design requirements.

4.3. Evaluation Metrics

In the previous research, many metrics were used to evaluate the function tree, including identifying functions, evaluating tree efficiency [5], following the correct syntax format [5], and containing solution neutrality [2], and design space exploration [1, 2, 5]of the function tree. An example of the function tree is shown as Fig.3. For a node F_{ij} in a function tree, i denotes the layer of the function tree (starting from 0), and j indicates the node index within each layer (starting from 1). The definition and calculation method of the following metrics were identified:

- The basic metric *Vertices*, representing the total number of functions in a function tree, reflects the performance of function identification and supports the calculation of various metrics in this study. The formula for *Vertices* is shown below in Eq. (1)

$$\text{Vertices} = \sum_{i=0}^{m-1} \sum_{j=1}^{n} F_{ij} \tag{1}$$

- For measuring Tree efficiency

  Unique Functions: Total number of semantically non-repeated functions, which refer to functions without synonyms, so semantic similarity is measured using the S-BERT metrics with a threshold of 0.9. The function was rated as 1 if unique and 0 otherwise, represented by $U(F_{ij})$, where $U(F_{ij})$= 1 if the function $U(F_{ij})$ is unique and $U(F_{ij})$ = 0 if otherwise. The formula for *Unique functions* is shown below in Eq. (2)

$$\text{Unique Functions} = \sum_{i=0}^{m-1} \sum_{j=1}^{n} U(F_{ij}) \tag{2}$$

  Tree efficiency: The ratio of unique functions refers to the total number of functions in a function tree. Higher tree efficiency (above 70%) indicates less redundancy and more unique and better function identification, while lower tree efficiency(less than 70%) indicates more redundancy, less unique, and less clear function identification. The formula for *Tree efficiency* is shown below in Eq. (3)

$$\text{Tree Efficiency} = \frac{\text{Unique Functions}}{\text{Vertices}} \times 100\% \tag{3}$$

- For measuring Syntax Correctness:

  Syntax errors: the number of functions in a function tree that incorrectly use the syntax format, meaning functions that do not follow the "verb + noun" structure. A syntax error rating of 1 indicates the function is written in the expected format, and 0 otherwise. The calculation represented by the formula below as $S(F_{ij})$, where $S(F_{ij})$= 1 if the function $S(F_{ij})$ is correctly formatted, and $S(F_{ij})$ = 0 if the function $S(F_{ij})$ is incorrectly formatted. The formula for *Syntax error* is shown below in Eq. (4)

$$\text{Syntax Error} = \sum_{i=0}^{m-1} \sum_{j=1}^{n} (1 - S(F_{ij})) \tag{4}$$

  % Syntax error: The ratio of syntax error to the total number of functions is recorded in the function tree.

- For measure Solution neutrality

  SolutionNeutral: Number of functions express solution-neutral writing on a function tree, without specifying how the product implements them. A rating of 1 indicates a function written as solution-neutral and 0 otherwise. The rating is represented in the formula below as $N(F_{ij})$, where $N(F_{ij})$= 1 if the function $N(F_{ij})$ is solution-neutral, and $N(F_{ij})$ = 0 if otherwise. The formula for *SolutionNeutral* is shown below in Eq. (5)

Copyright © 2025 by ASME; reuse license CC-BY 4.0

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89220/V03AT03A008/7557535/v03at03a008-detc2025-165946.pdf by guest on 12 February 2026

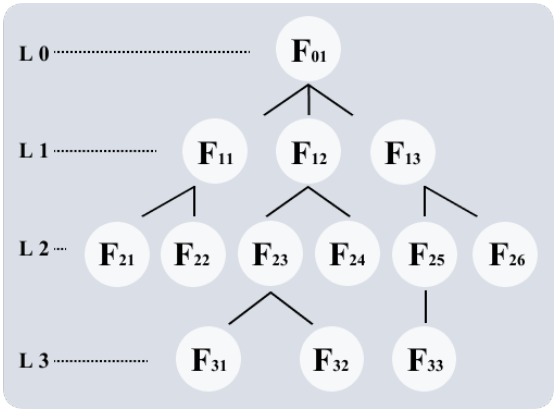


FIGURE 3: FUNCTION TREE EXAMPLE

$$SolutionNeutral = \sum_{i=0}^{m-1} \sum_{j=1}^{n} N(F_{ij}) \quad (5)$$

%SolutionNeutral: The proportion of solution-neutral functions to the total number of functions in a function tree is calculated.

- For measuring Design spaces

  Geodesic distances(GD) Metrics: *GD* Metrics include *Average Geodesic distances (AGD)*: the largest of the shortest path between verticals writing in a function tree. And the *Mean Geodesic distances(MGD)*: the mean number of the shortest paths between all the verticals in a function tree. The combination of *AGD* and *MGD* indicates the business and flatness of a function tree. A pair of high *GD*s indicates the business of the function tree, while a pair of low *GD*s indicates the flatness of the function tree.

  TotalLevel: the number of levels in a function tree, which indicates the depth of the design space morphologically.

  MaxNFperL: the largest number of functions in a level of the function tree, which indicates the breadth of the design space morphologically.

  MeanNFperL: the mean number of functions per level in a function tree. The statistical maximum can be sensitive to outliers and potential bias, the mean is also considered as an alternative measure of the exploration breadth of a function tree.

- Metric *ContextualCorrect* assesses a contextual correctness of the decomposition. It is conducted to assess if the parent–child function pairs in a function tree are domain-relevant and contextual correctness. *ContextualCorrect* represents number of parent–child function pairs express contextual correctness.

  % ContextualCorrect: The proportion of contextually correct parent–child function pairs to the total number of parent–child function pairs in a function tree is calculated.

This study aimed to address challenges in exploring the design space, expressing functions using the correct syntax format, and maintaining solution neutrality. To achieve this, we conducted a comprehensive review of all unique metrics used in previous research. At the end of the experiment, we collected a total of 632 unique functions across two groups. While most metrics were objective, the evaluation of syntax format, solution neutrality, and contextual correctness required subjective judgment. Therefore, two independent evaluators from the author list assessed these criteria separately and then compared their evaluations. The inter-rater consistency rates for these three metrics are all over 94%. Any discrepancies were resolved through discussion until a full agreement was reached.

TABLE 3: EVALUATION RESULTS OF FUNCTION TREES IN EACH GROUP

Metrics	Human		Machine		p-value
	Mean	SD	Mean	SD	
Vertices	15.00	4.31	**40.30**	11.38	0.000
Unique functions	13.93	4.18	**32.83**	09.36	0.000
Tree efficiency	**0.93**	0.11	0.82	0.05	0.000
Syntax errors	2.00	2.63	**0.03**	0.18	0.000
% Syntax error	0.13	0.17	**0.00**	0.00	0.000
SolutionNeutral	14.30	4.30	**39.93**	11.13	0.000
% SolutionNeutral	0.95	0.08	**0.99**	0.02	0.059
MGD	5.00	1.29	**9.80**	1.69	0.000
AGD	2.96	0.52	**4.81**	0.71	0.000
TotalLevel	2.83	0.83	**5.33**	0.76	0.000
MaxNFperL	7.07	2.15	**12.00**	3.51	0.000
MeanNFperL	3.95	0.96	**6.33**	1.51	0.000
Functions level 1	4.23	1.14	**4.33**	0.88	0.560
Functions level 2	6.83	2.35	**9.50**	3.03	0.000
Functions level 3	2.23	2.39	**10.76**	3.99	0.000
Functions level 4	0.63	1.56	**7.90**	3.95	0.000
Functions level 5	0.03	0.18	**4.80**	3.24	0.000
Functions level 6	0.00	0.00	**1.77**	2.69	0.000
ContextualCorrect	13.43	4.00	**33.67**	9.22	0.000
% ContextualCorrect	**0.96**	0.06	0.86	0.07	0.000

4.4. Results

The evaluation results of function trees between the human group and the machine group are shown in Table 3. A Shapiro-Wilk test was conducted to check for the normal distribution of results for all metrics. If the metric from two groups both follows normal distributions, the T-test is used to calculate the statistical differences; correspondingly, if the data is not normally distributed, the non-parametric test Kruskal-Wallis is used instead.

Based on Table 3, it can be observed that the ability to identify functions and assess tree efficiency varies across conditions. The machine group has significantly more vertices (mean=40.30), which is more than two times higher than the human group (mean=15.00). This indicates that the overall complexity of the function tree in the machine group is higher. The machine group also produces two times more unique functions than the human group (mean=32.83 vs. 13.93, p=0.000), which indicates the machine group produced more unique functions significantly. The

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89220/V03AT03A008/7557535/v03at03a008-detc2025-165946.pdf by guest on 12 February 2026

Copyright © 2025 by ASME; reuse license CC-BY 4.0

tree efficiency of both groups is higher than 70%, which indicates high efficiency in both groups, but the machine group showed lower efficiency than the human group, which suggests higher redundancy.

In terms of syntax correctness and solution neutrality, the human group makes significantly more *Syntax errors* (mean=2.00 vs. 0.03, p=0.000), which means the machine group is constrained strictly to the expected format. The results of *SolutionNeutral* metrics are also significant, as the rating of the machine group (mean=39.93) is much higher than the human group (mean=14.30). However, the ratios of solution neutral(*% SolutionNeutral*) do not show significant differences, which indicates that the performance of the machine group is only slightly better than the human group.

As the metrics for design space exploration, the *GD metrics*, *AGD*, and *MGD* are both significantly higher for machine-generated trees, which indicates that this function tree is bushier than the function tree that was human-generated. The machine group also showed a more significant performance than the human group regarding the design space exploration of the function tree. The metrics of *MaxNFperL* and *MeanNFperL* also show that the results from the machine group are significantly higher than the human group, which indicates that the function tree from the machine group is more broader than the human group, suggesting the breadth of design space exploration in the machine group is better than the human group. The *TotalLevel* metric on the function tree in the machine group (mean=5.33) is more than two times higher than the human group (mean=2.83), which suggests the level of the function tree in the machine group is significantly deeper than the human group, and the function tree generated by the machine group has deeper design space exploration. Additionally, the machine group consistently generates more functions at deeper levels, particularly from levels 3 to 6, indicating that machine-generated trees expand functions further into lower hierarchical levels. In contrast, human-constructed function trees are generally shallower, with few reaching as deep as level 4.

For contextual correctness, while the machine group produced more contextually correct parent-child function pairs in general, the contextual correctness rate was lower than that of the human group. Both differences are statistically significant.

Overall, the machine group generated more complex function trees with more vertices and unique functions while having better performance on constraints expected syntax format, but also less efficient due to redundancy. The tree efficiency is lower for the machine group, indicating more redundancy. The solution neutrality of the machine group and the human group is not statistically significant.

5. DISCUSSION

5.1. Result analysis

The experimental results provide insights into how the proposed LLM-based automated method improves functional decomposition across three key aspects: function identification, design space exploration, and adherence to syntax and solution neutrality principles.

Function Identification: AI-generated function trees contained significantly more unique functions than human-generated trees (mean = 32.83 vs. 13.93, p=0.000), suggesting that the method effectively broadens the functional coverage. This supports the idea that LLMs help mitigate cognitive biases that often lead designers to overlook certain functional aspects. However, the increase in function count also introduced redundancy, resulting in lower tree efficiency scores.

Design Space Exploration: AI-generated trees exhibited greater depth (mean = 5.33 vs. 2.83 levels, p=0.000) and broader structures (higher MaxNFperL and MeanNFperL scores), indicating a more systematic exploration of hierarchical relationships. Fig.4(a) illustrates this expansion, where the LLM-generated function tree contains multiple layers of decomposition, contrasting with the more compact structure in Fig.4(b). This suggests that the method promotes a more structured and exhaustive exploration of design functions.

Syntax and Solution Neutrality: The LLM-generated functions strictly adhered to Verb-Noun syntax and solution neutrality constraints, as expected. The machine-generated trees contained significantly fewer syntax errors (mean = 0.03 vs. 2.00, p=0.000), confirming improved structural compliance. However, while the absolute number of solution-neutral functions was higher, the overall proportion of solution-neutral functions did not significantly differ from human results. This indicates that designers may already possess an inherent understanding of neutrality principles, reducing the relative impact of LLM-based standardization.

In addition to quantitative results, designer feedback provides deeper insights into the cognitive challenges and benefits of AI-driven functional decomposition. Several participants struggled with identifying a comprehensive set of functions, often relying on external searches to supplement their knowledge. Another major challenge was expressing functions in the required Verb-Noun format while maintaining solution neutrality. Many participants reported that describing abstract functions without referencing specific design components was difficult. This suggests that while LLM-generated function trees provide structural consistency, additional scaffolding may be necessary to help designers articulate functions effectively. Despite these challenges, designers found that the structured decomposition process helped clarify functional relationships and improve design completeness. Some noted that function trees prevented early fixation on specific solutions, encouraging a more exploratory mindset. Others highlighted that defining functions hierarchically allowed them to iteratively refine and expand their design space, ensuring that critical aspects were not overlooked. These insights indicate that while LLMs assist in structuring decomposition, further refinements, such as interactive refinement mechanisms, could enhance usability.

5.2. Limitations and Future Work

While our method improves scalability and enables systematic decomposition, several limitations remain. **Function Basis Generalization**: Our function basis was built using only the "Industrial Design & Robotics" category from the Red Dot Award website, which may limit its applicability to other product types.

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89220/V03AT03A008/7557535/v03at03a008-detc2025-165946.pdf by guest on 12 February 2026

 Copyright © 2025 by ASME; reuse license CC-BY 4.0

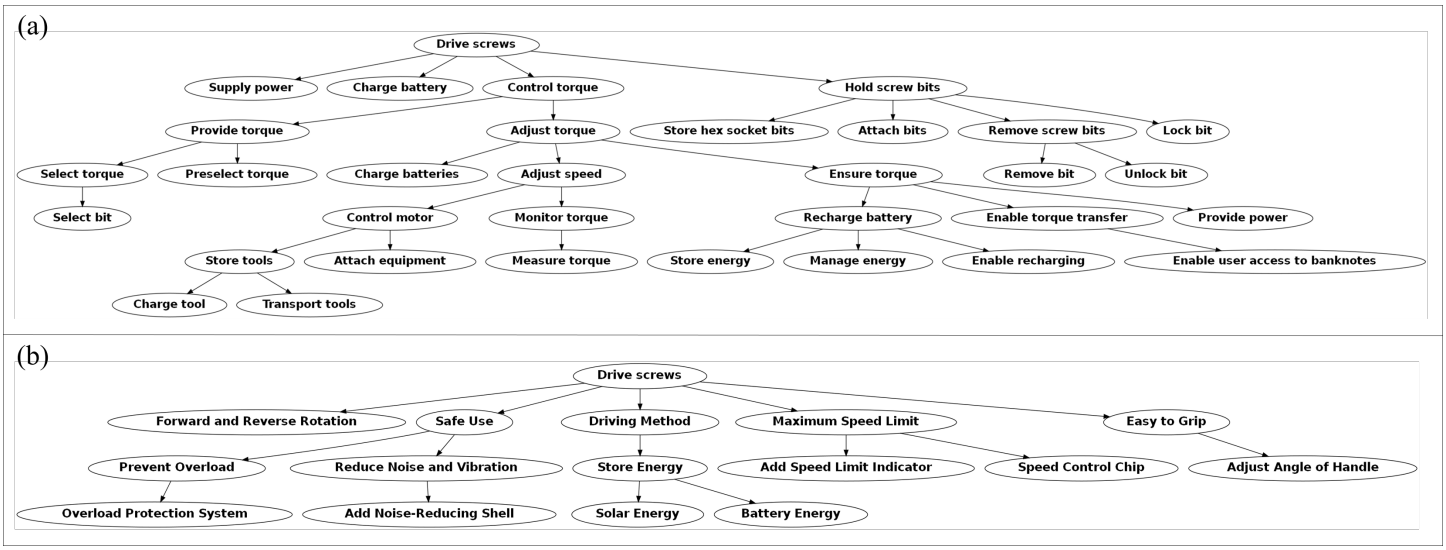


FIGURE 4: EXAMPLE FUNCTION TREES OF THE POWER SCREWDRIVER GENERATED BY (A) MACHINE GROUP AND (B) HUMAN GROUP

Future work should expand the data set to multiple categories to improve the generality and adaptability of the basis of the function. **Contextual error of the decomposition**: The current trees seem to have either errors or, at the very least, ambiguous vertices. For instance, in Fig 4 (a), "charge batteries" is provided as a subfunction of "adjust torque", which does not make sense. Even if such a relationship were arguable, "charge batteries" would more accurately be expressed as "store energy." The issue is also represented by Table 3, which shows that machine-generated trees have a lower contextual correctness rate than those created by humans. We suspect that hallucinations of GPT-4o are a contributor to the errors. Future work could leverage a more intelligent filtering mechanism (e.g. using another LLM), helping with the current rule-based filtering mechanism to deal with error subfunctions. **Stop Point Determination**: The current termination criterion for decomposition is simplistic, relying only on structural constraints. Prior studies [25] develop "rules" that map functions to physical components, which may provide a more reliable stopping condition for functional decomposition. Future approaches could incorporate component-function matching techniques or adaptive rule-based criteria to better define when a function has been fully decomposed. **Enhancing LLM Reasoning Capabilities**: Our method relies solely on prompt engineering without fine-tuning the LLM on domain-specific data. While this provides flexibility, it may also constrain the model's performance on tasks involving highly complex functional reasoning. Fine-tuning on a curated dataset of function trees could further improve accuracy and contextual understanding. Finally, future research could also explore how function trees can be integrated into later design stages, bridging the gap between functional decomposition and conceptual design development.

6. CONCLUSION

This study introduces an automated functional decomposition method using a Large Language Model (LLM) to overcome cognitive challenges in traditional decomposition, including abstracting functionalities, structured syntax adherence, and solution neutrality. A comprehensive function basis, derived from product and concept design documents, provides structured vocabulary. An iterative decomposition approach using an LLM-based decomposer and filtering mechanism systematically generates and filters subfunctions. A comparative experiment with fifteen participants who had engineering design-related backgrounds was conducted to evaluate the performance of function trees generated by the proposed method compared to human-generated ones. The results indicate that machine-generated function trees are significantly more complex and can better explore the design space. They also show improved syntax correctness and solution neutrality, highlighting the effectiveness of the LLM-based functional decomposition approach. This study and the arising methodology help reduce skill dependency and accelerate the functional decomposition process.

ACKNOWLEDGMENTS

This research is supported by National Key R&D Program of China (2022YFB3303304).

REFERENCES

[1] Eckert, Claudia, Alink, Thomas, Ruckpaul, Anne and Albers, Albert. "Different notions of function: results from an experiment on the analysis of an existing product." *Journal of Engineering Design* Vol. 22 No. 11-12 (2011): pp. 811–837.

[2] She, Jinjuan, Belanger, Elise, Bartels, Caroline and Reeling, Hunter. "Improve syntax correctness and breadth of design space exploration in functional analysis." *Journal of Mechanical Design* Vol. 144 No. 11 (2022): p. 111402.

[3] Garbacz, Pawel, Borgo, Stefano, Carrara, Massimiliano and Vermaas, Pieter E. "Two ontology-driven formalisations of

Copyright © 2025 by ASME; reuse license CC-BY 4.0

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89220/V03AT03A008/7557535/v03at03a008-detc2025-165946.pdf by guest on 12 February 2026

functions and their comparison." *Journal of Engineering Design* Vol. 22 No. 11-12 (2011): pp. 733–764.

[4] Atilola, Olufunmilola, Tomko, Megan and Linsey, Julie S. "The effects of representation on idea generation and design fixation: A study comparing sketches and function trees." *Design studies* Vol. 42 (2016): pp. 110–136.

[5] Booth, Joran W, Reid, Tahira N, Eckert, Claudia and Ramani, Karthik. "Comparing functional analysis methods for product dissection tasks." *Journal of Mechanical Design* Vol. 137 No. 8 (2015): p. 081101.

[6] Stone, Robert B and Wood, Kristin L. "Development of a functional basis for design." *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 19739: pp. 261–275. 1999. American Society of Mechanical Engineers.

[7] Bohm, Matt R and Stone, Robert B. "Product design support: exploring a design repository system." *ASME International Mechanical Engineering Congress and Exposition*, Vol. 47047: pp. 55–65. 2004.

[8] Yuan, Lin, Liu, Yusheng, Lin, Yunfeng and Zhao, Jianjun. "An automated functional decomposition method based on morphological changes of material flows." *Journal of Engineering Design* Vol. 28 No. 1 (2017): pp. 47–75.

[9] Bhasin, Devesh, McAdams, Daniel A and Layton, Astrid. "A product architecture-based tool for bioinspired function-sharing." *Journal of Mechanical Design* Vol. 143 No. 8 (2021): p. 081401.

[10] Eisenbart, Boris, Gericke, Kilian and Blessing, Lucienne TM. "Taking a look at the utilisation of function models in interdisciplinary design: insights from ten engineering companies." *Research in Engineering design* Vol. 28 (2017): pp. 299–331.

[11] Yuan, Lin, Liu, Yusheng, Sun, Zhongfei, Cao, Yanlong and Qamar, Ahsan. "A hybrid approach for the automation of functional decomposition in conceptual design." *Journal of Engineering Design* Vol. 27 No. 4-6 (2016): pp. 333–360.

[12] Chen, Chen, Tao, Ye, Li, Yan, Liu, Qiyu, Li, Song and Tang, Zhong. "A structure-function knowledge extraction method for bio-inspired design." *Computers in Industry* Vol. 127 (2021): p. 103402.

[13] Liu, Qiyu, Wang, Kai, Li, Yan, Chen, Chen and Li, Wenqiang. "A novel function-structure concept network construction and analysis method for a smart product design system." *Advanced Engineering Informatics* Vol. 51 (2022): p. 101502.

[14] Murphy, Jeremy, Fu, Katherine, Otto, Kevin, Yang, Maria, Jensen, Dan and Wood, Kristin. "Function Based Design-by-Analogy: A Functional Vector Approach to Analogical Search." *Journal of Mechanical Design* Vol. 136 No. 10 (2014): p. 101102.

[15] Fu, Katherine, Murphy, Jeremy, Yang, Maria, Otto, Kevin, Jensen, Dan and Wood, Kristin. "Design-by-analogy: experimental evaluation of a functional analogy search methodology for concept generation improvement." *Research in Engineering Design* Vol. 26 No. 1 (2015): pp. 77–95.

[16] Wenguang Lin, Yu Wang and Xiao, Renbin. "A hypergraph model of product function recombination based on online reviews." *Journal of Engineering Design* Vol. 34 No. 9 (2023): pp. 746–777.

[17] Liu, Longfan, Li, Yan, Xiong, Yan and Cavallucci, Denis. "A new function-based patent knowledge retrieval tool for conceptual design of innovative products." *Computers in Industry* Vol. 115 (2020): p. 103154.

[18] Lou, Shanhe, Feng, Yixiong, Gao, Yicong, Zheng, Hao, Peng, Tao and Tan, Jianrong. "A function-behavior mapping approach for product conceptual design inspired by memory mechanism." *Advanced Engineering Informatics* Vol. 58 (2023): p. 102236.

[19] Chen, Liuqing, Jing, Qianzhi, Tsang, Yixin, Wang, Qianyi, Sun, Lingyun and Luo, Jianxi. "DesignFusion: Integrating Generative Models for Conceptual Design Enrichment." *Journal of Mechanical Design* Vol. 146 No. 11 (2024): p. 111703.

[20] Chen, Liuqing, Zuo, Haoyu, Cai, Zebin, Yin, Yuan, Zhang, Yuan, Sun, Lingyun, Childs, Peter and Wang, Boheng. "Toward Controllable Generative Design: A Conceptual Design Generation Approach Leveraging the Function–Behavior–Structure Ontology and Large Language Models." *Journal of Mechanical Design* Vol. 146 No. 12 (2024): p. 121401.

[21] Chen, Liuqing, Cai, Zebin, Jiang, Zhaojun, Luo, Jianxi, Sun, Lingyun, Childs, Peter and Zuo, Haoyu. "AskNatureNet: A divergent thinking tool based on bio-inspired design knowledge." *Advanced Engineering Informatics* Vol. 62 (2024): p. 102593.

[22] Reimers, Nils and Gurevych, Iryna. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*: pp. 3982–3992. 2019.

[23] Johnson, Jeff, Douze, Matthijs and Jégou, Hervé. "Billion-scale similarity search with GPUs." *IEEE Transactions on Big Data* Vol. 7 No. 3 (2019): pp. 535–547.

[24] Cao, Guozhong, Sun, Yindi, Tan, Runhua, Zhang, Jinpu and Liu, Wei. "A function-oriented biologically analogical approach for constructing the design concept of smart product in Industry 4.0." *Advanced Engineering Informatics* Vol. 49 (2021): p. 101352.

[25] Kurtoglu, Tolga, Campbell, Matthew I and Linsey, Julie S. "An experimental study on the effects of a computational design tool on concept generation." *Design Studies* Vol. 30 No. 6 (2009): pp. 676–703.

[26] Chen, Liuqing, Cai, Zebin, Cheang, Wengteng, Long, Qi, Sun, Lingyun, Childs, Peter and Zuo, Haoyu. "AskNatureGPT: An LLM-driven concept generation method based on bio-inspired design knowledge." *Journal of Engineering Design* (2025): pp. 1–31.

Downloaded from http://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89220/V03AT03A008/7557535/v03at03a008-detc2025-165946.pdf by guest on 12 February 2026

 Copyright © 2025 by ASME; reuse license CC-BY 4.0