Proceedings of the Design Society, Volume 6: DESIGN 2026
https://doi.org/10.1017/pds.2026.10416

# Compatibility-optimized selection of solution principles using mixed-integer linear programming

**Meno-Said Haddad and Arthur Seibel**✉

*Leuphana University Lüneburg, Germany*

✉ arthur.seibel@leuphana.de

**ABSTRACT:** Conceptual design methods rarely optimize both requirement fit and cross-principle compatibility, leaving a gap in generating coherent early-stage solutions. Here, we introduce a mixed-integer linear programming formulation that selects one solution principle per function while jointly minimizing local requirement mismatch and system-level incompatibility. Using a small case study, we show how a trade-off parameter controls the balance between functional quality and integration robustness. The results demonstrate that the approach enables transparent, compatibility-aware conceptual synthesis.

**KEYWORDS:** conceptual design, functional modelling, solution principles, mixed-integer linear programming

## 1. Introduction

The morphological approach to engineering design promotes systematic exploration of the complete solution space by combining alternative means for each defining function of a system. While this structured combinatorial reasoning encourages innovation and reduces cognitive bias, it also generates a fundamental challenge: as the number of functions and alternatives increases, the number of possible configurations grows exponentially.

Motte and Bjärnemo (2013) demonstrate that even moderately sized morphological matrices rapidly become unmanageable in manual engineering design contexts. Common heuristics such as eliminating unsuitable alternatives or applying compatibility matrices provide only limited reduction. Rather than eliminating combinatorial explosion, designers must strategically navigate large discrete design spaces while accounting for incompatibilities between solution principles. Almefelt (2005) addresses this challenge through semi-quantitative synergy analysis across subsystems, emphasizing balanced system performance over isolated component optimization. More recently, Martinsson Bonde et al. (2025) introduced a computational approach that represents morphological matrices as graphs and applies a shortest-path algorithm to identify optimal configurations while incorporating incompatibilities and flexibility considerations.

Parallel advances in computational conceptual design have further formalized parts of this reasoning process. Rosenthal et al. (2025) formulate solution principle selection as a search problem and use an A* algorithm to identify minimal sets of principles covering a functional structure. Complementarily, Haddad and Seibel (2025) demonstrated how functional decompositions themselves can be generated automatically using large language models combined with Monte Carlo tree search, thereby automating the derivation of structured function networks from textual descriptions. Together, these contributions move conceptual design towards algorithmic synthesis.

However, existing approaches either focus on exhaustive combinatorial exploration, heuristic reduction strategies, search-based minimality, or pathfinding algorithms. A unified optimization model that simultaneously accounts for (i) requirement-level suitability of solution principles and (ii) quantified cross-functional compatibility within a strict one-principle-per-function configuration structure remains underdeveloped.

This paper addresses that gap by reformulating solution principle selection as a mixed-integer linear programming (MILP) problem (Grossmann, 2021). Given a functional decomposition, each function is associated with a discrete set of candidate solution principles and a set of requirements. The objective is to select exactly one principle per function such that local requirement incompatibility and cross-functional incompatibility between selected principles are jointly minimized. Unlike heuristic or pathfinding approaches, the proposed formulation provides globally optimal configurations under a transparent incompatibility model.
The contributions of this work are threefold:

1. A mathematical MILP formulation that integrates requirement compatibility and cross-principle interactions in a single optimization framework.
2. A solver-based implementation guaranteeing global optimality and producing interpretable decision variables for design analysis.
3. A proof-of-concept case study illustrating how a tuneable trade-off parameter enables systematic exploration of the balance between local performance and system-level coherence in early conceptual design.

By bridging morphological reasoning, algorithmic synthesis, and quantitative optimization, the proposed method contributes to compatibility-aware computational support for large categorical configuration spaces in conceptual engineering.

## 2. Background

### 2.1. Conceptual design and solution principles

In technical product development, the conceptual design phase forms the critical link between defining what a product must achieve and determining how it will achieve it. The process begins with a clear problem definition and the creation of a requirements list, which compiles all essential specifications, constraints, and target functions for the design task (Pahl et al., 2007; VDI 2221-1, 2019; Schlattmann & Seibel, 2021). Typically organized in tabular form, this list serves as a living document that evolves throughout the project and provides a consistent reference for evaluating and selecting design concepts. It ensures that subsequent decisions during concept development remain traceable to the original technical and economic objectives.
According to VDI 2222-1 (1997), a function defines the basic transformation relationship between a system's inputs, internal processes, and outputs as it fulfils its intended purpose. Expressing these functions in concise verb–noun formulations helps avoid premature fixation on specific solutions, thereby fostering innovative thinking. The product's purpose is first described through an overall function, often represented as a black-box model that illustrates the transformation of material, energy, and information. This overall function is then decomposed into subfunctions, each representing a partial task necessary to achieve the overall goal. The resulting functional structure—expressed in tables or block diagrams—clarifies the relationships among subfunctions and provides a logical framework for exploring alternative technical realizations.
Based on this functional model, designers identify and combine physical effects to form solution principles. A solution principle, as defined by Pahl et al. (2007), is the fundamental idea that determines how a technical function is realized. It combines an appropriate physical effect with the required geometric and material characteristics to achieve the desired transformation. The development process involves identifying physical effects capable of fulfilling subfunctions, combining them into working principles, and synthesizing these into a coherent working structure that performs the overall function. The result is a set of principle solution variants that represent potential technical concepts for further evaluation.
Roth (2000) emphasizes that a solution principle is the core physical and geometric method by which a function is fulfilled, independent of its detailed embodiment. In his approach, design catalogues (Roth, 2002) serve as a structured source of proven principles that can be systematically combined to address specific subfunctions. Hubka and Eder (1996) interpret the solution principle as the organ structure—the network of effect carriers that realize the functions defined in the function structure. Their model highlights that developing a solution principle means translating functional requirements into interacting organs that together achieve the overall purpose.
Once several solution principle variants have been generated, a systematic selection and evaluation process ensures that the most suitable concept is chosen. Pahl et al. (2007) describe this as a two-step

procedure: first, an initial screening eliminates infeasible or incompatible variants based on hard criteria such as technical viability and cost; second, a detailed evaluation applies a weighted point-scoring method that assesses both technical and economic performance, akin to the cost-benefit procedure outlined in VDI 2225-1 (1997). Roth (2000) and Koller and Kastrup (1998) propose similar approaches, dividing evaluation criteria into technical value (e.g., functionality, safety, ergonomics) and economic value (e.g., material and production costs). Weighted aggregation of these criteria allows objective comparison of alternatives and supports transparent design decisions.

The conceptual design phase thus establishes a structured progression from requirements to functions and finally to solution principles. The requirements list defines what must be achieved, the functional model clarifies what must be done, and the solution principles specify how it can be realized. This hierarchical linkage ensures that design concepts remain consistent with functional needs and project goals, while maintaining flexibility for innovation. By systematically developing, evaluating, and selecting solution principles, designers create a rational foundation for embodiment and detailed design, ensuring that each decision is traceable to both technical logic and economic justification (Koller & Kastrup, 1998; Roth, 2000). In the context of this work, these established models provide the conceptual scaffolding, while the MILP formulation offers a quantitative mechanism for automating parts of the selection and evaluation process.

## 2.2. Computational methods in conceptual design synthesis

The automation of solution principle (SP) selection marks a significant progression in conceptual design research, aiming to reduce reliance on manual reasoning and improve the efficiency of mapping functions to physical solutions. Earlier conceptual design frameworks such as those developed by Chakrabarti and Bligh (1994, 1996) laid the groundwork by establishing systematic, rule-based methods for functional reasoning and synthesis. These approaches introduced recursive problem decomposition and knowledge-guided synthesis, where functions were transformed into structural solutions through iterative refinement. However, the process remained largely manual, requiring expert designers to interpret functional requirements and select appropriate SPs from existing knowledge bases.

Later developments built upon this foundation by introducing computational and knowledge-based synthesis methods. For example, Qi et al. (2015, 2018) automated functional–structural synthesis through the integrated and the hybrid principle solution synthesis (IPSS and HPSS) methods. These frameworks employed intelligent agents to chain compatible principle solutions based on input–output flow matching, while fuzzy multi-criteria decision-making techniques (Kahraman, 2008) were used to evaluate and select the most promising combinatorial solutions. By incorporating Extenics theory to resolve structural conflicts (e.g., vibration or interference), these systems demonstrated how automation could not only accelerate design generation but also improve feasibility and multidisciplinary coherence.

Building on these advances, Rosenthal et al. (2025) proposed a formalized and optimization-driven approach for automated SP selection. Their solution principle structure finder applies an A*-based search algorithm to identify minimal, non-redundant sets of SPs that completely satisfy a function structure. Using standardized Roth functions (Roth, 2000) and constraint-based combination rules, the SPSF systematically explores and ranks potential SP configurations. The approach ensures that the resulting conceptual designs are both complete and computationally efficient, offering a reproducible framework that reduces human effort while maintaining interpretability.

Meanwhile, the universal conceptual modelling (UCM) framework (Lukyanenko et al., 2024) provides a broader theoretical context for such automation. UCM emphasizes modularity, minimalism, and accessibility in conceptual modelling, principles that can guide the development of automated SP selection systems that remain understandable and adaptable for both experts and non-experts. When combined with algorithmic synthesis methods (Rosenthal et al., 2025) and knowledge-based reasoning frameworks (Chakrabarti & Bligh, 1994, 1996; Qi et al., 2018), these principles help ensure that automated systems not only generate solutions but also remain transparent and human-centred. In summary, the evolution from manual to automated SP selection reflects a gradual integration of functional reasoning, computational synthesis, and universal modelling principles.

However, most existing approaches either focus on minimizing the number of selected solution principles, clustering requirements, or rule-based synthesis; they rarely incorporate a quantitative, system-wide compatibility objective into a single optimization model. This gap motivates the MILP-based approach proposed in this paper.

# 3. Methodology

## 3.1. Problem overview and complexity

We address the problem of selecting an optimal set of categorical solution principles in early-stage conceptual design. Given a functional decomposition $F$, each function $f \in F$ is associated with a finite set of candidate solution principles $P_f$ and a set of local requirements $T_f$. The objective is to select exactly one principle per function such that both requirement satisfaction and cross-functional compatibility are jointly optimized.

Two types of incompatibility are considered:

- Local requirement incompatibility $d_{fpt} \in [0, 1]$, representing how poorly principle $p \in P_f$ satisfies requirement $t \in T_f$.
- Cross-functional incompatibility $D_{fp,gq} \in [0, 1]$, representing the incompatibility between principles $p \in P_f$ and $q \in P_g$ for $f \neq g$.

Requirement importance weights $w_{ft} \in \mathbb{R}^+$ may also be specified. In this study, all weights are set to 1. Binary decision variables are defined as Equations (1) and (2):

$$x_{fp} = \begin{cases} 1 & \text{if principle } p \text{ is selected for function } f, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

$$y_{fp,gq} = \begin{cases} 1 & \text{if both } x_{fp} = 1 \text{ and } x_{gq} = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The total incompatibility to be minimized is Equation (3)

$$\min \left( \sum_{f \in F} \sum_{p \in P_f} \left( \sum_{t \in T_f} w_{ft}\, d_{fpt} \right) x_{fp} + \beta \sum_{\substack{f,g \in F \\ f < g}} \sum_{p \in P_f} \sum_{q \in P_g} D_{fp,gq}\, y_{fp,gq} \right), \tag{3}$$

where $\beta \in [0, 1]$ is a trade-off parameter controlling the relative importance of cross-functional compatibility. The parameter $\beta$ enables systematic exploration of different conceptual design strategies. For $\beta = 0$, the model reduces to purely requirement-driven selection, where each function is optimized independently with respect to its local requirements. For increasing values of $\beta$, cross-functional compatibility progressively influences the solution, coupling the individual function selections into a global configuration problem. At $\beta = 1$, local and systemic incompatibility are weighted equally under the chosen normalization.

Because both incompatibility measures are normalized to the interval $[0, 1]$, the trade-off parameter directly governs the relative impact of subsystem integration versus local performance. This formulation therefore allows controlled investigation of architecture transitions as design priorities shift.

The following constraints ensure feasibility:

1. Exactly one principle per function Equation (4):

$$\sum_{p \in P_f} x_{fp} = 1 \ \forall f \in F. \tag{4}$$

2. Linearization of pairwise interactions Equation (5):
   For all $f < g$, $p \in P_f$, $q \in P_g$:

$$\begin{aligned} y_{fp,gq} &\leq x_{fp}, \\ y_{fp,gq} &\leq x_{gq}, \\ y_{fp,gq} &\geq x_{fp} + x_{gq} - 1. \end{aligned} \tag{5}$$

The resulting model is a mixed-integer linear program (MILP). Due to cross-functional compatibility interactions, decisions become globally coupled, and the number of feasible configurations grows exponentially as $\prod_{f \in F} |P_f|$. The problem is therefore NP-hard and requires solver-based optimization for scalable application. The proposed linear binary formulation was selected because it enables (i) guarantees of global optimality, (ii) transparent decomposition of objective contributions into local and cross-functional components, and (iii) compatibility with mature and widely available MILP solvers.

## 3.2. Implementation

The MILP formulation described in Section 3.1 is implemented using a structured model-building procedure that translates the functional decomposition and incompatibility data into a solver-ready optimization problem. The implementation follows a clear sequence of steps: initialization of sets and parameters, declaration of binary decision variables, construction of constraints, definition of the objective function, and execution of the solver.

First, the sets of functions $F$, candidate principles $P_f$, and local requirements $T_f$ are defined. Local incompatibility values $d_{fpt}$, cross-functional incompatibilities $D_{fp,gq}$, and requirement weights $w_{ft}$ are stored in structured data formats. These serve as direct inputs to the optimization model.

Binary variables $x_{fp}$ are then introduced to represent the selection of a specific principle for a given function. Auxiliary binary variables $y_{fp,gq}$ are defined to capture pairwise compatibility relationships between principles of different functions. A constraint enforcing exactly one selected principle per function is added, followed by linearization constraints that activate pairwise incompatibility contributions only when both associated principles are selected.

The overall implementation procedure is summarized in Figure 1.

**Algorithm 1** MILP-Based Compatibility Optimization

**Require:**

$F$: Set of functions
$P_f$: Candidate principles for each $f \in F$
$T_f$: Requirements for each $f \in F$
$w_{ft}$: Requirement importance weight
$d_{fpt}$: Local incompatibility between principle $p$ and requirement $t$
$D_{fp,gq}$: Cross-principle incompatibility
$\beta$: Trade-off parameter

**Ensure:**

Optimal set of principles, one per function

Initialize MILP model using Pyomo
**for** each $f \in F$ **do**
    Define set of candidate principles $P_f$
    Define local requirements $T_f$
**end for**
Define binary variables $x_{fp}$ for selecting principle $p$ for function $f$
Define auxiliary binary variables $y_{fp,gq}$ for compatibility pairs
**for** each $f \in F$ **do**
    Add constraint: $\sum_{p \in P_f} x_{fp} = 1$
**end for**
**for** each $f < g$, $p \in P_f$, $q \in P_g$ **do**
    Add linearization constraints:
        $y_{fp,gq} \leq x_{fp}$
        $y_{fp,gq} \leq x_{gq}$
        $y_{fp,gq} \geq x_{fp} + x_{gq} - 1$
**end for**
Define objective:
    $\min \sum_{f,p,t} w_{ft} \cdot d_{fpt} \cdot x_{fp} + \beta \sum_{f<g,p,q} D_{fp,gq} \cdot y_{fp,gq}$
Solve MILP using SCIP solver
Return selected principles where $x_{fp} = 1$

**Figure 1.** Pseudocode of MILP compatibility optimization

After assembling the model, the objective function is constructed as the weighted sum of local and cross-functional incompatibilities. The resulting MILP is solved using a standard optimization solver. The solution consists of one selected principle per function, together with the associated objective value and its decomposition into local and pairwise contributions.

Because the decision variables correspond directly to conceptual design choices, the output is fully interpretable: designers can identify which principles are selected, which incompatibility contributions dominate the objective, and how configuration decisions change when adjusting the trade-off parameter. This interpretability is essential for embedding the approach into early-stage conceptual design workflows.

### 3.3. Experimental setup

To evaluate the proposed compatibility-based configuration model, a simplified waffle maker system is used as a proof-of-concept case. The system consists of four functional blocks: Store_Electricity, Store_UserSettings, Store_Batter, and Guide_ElectricityToHeatingElement. For each function, two alternative solution principles are defined, resulting in a discrete configuration space of $2^4 = 16$ possible combinations. The candidate principles assigned to each function are summarized in Table 1.

**Table 1.** Function—principle mapping

| Function | Candidate Principles |
|---|---|
| Store_Electricity | Battery, Capacitor |
| Store_UserSettings | EEPROM, FlashMemory |
| Store_Batter | Tank, Container |
| Guide_ElectricityToHeatingElement | Wire, PCB_Trace |

Each function is associated with two technical requirements that characterize the desired performance of the selected principle. These requirements represent simplified engineering targets such as capacity, insulation, or flexibility. For clarity and generality, all requirement importance weights are set to $w_{ft} = 1$, allowing the structural behaviour of the optimization model to be examined without additional weighting effects. The functional requirements considered in this study are listed in Table 2.

**Table 2.** Functional requirements

| Function | Candidate Principles |
|---|---|
| Store_Electricity | capacity, safety |
| Store_UserSettings | retention, speed |
| Store_Batter | insulation, cleanability |
| Guide_ElectricityToHeatingElement | resistance, flexibility |

For every function–principle–requirement combination, a normalized local incompatibility score $d_{fpt} \in [0, 1]$ is defined. Lower values indicate better alignment with the corresponding requirement. These values quantify how well a candidate principle satisfies its associated functional targets.

For example, Battery performs moderately on capacity (0.32) but well on safety (0.15), whereas FlashMemory performs very well on retention (0.06) but less well on speed (0.51). The complete set of local incompatibility scores used in the experiment is presented in Table 3.

In addition to local requirement satisfaction, cross-functional compatibility between principles is explicitly modelled. For each valid pair of principles belonging to different functions, a normalized incompatibility value $D_{fp,gq} \in [0, 1]$ is defined. These values represent potential integration conflicts such as thermal interaction, spatial interference, or system-level integration challenges. For instance, Battery and Tank exhibit strong incompatibility (0.90), while EEPROM and Container are fully compatible (0.00). The complete cross-functional incompatibility matrix is shown in Table 4.

Although the case study is intentionally small, it retains the essential combinatorial structure of conceptual configuration problems. Even with only four functions and two alternatives per function, the configuration space already contains 16 feasible combinations exhibiting competing trade-offs between local requirement satisfaction and cross-functional compatibility.

To analyse this trade-off systematically, the optimization model is evaluated for $\beta \in \{0.0, 0.1, \ldots, 1.0\}$. For each value of $\beta$, the globally optimal configuration and its decomposition into local and pairwise incompatibility contributions are recorded. This enables clear visualization of architecture transitions as compatibility importance increases.

**Table 3.** Incompatibility scores

| Function | Principle | Requirement | Score |
|---|---|---|---|
| Store_Electricity | Battery | capacity | 0.32 |
| | | safety | 0.15 |
| | Capacitor | capacity | 0.65 |
| | | safety | 0.07 |
| Store_UserSettings | EEPROM | retention | 0.54 |
| | | speed | 0.37 |
| | FlashMemory | retention | 0.06 |
| | | speed | 0.51 |
| Store_Batter | Tank | insulation | 0.04 |
| | | cleanability | 0.43 |
| | Container | insulation | 0.58 |
| | | cleanability | 0.25 |
| Guide_Electricity | Wire | resistance | 0.12 |
| ToHeatingElement | | flexibility | 0.48 |
| | PCB_Trace | resistance | 0.33 |
| | | flexibility | 0.16 |

**Table 4.** Cross-functional incompatibility

| Pair ($f$, $p$, $g$, $q$) | Score |
|---|---|
| Store_Electricity-Battery → Store_UserSettings-EEPROM | 0.05 |
| Store_Electricity-Battery → Store_UserSettings-FlashMemory | 0.09 |
| Store_Electricity-Capacitor → Store_UserSettings-EEPROM | 0.41 |
| Store_Electricity-Capacitor → Store_UserSettings-FlashMemory | 0.11 |
| Store_Electricity-Battery → Store_Batter-Tank | 0.90 |
| Store_Electricity-Battery → Store_Batter-Container | 0.04 |
| Store_Electricity-Capacitor → Store_Batter-Tank | 0.54 |
| Store_Electricity-Capacitor → Store_Batter-Container | 0.33 |
| Store_Electricity-Battery → Guide_ElectricityToHeatingElement-Wire | 0.85 |
| Store_Electricity-Battery → Guide_ElectricityToHeatingElement-PCB_Trace | 0.16 |
| Store_Electricity-Capacitor → Guide_ElectricityToHeatingElement-Wire | 0.34 |
| Store_Electricity-Capacitor → Guide_ElectricityToHeatingElement-PCB_Trace | 0.33 |
| Store_UserSettings-EEPROM → Store_Batter-Tank | 0.25 |
| Store_UserSettings-EEPROM → Store_Batter-Container | 0.00 |
| Store_UserSettings-FlashMemory → Store_Batter-Tank | 0.44 |
| Store_UserSettings-FlashMemory → Store_Batter-Container | 0.09 |
| Store_UserSettings-EEPROM → Guide_ElectricityToHeatingElement-Wire | 0.60 |
| Store_UserSettings-EEPROM → Guide_ElectricityToHeatingElement-PCB_Trace | 0.07 |
| Store_UserSettings-FlashMemory → Guide_ElectricityToHeatingElement-Wire | 0.32 |
| Store_UserSettings-FlashMemory → Guide_ElectricityToHeatingElement-PCB_Trace | 0.45 |
| Store_Batter-Tank → Guide_ElectricityToHeatingElement-Wire | 0.91 |
| Store_Batter-Tank → Guide_ElectricityToHeatingElement-PCB_Trace | 0.09 |
| Store_Batter-Container → Guide_ElectricityToHeatingElement-Wire | 0.14 |
| Store_Batter-Container → Guide_ElectricityToHeatingElement-PCB_Trace | 0.79 |

## 4. Results and discussion

To investigate the behaviour of the MILP-based configuration model, the optimization problem was solved for trade-off parameters $\beta \in \{0.0,\ 0.1,\ \ldots,\ 1.0\}$.

The parameter $\beta$ balances the influence of local requirement satisfaction and cross-functional compatibility. Lower values emphasize alignment with individual functional requirements, whereas higher values prioritize systemic integration robustness.

Table 5 summarizes the globally optimal configuration for each value of $\beta$, together with the decomposition of the objective into local and cross-functional contributions.

**Table 5.** Optimal configurations for varying trade-off parameter $\beta$

| $\beta$ | Store_Electricity | Store_UserSettings | Store_Batter | Guide_Electricity | Local | Pair | Objective |
|---|---|---|---|---|---|---|---|
| 0.0 | Battery | FlashMemory | Tank | PCB_Trace | 2.00 | 2.13 | 2.000 |
| 0.1 | Battery | FlashMemory | Tank | PCB_Trace | 2.00 | 2.13 | 2.213 |
| 0.2 | Battery | FlashMemory | Tank | PCB_Trace | 2.00 | 2.13 | 2.426 |
| 0.3 | Battery | FlashMemory | Tank | PCB_Trace | 2.00 | 2.13 | 2.639 |
| 0.4 | Battery | FlashMemory | Tank | PCB_Trace | 2.00 | 2.13 | 2.852 |
| 0.5 | Battery | FlashMemory | Tank | PCB_Trace | 2.00 | 2.13 | 3.065 |
| 0.6 | Battery | EEPROM | Tank | PCB_Trace | 2.34 | 1.52 | 3.252 |
| 0.7 | Battery | EEPROM | Tank | PCB_Trace | 2.34 | 1.52 | 3.404 |
| 0.8 | Battery | EEPROM | Tank | PCB_Trace | 2.34 | 1.52 | 3.556 |
| 0.9 | Battery | EEPROM | Container | PCB_Trace | 2.70 | 1.11 | 3.699 |
| 1.0 | Battery | EEPROM | Container | PCB_Trace | 2.70 | 1.11 | 3.810 |

Three distinct architectural regions emerge as $\beta$ increases. For $\beta \leq 0.5$, the configuration Battery / FlashMemory / Tank / PCB_Trace is selected, minimizing local incompatibility (2.00) but exhibiting high cross-functional conflict (2.13). For $0.6 \leq \beta \leq 0.8$, the model shifts to Battery / EEPROM / Tank / PCB_Trace, accepting a moderate increase in local mismatch (2.34) to reduce pairwise incompatibility (1.52). For $\beta \geq 0.9$, the configuration Battery / EEPROM / Container / PCB_Trace further improves compatibility (1.11) at the expense of higher local incompatibility (2.70). These transitions illustrate how explicit compatibility modelling can alter conceptual architectures and how the trade-off parameter functions as a steering mechanism in early-stage design.

Solver runtime experiments with 10 functions show a strong sensitivity to the number of candidate principles per function. When three principles per function are considered, solution times remain in the range of approximately 3–5 seconds, largely independent of the number of requirements. Increasing the alternatives to five principles raises runtimes substantially, with observed values between roughly 17 and 58 seconds. For ten principles per function, computational effort increases dramatically, reaching several hundred to over 2000 seconds in the tested cases. These results reflect the exponential growth of the configuration space, given by $\prod |P_f|$, and confirm the combinatorial nature of compatibility-based configuration. While small cases remain tractable, larger instances quickly become computationally demanding, underscoring the NP-hard character of the problem and justifying the use of structured MILP solvers rather than exhaustive enumeration.

## 5. Conclusion and future work

This work presented a mixed-integer linear programming approach for supporting conceptual design through the automated selection of functionally compatible solution principles. Using a structured waffle maker case study, we demonstrated how the model simultaneously accounts for local requirement satisfaction and cross-functional compatibility within a unified optimization framework. By explicitly quantifying both local and pairwise incompatibility contributions, the approach provides transparent and interpretable configuration decisions that can be traced back to underlying compatibility data.

The systematic evaluation of trade-off parameters $\beta \in [0, 1]$ revealed three distinct architectural regions, illustrating how compatibility modelling can qualitatively alter the selected conceptual architecture.

The results show that even small changes in the relative importance of systemic compatibility can trigger configuration transitions. The trade-off parameter $\beta$ therefore functions as a strategic design lever, enabling controlled exploration of requirement-driven versus integration-driven conceptual solutions in early-stage design.

Beyond architectural transitions, scalability experiments confirmed the exponential growth of the configuration space with increasing numbers of principles per function. Solver runtimes increased significantly for larger instances, reflecting the NP-hard character of compatibility-based configuration problems. These findings underline the necessity of structured optimization methods rather than exhaustive enumeration when addressing realistic design spaces.

The presented case study serves as a proof of concept. The effectiveness of the method depends on the availability and quality of incompatibility data, including requirement weights and cross-functional interaction metrics, which may require structured expert elicitation in industrial contexts. Furthermore, while the waffle maker example enables clear interpretation of model behaviour, larger industrial systems will introduce additional modelling and computational challenges.

Future research should address scalability through decomposition strategies, heuristics, or hybrid optimization approaches. Incorporating uncertainty or learning-based estimation of incompatibility values could further enhance realism. Another promising direction is uncertainty modelling. In practice, incompatibility assessments are rarely deterministic. Stochastic formulations or fuzzy representations could capture uncertainty in compatibility evaluations while maintaining the structure of the optimization model. In addition, tighter integration with conceptual design environments, such as catalogue-based systems and automated function-structure generation methods (Haddad & Seibel, 2025), would support embedding the approach into practical engineering workflows.

In summary, the proposed MILP-based framework provides a structured and explainable decision-support mechanism for compatibility-aware conceptual configuration. It contributes to the growing integration of formal optimization methods into early-stage engineering design, offering a complementary tool that augments, rather than replaces, human design reasoning.

## Generative AI statement

This manuscript was partially revised using generative artificial intelligence (ChatGPT 5.1 by OpenAI). AI support was used exclusively for language editing and structural refinement. All conceptual contributions, technical content, mathematical formulations, and interpretations of results were created by the authors.

## References


Achterberg, T. (2009). SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1), 1–41. https://doi.org/10.1007/s12532-008-0001-1

Almefelt, L. (2005). Balancing properties while synthesising a product concept—A method highlighting synergies. In *DS 35: Proceedings of ICED 05, the 15th International Conference on Engineering Design, Melbourne, Australia*, 15.–18.08.2005

Balas, E., Chvátal, V., & Nešetřil, J. (1987). On the maximum weight clique problem. *Mathematics of Operations Research*, 12(3), 522–535. https://doi.org/10.1287/moor.12.3.522

Chakrabarti, A., & Bligh, T. P. (1994). 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. https://doi.org/10.1007/BF01607275

Chakrabarti, A., & Bligh, T. P. (1996). An approach to functional synthesis of solutions in mechanical conceptual design. Part II: Kind synthesis. *Research in Engineering Design*, 8(1), 52–62. https://doi.org/10.1007/BF01616556

Green, E., Estrada, S., Gopalakrishnan, P. K., Jahanbekam, S., & Behdad, S. (2022). A graph partitioning technique to optimize the physical integration of functional requirements for axiomatic design. *Journal of Mechanical Design*, 144(5). https://doi.org/10.1115/1.4052702

Grossmann, I. E. (2021). *Advanced Optimization for Process Systems Engineering*. Cambridge University Press. https://doi.org/10.1017/9781108917834

Haddad, M.-S., & Seibel, A. (2025). Functional decomposition of technical products based on large language models and Monte Carlo tree search. *Proceedings of the Design Society*, 5, 1913–1922. https://doi.org/10.1017/pds.2025.10205

Hart, W. E., *et al.* (2017). *Pyomo: Optimization Modeling in Python* (Vol. 67). Springer. https://doi.org/10.1007/978-3-319-58821-6

Hubka, V., & Eder, W. E. (1996). *Design Science: Introduction to the Needs, Scope and Organization of Engineering Design Knowledge*. Springer.

Kahraman, C. (Ed.). (2008). *Fuzzy Multi-Criteria Decision Making* (Vol. 16). Springer. https://doi.org/10.1007/978-0-387-76813-7
Koller, R., & Kastrup, N. (1998). *Prinziplösungen zur Konstruktion technischer Produkte*. Springer. https://doi.org/10.1007/978-3-642-58755-9
Lawler, E. L. (1963). The quadratic assignment problem. *Management Science*, 9(4), 586–599. https://doi.org/10.1287/mnsc.9.4.586
Lukyanenko, R., Samuel, B. M., Parsons, J., Storey, V. C., Pastor, O., & Jabbari, A. (2024). Universal conceptual modeling: Principles, benefits, and an agenda for conceptual modeling research. *Software & Systems Modeling*, 23(5), 1077–1100. https://doi.org/10.1007/s10270-024-01207-8
Martinsson Bonde, J., Alonso Fernández, I., Kokkolaras, M., Malmqvist, J., Panarotto, M., & Isaksson, O. (2025). Managing combinatorial design challenges using flexibility and pathfinding algorithms. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 39*, e16*. https://doi.org/10.1017/S0890060425100048
Motte, D., & Bjärnemo, R. (2013). Dealing with the combinatorial explosion of the morphological matrix in a "manual engineering design" context. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (Vol. 55928, p. V005T06A014). American Society of Mechanical Engineers. https://doi.org/10.1115/DETC2013-12040
Pahl, G., Beitz, W., Feldhusen, J., & Grote, K.-H. (2007). *Engineering Design: A Systematic Approach* (3rd ed.). Springer. https://doi.org/10.1007/978-1-84628-319-2
Qi, J., Hu, J., Zhu, G., & Peng, Y. (2015). Automatically synthesizing principle solutions in multi-disciplinary conceptual design with functional and structural knowledge. In *Proceedings of the ASME 2015 Design Automation Conference*. https://doi.org/10.1115/DETC2015-46373
Qi, J., Hu, J., & Peng, Y.-H. (2018). An integrated principle solution synthesis method in multi-disciplinary mechatronic product conceptual design. *Concurrent Engineering*, 26(4), 341–354. https://doi.org/10.1177/1063293X18799488
Rosenthal, P., Liebert, A., & Niggemann, O. (2025). Finding optimal solution principles in conceptual design. *Proceedings of the Design Society*, 5, 1803–1812. https://doi.org/10.1017/pds.2025.10194
Roth, K. (2000). *Konstruieren mit Konstruktionskatalogen – Vol. I: Konstruktionslehre* (3rd ed.). Springer. https://doi.org/10.1007/978-3-642-17466-7
Roth, K. (2002). Design catalogues and their usage. In Chakrabarti, A. (ed) *Engineering Design Synthesis* (pp. 121–129). Springer. https://doi.org/10.1007/978-1-4471-3717-7_8
Schlattmann, J., & Seibel, A. (2021). *Structure and Organization of Product Development Projects*. Springer. https://doi.org/10.1007/978-3-030-81046-7
VDI 2221 Part 1. (2019). *Design of Technical Products and Systems—Model of Product Design*. Beuth.
VDI 2222 Part 1. (1997a). *Design Engineering Methodics—Methodic Development of Solution Principles* (in German). Beuth.
VDI 2225 Part 1. (1997b). *Design Engineering Methodics—Engineering Design at Optimum Cost—Simplified Calculation of Costs* (in German). Beuth.