Proceedings of the Design Society, Volume 5: ICED25
https://doi.org/10.1017/pds.2025.10194

Finding optimal solution principles in conceptual design

Philipp Rosenthal✉, **Artur Liebert and Oliver Niggemann**

Helmut Schmidt University, Germany

✉ philipp.rosenthal@hsu-hh.de

ABSTRACT: Automating the structuring of Solution Principles within conceptual design is crucial for efficiently covering Function Structures while reducing time-intensive manual processes. Solution Principles are central in bridging functional requirements and technical implementations, yet traditional methods depend heavily on human expertise. To address this, a novel approach leveraging a search algorithm is proposed to automatically identify an optimal set of Solution Principles for a given Function Structure. The approach formalizes the problem and provides rules for the selection and application of Solution Principles. Key components include a function for applying Solution Principles to functions and a heuristic that minimizes principle selection, guiding the search toward optimal solutions. An evaluation shows the potential of this method to reduce time and effort in early product design.

KEYWORD: Conceptual design, Artificial intelligence, Computational design methods, Solution Principles, Function Structures

1. Introduction

Recent advancements in Artificial Intelligence (AI) have reshaped engineering design, particularly during the early conceptual stages, where traditional methods and cognitive limitations often constrain the exploration of expansive design spaces (Allison et al., 2022). The early design phase plays a pivotal role in product development, laying the groundwork for the entire process and significantly influencing the final product's performance, cost, and feasibility. During this phase, design engineers transition from abstract functional requirements to concrete solutions, commonly referred to as the "conceptual design" phase (Bender and Gericke, 2021). A critical aspect of this phase is identifying Solution Principles that fulfill specific functional requirements (Hubka and Eder, 1996).

The concept of Solution Principles plays a central role in engineering design, serving as the bridge between abstract functions and their physical realizations. Rooted in the foundational work of Pahl and Beitz (Bender and Gericke, 2021), they are described as technical concepts that combine physical effects with geometrical and material characteristics to fulfill specific functions. Unlike working principles, Solution Principles avoid detailing specific physical specifications, focusing instead on their functional essence. According to Rot. (2000), the realization of a function is inherently tied to an effect, a relationship formalized as the function-effect-law. This law underscores the necessity of linking functions to effects, as no function can be practically realized without one. Complementing this perspective, the German VDI guideline (Verein Deutscher Ingenieure, 1997) defines principal solutions as basic solutions for delimited design tasks that outline modes of operation without specifying them in detail. These principal solutions, referred to as Solution Principles when detached from specific construction tasks, provide a systematic means of addressing design challenges. In practice, Solution Principles encompass a wide array of technical solutions, including mechanisms, entire systems, or processes that address individual functions within a product's overall structure.

https://doi.org/10.1017/pds.2025.10194 Published online by Cambridge University Press

The classical method in conceptual design for systematically finding and structuring Solution Principles begins with a Function Structure as a starting point. A Function Structure is created by breaking down the overall function of a product into manageable subfunctions, a process known as functional decomposition. This decomposition is described by Pahl and Beitz (Bender and Gericke, 2021) and is also present in the VDI guideline (Verein Deutscher Ingenieure, 2019a,b). Then the systematic exploration for Solution Principles often involves searching in catalogs. This approach is supported by Roth (2000, 2001) and Koller and Kastru. (1998). While these catalogs reduce the effort of finding Solution Principles for given functions, significant human effort is required to interpret the descriptions and adapt them for the Function Structure.

The process of identifying suitable Solution Principles for Function Structures is tedious and complex due to the vast number of possible combinations and interactions between functions and Solution Principles. Each function may have multiple potential solutions, leading to a combinatorial explosion of possibilities (Ehrlenspiel and Meerkamm, 2017). Constraints, such as input and output requirements, further restrict the search space, making the lack of systematic methods for automating this process particularly limiting. The result is often a trial-and-error approach, consuming considerable time and resources.

Automating the identification of Solution Principles has the potential to significantly reduce the manual effort involved, leading to more efficient design processes. However, despite advancements in computational design tools, identifying appropriate Solution Principles remains a major challenge. The complexity arises from the vast solution space and the interdependencies between functions within a product's Function Structure. While algorithms and AI-based tools have been developed to assist designers in exploring potential solutions, the process is still not fully automated or reliable enough to consistently deliver optimal solutions. To address this challenge, three key research questions are derived:

- *RQ 1*: How do Solution Principles and their structures need to be defined to make them accessible for AI algorithms to find optimal Solution Principle Structures?
- *RQ 2*: What set of rules exist that guide the application of Solution Principles on functions in a Function Structure?
- *RQ 3*: What algorithm is suitable for finding an optimal Solution Principle Structure given a Function Structure?

This work introduces the Solution Principle Structure Finder, a new algorithm, which efficiently identifies optimal Solution Principle Structures. The algorithm is supported by a formalized framework for defining and applying Solution Principles in conceptual design, demonstrating scalability and robustness across Function Structures of varying complexity. The algorithm is evaluated on synthetic Function Structures, assessing its efficiency in identifying optimal Solution Principle Structures.

2. Related work

Recent work by Khanolkar et al. (2023) systematically maps AI methods across the stages of engineering design, emphasizing their critical role in the conceptual design phase. This phase, marked by high uncertainty and iterative problem-solving, relies on methods, which can be subdivided into symbolic and subsymbolic approaches.

The symbolic approaches often revolve around functional reasoning and expand upon the generation of solutions and their structures. These methods were mainly driven by Chakrabarti and Bligh (1994, 1996). Later the authors tested several of these models and concluded that none are guaranteed to find solution concepts, i.e. Solution Principles and their structures on their own (Chakrabarti and Bligh, 2001). Bhatta and Goe. (1994) introduce the Structure-Behavior-Function (SBF) modeling language, which formalizes the link between structural elements, causal behaviors, and functions within complex systems, offering a structured approach similar to the organization of Solution Principles within Function Structures. SBF's emphasis on causal mechanisms parallels the role of Solution Principles in bridging abstract functions and concrete implementations, highlighting the need for systematic, interpretable models in design. Additionally, an interactive tool, SBFAuthor, supports model refinement, underscoring the value of clear, adaptable structures that facilitate human understanding in AI-assisted design (Goel et al., 2009). Kitamura and Mizoguch. (2003) developed a framework to systematically represent knowledge about functional decomposition and functional achievement.

https://doi.org/10.1017/pds.2025.10194 Published online by Cambridge University Press

Subsymbolic approaches are methods that represent and process knowledge not through explicit symbols or logical rules but through distributed, numerical representations, such as neural networks, and are driven by data. Rosenthal and Niggeman. (2021) examine the limitations of AI methods in conceptual product design, highlighting challenges that subsymbolic approaches encounter, particularly with problem characteristics like decomposability and interdependencies. In complex design problems, where functions must be decomposed and interrelated, subsymbolic methods, such as neural networks, often struggle with interpretability and flexibility, functioning more as "black-box" models. The authors emphasize that, while data-driven techniques excel in recognizing patterns, they frequently rely on existing knowledge bases. This reliance underscores a key limitation in early-stage design, where minimal predefined information is available. Jiang et al. (2021) examine the use of design-by-analogy methods that apply data-driven approaches, particularly machine learning and neural networks, to support designers in identifying Solution Principles across domains. Their work addresses subsymbolic methods for retrieving analogous Solution Principles. However, it points out the challenging need for extensive, well- curated datasets, which are usually not widely available. Berton. (2020) also highlights the shortcomings of data driven approaches in concept development. These mainly revolve around the lack of proactive data collection.

Despite the structured approach provided by functional decomposition and construction catalogs, identifying and adapting Solution Principles remains a labor-intensive process. While symbolic approaches have contributed to functional reasoning, none directly addresses the automatic generation of Solution Principle Structures for given Function Structures. Recent developments in subsymbolic design synthesis have aimed to automate parts of this process. However, the early stages of conceptual design are marked by high uncertainty, and relevant data are sparse and not readily available. Additionally, as the design process is largely team-oriented and involves various human stakeholders, there is a critical need for methods that are interpretable and comprehensible. Subsymbolic algorithms generally lack transparency, making them challenging for designers to interpret. This highlights the need for AI tools in conceptual design that not only assist in finding Solution Principles and their structures but are also accessible and understandable to design teams.

3. Solution

This paper aims to find optimal Solution Principle Structures for given Function Structures by representing the problem as a search problem. A Solution Principle Structure is then correct, if and only if every function in a Function Structure is covered by a Solution Principle leading to the fulfillment of the function. A minimal number of Solution Principles is optimal as it reduces the effort required for detailing and embodiment in later design stages. So the goal state is an optimal Solution Principle Structure.

Crucial for the presented solution are Roth Functions (Roth, 2000). These are a set of 30 generalized functions used in engineering design to represent the fundamental operations a product or system must perform to fulfill its intended purpose. Developed as part of Karlheinz Roth's framework for product design, these functions provide a standardized way of breaking down complex product functions into manageable, well-defined operations. Roth Functions are particularly useful in conceptual design and Function Structure modeling because they offer a common language for designers to describe what a system needs to do in terms of transforming entities, i.e. material, energy and information.

3.1. Problem formalization

Definition 1. *The **Roth Function Set** R is the set of all Roth Functions*:

$$R = \{f \mid f \text{ represents a Roth Function } \}$$

Definition 2. *A Structure graph S is a directed graph $S = (V, E)$ with*:

- $V = \{(f, c) \mid f \in R, c \in \{0, 1\}\}$, *with Roth Function Set R*
- $E = \{(u, w) \mid u, w \in V\}$, *where E is subject to the combination rules of the Roth Functions.*

https://doi.org/10.1017/pds.2025.10194 Published online by Cambridge University Press

Definition 3. *A **Solution Principle** is a function* $sp : \mathcal{S} \rightarrow \mathcal{S}$, *where* $\mathcal{S} = \{S \mid S \text{ is a Structure graph}\}$, *such that applying sp to a Structure graph* $S = (V, E)$ *results in a new Structure graph* $S' = (V', E')$, *denoted as* $sp\ (S) = S'$, *with the following property*:

(a) *sp updates up to two specific vertices* $v_1, v_2 \in V$, *where:*

$$v_i = (f, 0) \mapsto v_i' = (f, 1) \in S', i \in \{1, 2\}$$

*meaning that f is **covered by** the Solution Principle sp*

Additionally sp may satisfy one, both or neither of the following properties:

(b) *sp adds a vertex set* V^+ *and a correspondent edge set* E^+ *to S, hence:*

$$S' = S \cup (V^+, E^+)$$

with

$$V^+ = \{(f, c) \mid f \in R, c \in \{0, 1\}\} \text{ and } E^+ = \{(u, w) \mid u \in V \cup^V +, w \in V^+\}$$

(c) *sp removes a vertex set* $V^- \subset V$ *and the correspondent edge set* $E^- \subset E$, *hence:*

$$S' = S \backslash (V^-, E^-)$$

where $V^- = \{(f, 0) \mid (f, 0) \in V\}$ *and* $E^- = \{(u, w) \mid u \in V, w \in V^-\}$

Definition 4. *A **Function Structure*** S_0 *is a Structure graph* $S_0 = (V_0, E_0)$, *where none of the Roth Functions f R are covered by a Solution Principle sp yet, i.e.*:

$$\forall v \in V_0 : v = (f, 0),\ f \in R$$

Definition 5. *A **Solution Principle Structure*** S_1 *is a Structure graph* $S_1 = (V_1, E_1)$, *where each Roth Function f R in* V_1 *is covered by a Solution Principle sp, i.e.*:

$$\forall v \in V_1 : v = (f, 1),\ f \in R$$

Definition 6. *A **valid Solution*** sp_s *is a finite ordered sequence of Solution Principles* $\{sp_0, \ldots, sp_n\}$, *so that when applied to the initial Function Structure S0 the output is a Solution Principle Structure S1, i.e.*:

$$sp_s = \{sp_0, \ldots, sp_n\}, n \in N,\ > \text{ with } sp_s(S_0) = (sp_n \circ\ sp_{n-1} \cdots^{\circ} sp_0)(S_0) = S_1$$

Definition 7. *The **Goal Test** is a function* $\mathcal{S} \rightarrow \{0, 1\}$. *It checks for a given Structure graph* $S \in \mathcal{S}$ *whether it is a Solution Principle Structure. If that is the case it returns* 1 *otherwise* 0.

The **Optimization Problem** can be formalized as finding an optimal Solution Principle Structure $S_1^* = sp_S^*(S_0)$, where S_1^* is the output of a Solution sp_s^* with minimal sequence length, i.e.:

$$sp_s^* = \min_{n \in N}\{\{sp_0, \ldots, sp_n\} \mid \{sp_0, \ldots, sp_n\} \text{is a solution to } S_0\}$$

This section directly answers *RQ 1*. By using these formalizations, the problem of finding optimal Solution Principle Structures is made accessible for AI algorithms.

3.2. Rules for applying solution principles

When applying a Solution Principle to a function, the Solution Principle itself may influence the Structure graph, as shown in Definition 3. This influence arises from constraints that may come with the Solution Principle and is rooted in how Roth Functions can connect to each other. These constraints can be expressed as additional functions that need to be fulfilled for the Solution Principle to work.

A practical example would be the conversion of energy from electrical to kinetic in a Battery Electric Vehicle. A viable Solution Principle might be an electric motor. This motor primarily transforms electrical energy into kinetic energy to drive the vehicle. However, due to electromagnetic induction, electric motors inherently transform kinetic energy back into electrical energy when rotating under specific conditions. This means that applying an electric motor as a Solution Principle may also cover the function of braking by transforming kinetic energy into electrical energy. It also introduces a new function of transmitting the generated electrical energy to the battery for storage. Thus, the electric motor exemplifies how a Solution Principle can both fulfill an additional intended function and introduce a new function as a consequence of its inherent physical constraints.

The intended algorithm will need a central function that can apply Solution Principles to structures with respect to the combination rules of Roth Functions. This central function is responsible for applying a

https://doi.org/10.1017/pds.2025.10194 Published online by Cambridge University Press

specific Solution Principle to a target function within the structure, updating the structure by covering that function and modifying other elements in the structure as needed. The central function operates by first applying the selected Solution Principle to a specified target function, then updating the structure to reflect the new relationships introduced by this application. Each Solution Principle includes conditions that dictate its usage, involving entities, input and output constraints.

When a Solution Principle is applied to a target function, the coverage attribute of the target function is updated. This action marks the function as addressed within the current structure, which is crucial for tracking the overall coverage.

Input constraints define the prerequisites that must be met before a Solution Principle can be applied to a function, like in the electric motor, electric energy needs to be transmitted from its source.

If the constraint entity matches with the function, the constraint function is added between the function covered by the Solution Principle and its predecessors in the Function Structure (Figure 1a).

When the entities do not match, the constraint function gets added to the structure without a successor. In the case that it is not a store function, such a function is added as a source for the entity (Figure 1b). Input constraints may involve combine functions that require specific input configurations. If an input constraint requires a combine operation, it will always involve the function entity and an additional one. A source function needs to be added in that case as well (Figure 1c).

Output constraints specify the expected conditions for downstream functions after the application of a Solution Principle. If such an output constraint does not match any uncovered function in the structure it gets added to the structure (Figure 1d).

It is important to point out that output constraints, in contrast to input constraints, are always covered by the Solution Principle itself. This is because they are a result and not a prerequisite of the Solution Principle. When looking at the electric motor again, thermal energy may be transmitted as a result of resistive losses. In case that an output constraint is similar to an uncovered function in the structure it replaces that function. In turn any of the predecessors that are not covered may get pruned from the structure (Figure 1e).

Output constraints may include split operations, where a function's output is distributed among multiple successors. When an output constraint involves a split function, it gets inserted between the target function and its successors, as needed. This ensures the split function aligns with the main entity associated with the Solution Principle. Then the successor links get updated, inserting the split function as required and adjusting the successors of the split function accordingly (Figure 1f).

This ensures that the output conditions created by the Solution Principle application propagate correctly through the structure.

In general, applying a Solution Principle can be defined as updating a structure by marking a function as covered, adding required functions or connections, and removing any redundant elements, ensuring the structure remains consistent. Additionally a unique identifier is assigned to each new function, to ensure compatibility within the structure. This identifier makes it also comprehensive for human designers.

This set of rules gives an answer to *RQ 2*. By adhering to these rules, handled by a central function, Solution Principles will be applied in a correct way on functions within a Function Structure.

3.3. Search strategy

An adaptation of the A* search algorithm is used to automatically identify optimal Solution Principle Structures for a given Function Structure. A state in the sense of search algorithms represents a Structure graph S as described in Definition 2. An action is to select a Solution Principle to cover a function. The goal is to cover all functions in the Function Structure with as few Solution Principles as possible. The cost function $g(S)$ is the number of Solution Principles chosen so far. The heuristic function $h(S)$ considers the number of uncovered functions. It is important to point out that $h(S)$ needs to be admissible (Pearl, 1984). Since every function could be covered by a dedicated Solution Principle, this heuristic will be admissible in regards of minimizing the number of chosen Solution Principles. The algorithm will choose the Structure graph S that minimizes the function $f(S) = g(S) + h(S)$.

3.4. The search algorithm

Russell and Norvi. (2021) give a good overview of informed search algorithms. The present approach leverages the knowledge about the A* search algorithm in particular and adapts it to solve the problem of

https://doi.org/10.1017/pds.2025.10194 Published online by Cambridge University Press

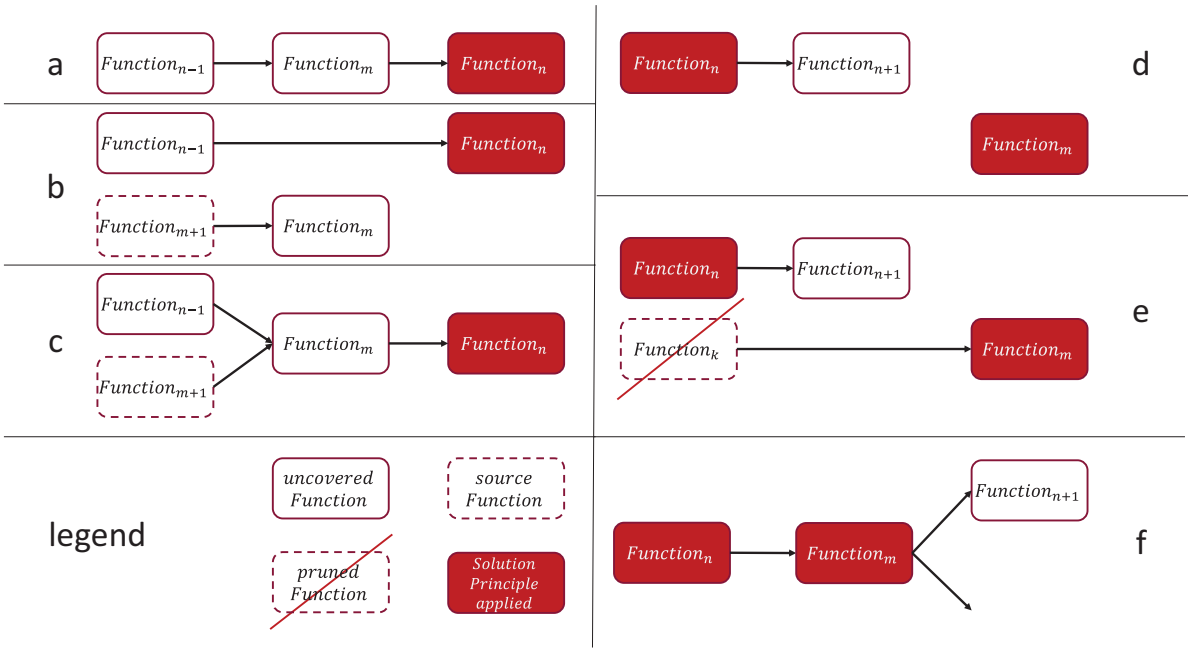


Figure 1. Rules for handling constraints when applying Solution Principles to functions

finding Solution Principle Structures. This similarity ensures that the presented algorithm is both sound and complete. In addition it also guarantees an optimal solution (Pearl, 1984). The algorithm applied to find Solution Principle Structure is shown in Algorithm 1.

Input: S_0: Function Structure, SP: Set of Solution Principles
Output: Optimal Solution Principle Structure S_1^*

1: $OpenSet \leftarrow \{(S_0, calculate_cost(S_0))\}$
2: $ClosedSet \leftarrow \{\}$
3: **while** $OpenSet \neq \emptyset$ **do**
4: $(CurrentStructure, f_cost) \leftarrow$ pop lowest-cost structure from $OpenSet$
5: **if** $goal_test(CurrentStructure)$ **then return** $CurrentStructure$
6: **end if**
7: $ClosedSet \leftarrow ClosedSet \cup \{CurrentStructure\}$
8: $UncoveredFunctions \leftarrow \{f \in S : S[f].\text{covered_by} = \emptyset\}$
9: **for** each $function \in UncoveredFunctions$ **do**
10: $ApplicablePrinciples \leftarrow \{sp \in SP : sp.\text{covers}(function)\}$
11: **for** each $principle \in ApplicablePrinciples$ **do**
12: $SuccessorStructure \leftarrow apply_principle(CurrentStructure, function, principle)$
13: **if** $SuccessorState \notin ClosedSet$ **then**
14: $f_cost \leftarrow calculate_cost(SuccessorStructure)$
15: add $(SuccessorStructure, f_cost)$ to $OpenSet$
16: **end if**
17: **end for**
18: **end for**
19: **end while**
20: **return** No solution found

Algorithm 1 Solution Principle Structure Finder

The input needed for the Solution Principle Structure Finder (SPSF) to work is a viable Function Structure S_0 consisting of Roth Functions and the set of Solution Principles SP. The algorithm begins by initializing the *OpenSet* with a single element: the initial Function Structure S_0, paired with its calculated cost $f(S_0)$. The *ClosedSet* in line 2 starts as an empty set, which will later store explored structures to avoid redundant calculations. As long as the *OpenSet* is not empty, the algorithm proceeds.
In line 4, it selects the structure in *OpenSet* with the lowest cost, assigning it to *CurrentStructure* and storing its associated cost. This follows the A* principle of expanding the most promising node first. Next, line 5 checks if the current structure satisfies the goal condition. The goal is achieved if all

https://doi.org/10.1017/pds.2025.10194 Published online by Cambridge University Press

functions within the Structure graph *S* are covered by Solution Principles, as described in Definition 7. If *goal_test* returns true, the algorithm terminates, returning *CurrentStructure* as the optimal solution.
If the goal is not yet met, line 7 adds *Current Structure* to the *ClosedSet*, ensuring that this structure is not revisited, which helps enhance efficiency by avoiding redundant calculations. The algorithm then identifies all uncovered functions within the current structure *S* in line 8. These functions are stored in the set *Uncovered Funct ions*, containing all functions in *S* whose *covered_by* attribute remains empty. For each *f unction* in *Uncovered Funct ions*, the algorithm proceeds to line 10 to determine which Solution Principles in the set *SP* are applicable. The set *ApplicablePrinciples* holds Solution Principles that can cover the selected *f unction*, considering any constraints associated with each principle.
At line 11, the algorithm iterates over each *principle* in *ApplicablePrinciples*. For each *principle*, it generates a new structure, *SuccessorStructure*, by applying the Solution Principle to the *f unction*. The function *apply_principle* described in section 3.2 handles this application, creating a new state that respects the constraints of the applied principle. Line 13 ensures that *SuccessorSt ructure* has not already been explored by checking if it exists in *ClosedSet*. If *SuccessorStructure* is in *ClosedSet*, it is skipped to avoid redundant calculations. Otherwise, the algorithm proceeds to line 14. Here, the $f(S)$ of *SuccessorStructure* is calculated using *calculate_cost* described in section 3.3. This cost helps prioritize structures in the *OpenSet*. Afterward, in line 15, *SuccessorStructure* along with its calculated cost is added to *OpenSet* for further exploration.
Once all functions in *Uncovered Functions* and all applicable Solution Principles have been evaluated, the algorithm loops back to line 4, continuing to expand the next lowest-cost structure in *OpenSet*. This process repeats until either all functions are covered and the goal is reached, or the *OpenSet* becomes empty. Finally, if no solution is found (i.e., *OpenSet* is empty), the algorithm terminates in line 20, returning "No solution found." This outcome indicates that it was not possible to cover all functions within the Function Structure S_0 using the available Solution Principles in *SP*.
The described algorithm SPSF is the answer to *RQ 3*. SPSF is able to solve the problem of finding optimal Solution Principle Structures for given Function Structures by leveraging the knowledge about A* search.

4. Experimental results

To evaluate the algorithm, synthetic Function Structures and Solution Principles were used to ensure systematic and scalable testing. Real-world Function Structures are often proprietary and they tend to be anecdotal, limiting their suitability for reproducible and comprehensive analysis. Similarly, catalogs of Solution Principles in the proposed formalized manner do not currently exist yet, necessitating the use of synthetic Solution Principles to build a consistent and accessible knowledge base. The synthetic data was generated using two custom algorithms that adhere to the formalizations in section 3.1 and were designed to randomly create Function Structures and Solution Principles, ensuring diversity and replicability. This approach offers precise control over complexity, enabling thorough testing across a wide range of scenarios. It also ensures comparability and neutrality, avoiding the biases and inconsistencies often present in real-world examples.

4.1. Case study

The application of Algorithm 1 is demonstrated on a synthetic example Function Structure with 13 interconnected nodes, representing distinct Roth Functions. The example illustrates how the SPSF systematically applies Solution Principles to achieve complete function coverage while adhering to constraints and optimizing for a minimal amount of chosen Solution Principles. The applied Solution Principles are enlisted in Table 1 and stem from a synthetic Solution Principle catalog with 10 individual Solution Principles for each of the 30 Roth Functions.

https://doi.org/10.1017/pds.2025.10194 Published online by Cambridge University Press

Table 1. Solution Principles

Name	Input Constraint	Output Constraint	SP
...	...	...	...
store energy 8	transform information	shape material	(a)
store information 6			(b)
split energy with material 2	transform information	shape material	(c)
transmit energy 2			(d)
combine material with information 10		transform information	(e)
transmit material 8		transform information	(f)
...	...	...	...

Figure 2 depicts the initial Function Structure as well as the generated Solution Principle Structure. Algorithm 1 was able to simplify the structure and cover all function with only 6 distinct Solution Principles. To achieve this, the functions *store material1*, *split information with material6* and *transmit material8* were pruned from the structure. However, the two functions *transform information14* and *transform information16* had to be added to adhere to the constraints of the chosen Solution Principles. By choosing Solution Principles that also cover the added functions an optimal result was achieved.

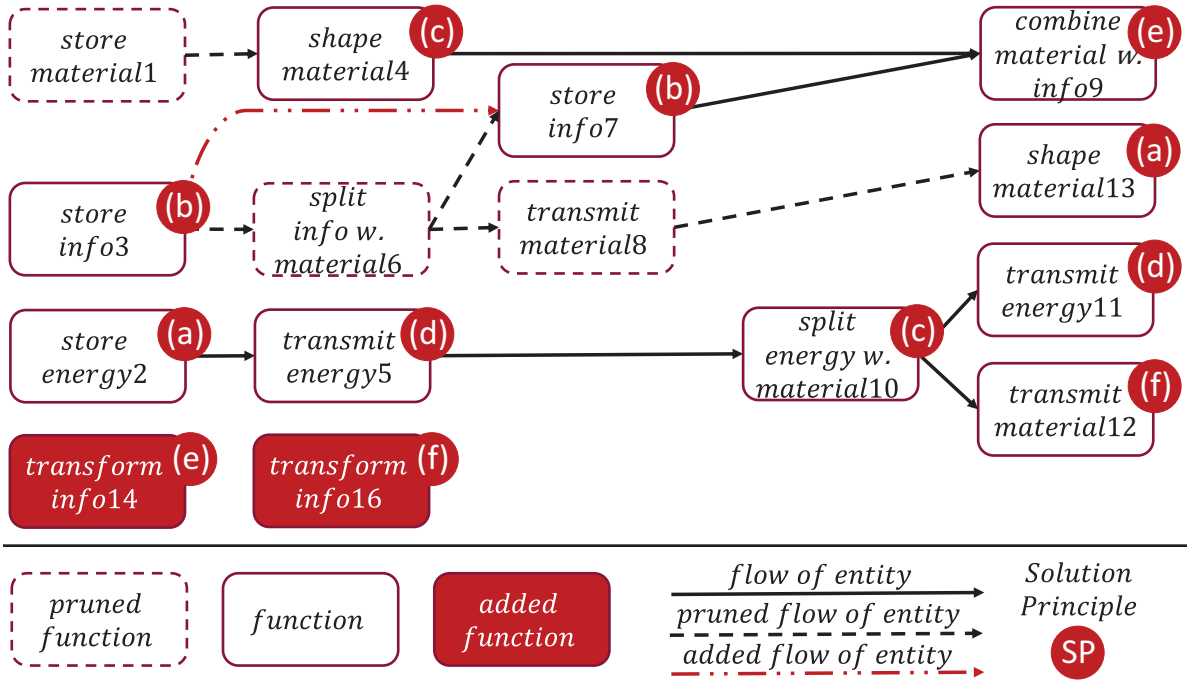


Figure 2. Solution Principle Structure for an initial Function Structure with 13 functions

4.2. Empirical performance testing

The empirical evaluation of Algorithm 1 was conducted using a synthetic dataset comprising 40 Function Structures, evenly distributed across four categories based on size: 10, 20, 50, and 100 functions. For each size category, 10 distinct Function Structures were generated to ensure diversity in complexity and topology. All experiments were conducted on a system with an Intel Core i5-10210U CPU @ 1.6 GHz, 8 GB DDR4 RAM (2667 MHz) and a 256 GB NVMe SSD.

The evaluation focused on three primary metrics: the number of Solution Principles chosen to achieve full function coverage, the total number of covered functions, and the execution time required for each instance. Figure 3 illustrates the relationship between the number of covered functions, the number of chosen principles, and the execution time for each Function Structure. With the latter being depicted in color gradient and size.

https://doi.org/10.1017/pds.2025.10194 Published online by Cambridge University Press

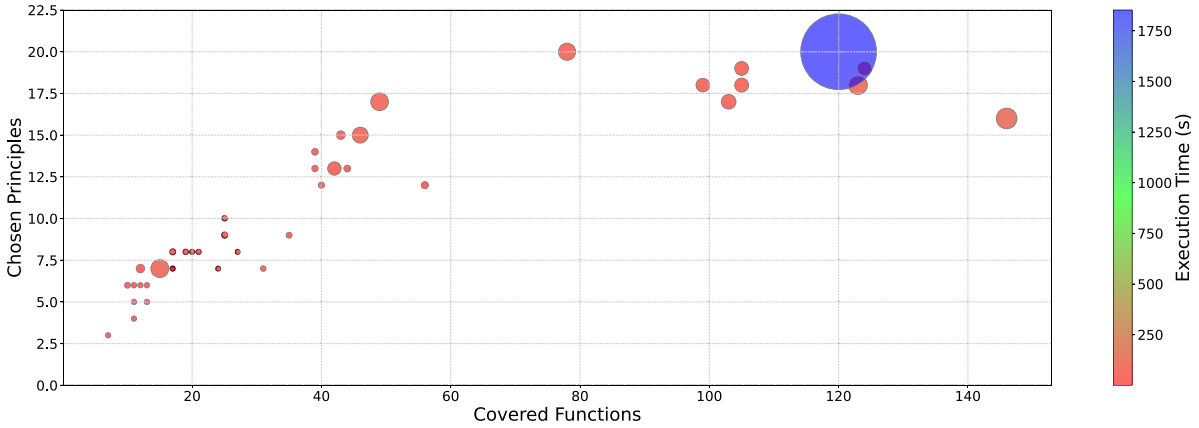


Figure 3. Relationship between Solution Principles, covered functions, and execution time

The results showcase SPSF's scalability and robustness across varying Function Structure complexities. For smaller Function Structures (10 and 20 functions), SPSF efficiently identified minimal sets of Solution Principles, resulting in shorter execution times. In contrast, larger structures (50 and 100 functions) exhibited greater variation in execution time, as seen in the spread of larger bubbles towards the upperright corner of the plot. This behavior is attributed to the increasing complexity of the search space and the interconnectedness of functions, leading to more iterations and is expected.

Notably, an outlier is visible in the upper-right corner of the plot, corresponding to a Function Structure with significantly higher execution time. This outlier suggests that specific structural properties of Function Structures, such as densely interconnected functions or an imbalance in constraints, can heavily influence computational performance.

Overall, the testing demonstrated the algorithm's capability to consistently handle diverse Function Structures. The execution times indicate that the algorithm performs predictably across most instances, with only a few edge cases requiring disproportionately high computational effort while still finding an optimal Solution Principle Structure.

5. Conclusion and outlook

This paper introduces a novel approach for automating the identification and structuring of Solution Principles in conceptual design, addressing key challenges in early-stage product development. The formalization presented in section 3.1 directly addresses *RQ 1*, making the problem accessible to AI algorithms. Section 3.2 introduces a structured set of rules that guide the application of Solution Principles, effectively answering *RQ 2*. Leveraging the A* search algorithm, the SPSF described in section 3.4 efficiently identifies minimal sets of Solution Principles to achieve full coverage of Function Structures, providing a robust answer to *RQ 3*.

The algorithm demonstrated scalability and robustness across synthetic Function Structures of varying complexities, from 10 to 100 functions. While occasional computational outliers emerged, the method consistently delivered optimal solutions, significantly reducing the manual effort required in the conceptual design phase. By adhering to standardized formalizations like Roth Functions and systematically managing constraints, the approach ensures transparency and interpretability, critical for adoption by design teams. Empirical evaluations with synthetic datasets validate its effectiveness and demonstrate its capacity to handle diverse scenarios systematically.

Nevertheless, several avenues for future research remain. Validation using real-world datasets is a priority, as it would bridge the gap between synthetic testing and practical application. The development of curated datasets of Function Structures and Solution Principles could enable broader applicability. Further exploration of alternative heuristics may enhance computational efficiency, while integrating user-defined preferences or additional constraints, such as cost or manufacturability, could increase the method's practical relevance. These steps will not only improve the proposed approach but also advance the integration of AI in conceptual design.

https://doi.org/10.1017/pds.2025.10194 Published online by Cambridge University Press

References

Allison, James T., Cardin, Michel-Alexandre, McComb, Chris, Ren, Max Yi, Selva, Daniel, Tucker, Conrad, Witherell, Paul, and Zhao, Yaoyao Fiona. Special Issue: Artificial Intelligence and Engineering Design. *Journal of Mechanical Design*, 144 (2), 2022. ISSN 1050-0472. http://dx.doi.org/10.1115/1.4053111.

Bender, Beate and Gericke, Kilian, editors. *Pahl/Beitz Konstruktionslehre: Methoden und Anwendung erfolgreicher Produktentwicklung*. Springer eBook Collection. Springer Vieweg, Berlin and Heidelberg, 9. auflage edition, 2021. ISBN 9783662573037. http://dx.doi.org/10.1007/978-3-662-57303-7.

Bertoni, Alessandro. DATA-DRIVEN DESIGN IN CONCEPT DEVELOPMENT: SYSTEMATIC REVIEW AND MISSED OPPORTUNITIES. *Proceedings of the Design Society: DESIGN Conference*, 1:101–110, 2020. ISSN 2633-7762.

Bhatta, Sambasiva R. and Goel, Ashok K. Discovery of physical principles from design experiences. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 8 (2):113–123, 1994. ISSN 0890-0604.

Chakrabarti, Amaresh and Bligh, Thomas P. An approach to functional synthesis of solutions in mechanical conceptual design. Part I: Introduction and knowledge representation. *Research in Engineering Design*, 6 (3): 127–141, 1994. ISSN 0934-9839. http://dx.doi.org/10.1007/BF01607275.

Chakrabarti, Amaresh and Bligh, Thomas P. An approach to functional synthesis of solutions in mechanical conceptual design. Part II: Kind synthesis. *Research in Engineering Design*, 8 (1):52–62, 1996. ISSN 09349839. http://dx.doi.org/10.1007/BF01616556.

Chakrabarti, Amaresh and Bligh, Thomas P. A scheme for functional reasoning in conceptual design. *Design Studies*, 22 (6):493–517, 2001. ISSN 0142-694X.

Ehrlenspiel, Klaus and Meerkamm, Harald. *Integrierte Produktentwicklung: Denkabläufe, Methodeneinsatz, Zusammenarbeit*. Hanser eLibrary. Carl Hanser Verlag, München, 6., überarbeitete und erweiterte auflage edition, 2017. ISBN 9783446449084. http://dx.doi.org/10.3139/9783446449084.

Goel, Ashok K., Rugaber, Spencer, and Vattam, Swaroop. Structure, behavior, and function of complex systems: The structure, behavior, and function modeling language. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 23 (1):23–35, 2009. ISSN 0890-0604.

Hubka, Vladimir and Eder, W. Ernst. *Design Science: Introduction to the Needs, Scope and Organization of Engineering Design Knowledge*. Springer London, London, 1996. ISBN 9781447130918.

Jiang, Shuo, Hu, Jie, and Luo, Jianxi. Data-Driven Design-by-Analogy: State of the Art. *ASME 2021 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2021.

Khanolkar, Pranav Milind, Vrolijk, Ademir, and Olechowski, Alison. Mapping artificial intelligence-based methods to engineering design stages: a focused literature review. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 37, 2023. ISSN 0890-0604. http://dx.doi.org/10.1017/S0890060423000203.

Kitamura, Yoshinobu and Mizoguchi, Riichiro. Ontology-based description of functional design knowledge and its use in a functional way server. *Expert Systems with Applications*, 24 (2):153–166, 2003. ISSN 0957-4174.

Koller, Rudolf and Kastrup, Norbert. *Prinziplösungen zur Konstruktion technischer Produkte*. Springer eBook Collection Computer Science and Engineering. Springer Berlin Heidelberg, Berlin, Heidelberg and s.l., 2., neubearbeitete auflage edition, 1998. ISBN 9783642587559.

Pearl, Judea. *Heuristics: Intelligent Search Strategies for computer problem solving*. Addison-Wesley Publishing Company, Reading, Massachusetts and Menlo Park, California and London and Amsterdam and Don Mills, Ontario and Sydney, 1984. ISBN 0-201-05594-5.

Rosenthal, Philipp and Niggemann, Oliver. Problem examination for AI methods in conceptual product design. In *IJCAI 2021 Workshop – AI and Product Design*, Montreal, Canada, 2021.

Roth, Karlheinz. *Konstruieren mit Konstruktionskatalogen: Band 1: Konstruktionslehre*. Springer, Berlin, 3. auflage, erweitert und neu gestaltet edition, 2000. ISBN 9783642174667. http://dx.doi.org/10.1007/978-3-642-17466-7.

Roth, Karlheinz. *Konstruieren mit Konstruktionskatalogen: Band 2: Kataloge*. Springer eBook Collection Computer Science and Engineering. Springer Berlin Heidelberg, Berlin, Heidelberg and s.l., 3. auflage, mit wesentlichen ergänzungen edition, 2001. ISBN 9783642174674.

Russell, Stuart J. and Norvig, Peter. *Artificial intelligence: A modern approach*. Pearson Series in Artificial Intelligence. Pearson, Hoboken, NJ, fourth edition edition, 2021. ISBN 9780134610993.

Verein Deutscher Ingenieure. VDI-Richtlinie 2222 Blatt 1: Konstruktionsmethodik Methodisches Entwickeln von Lösungsprinzipien, 1997.

Verein Deutscher Ingenieure. VDI-Richtlinie 2221 Blatt 2: Entwicklung technischer Produkte und Systeme Gestaltung individueller Produktentwicklungsprozesse, 2019a.

Verein Deutscher Ingenieure. VDI-Richtlinie 2221 Blatt 1: Entwicklung technischer Produkte und Systeme Modell der Produktentwicklung, 2019b.

https://doi.org/10.1017/pds.2025.10194 Published online by Cambridge University Press