Graph-based product prototype optimization using multi-objective reinforcement learning: A framework concept

Sven Münker [1]*, Hans Aoyang Zhou[1], Anas Abdelrazeq[1], Julian Haller[1] and Robert H. Schmitt[1]

[1]RWTH Aachen University, Chair of Intelligence in Quality Sensing, 52074 Aachen

*Corresponding author. E-Mail address: sven.muenker@wzl-iqs.rwth-aachen.de; Tel.:0241/80 25843

Abstract: Circular-economy targets and emerging regulatory constraints increase the need to assess manufacturability and end-of-life disassembly already in early design, where late discovery of feasibility issues can trigger costly redesign loops. For this purpose, we represent product designs as AND/OR assembly graphs, that encode feasible assembly alternatives. We ask whether constraining product-level edits on an AND/OR assembly graph and evaluation in a production simulation can target this issue. To answer the question, this paper specifies a framework concept for optimizing product configurations with respect to assembly and disassembly.Design edits are restricted to graph-valid operators and further pruned by a manufacturability knowledge graph that encodes admissible substitutions and hard constraints. Candidate designs are evaluated by a production simulation across assembly and disassembly scenarios, returning a vector of performance indicators. Training uses scalarization of this vector, while full vectors are retained for post-hoc trade-off analysis. We also describe a minimum viable prototype that instantiates a restricted subset of the concept: binary joining-method selection on editable operations with a scheduling-based evaluator and a reinforcement-learning loop.

Keywords:
AND/OR graphs, design for disassembly, multi-objective reinforcement learning

1 Introduction

Sustainability targets and regulatory constraints increase the need to consider manufacturability and circularity during early design stages [1]. Decisions made constrain assembly and disassembly options downstream and can induce expensive iteration cycles especially when issues are discovered late [2]. Increasing product complexity, growing numbers of feasible assembly alternatives, and the higher routing flexibility in matrix assembly systems increase the coupling between design and production decisions. As a result, feasibility and performance issues emerge from interactions between product structure, joining choices, and resource constraints and are difficult to anticipate without explicit models [3]. Model-based evaluation can reduce iteration by enabling earlier assessment of manufacturability and (dis-)assembly consequences, but it requires both a representation that supports structural edits and alternatives without violating the product's requirements and an evaluation method that quantifies production-level impacts.

Feasible product-level changes such as joint selection, part substitution, and alternative assembly structures are combinatorial. Unconstrained exploration wastes computation on infeasible candidates and makes systematic trade-off analysis difficult when objectives conflict (e.g., assembly time vs. planning flexibility). Because each design edit changes the feasible action set and the resulting production performance can only be assessed through expansive simulation, the search problem becomes a sequential decision problem under black-box evaluation.

A framework for optimizing early-stage product variants under production-level evaluation must fulfill the following requirements:

- Represent alternative product structures compactly and expose well-defined editable sites
- Apply validity preserving edit operations that maintain structural consistency and enforce hard disassembly-feasibility constraints
- Model disassembly explicitly as a feasibility- and method-dependent process
- Multi-objective handling that is able to capture the trade-off between assembly and disassembly

Our hypothesis is that reinforcement learning (RL) can solve this problem, by learning edit policies that adapt exploration to observed outcomes and improve sample-efficiency relative to unguided heuristics. This motivates the following research question:

Research Question: *Does constraining product-level design edits on an AND/OR assembly graph and evaluating candidates in a multi-objective production simulation enable a reinforcement-learning agent to find manufacturable designs more efficiently and with better trade-offs than heuristic baselines?*

This paper contributes (1) a framework concept with explicit modules, interfaces, and validity-preserving edit operators; (2) a test protocol including a toy use-case and baseline heuristics; and (3) a prototype instantiation that implements a restricted subset of the framework to validate key integration points (representation, action encoding, and evaluation loop) and identify engineering constraints for scaling. Methodologically, the work follows a design-science where the artifact (framework) is defined from requirements and validated through a computational experiment.

In Section 2, the state of the art of decision frameworks in the context of assembly graphs, joining methods and optimization over constrained design spaces is illustrated. In Section 3, the framework concept including product representation, product constraints, change operators, assembly evaluation and optimization loop is presented. In Section 4, a toy use-case training dataset and a minimum viable prototype (MVP) of the framework is presented. In Section 5, the content of the paper is summarized and an outlook for future research is given.

2 State of the Art

In the following section, we present a short overview of data-driven systems in the context of product design and assembly and disassembly, illustrating frameworks and components that are relevant to the proposed framework of this paper. We close the section by highlighting the current research gap that this paper aims to fill.

Assembly sequence planning has long used explicit and implicit plan representations to capture precedence constraints and alternative subassemblies. AND/OR assembly graphs (AOG) were introduced as a compact representation of families of assembly plans and have been used to represent alternative assembly decompositions and sequences without enumerating a single fixed plan [4,5]. In manufacturing and remanufacturing contexts, AOGs are also used to represent feasible (dis)assembly sequences derived from geometric and technical precedence constraints, supporting alternative subassemblies and parallelizable operations [6]. Beyond precedence-only modeling, recent work targets physically feasible sequence planning from CAD (Computer-Aided Design) -derived geometry, incorporating collision-free motion and gravitational stability constraints. ASAP (Automated Sequence Planning for Complex Robotic Assembly with Physical Feasibility) is a physics-based planner that generates physically feasible assembly sequences for complex multi-part assemblies and demonstrates performance across a large dataset of product assemblies [7]. While graph representations have been

established as a product representation in the context of assembly planning, they are rarely used in early design stages to improve the product quality regarding assembly and disassembly.

Design for Disassembly (DfD) in circular-economy contexts emphasizes that disassembly time is not a simple inverse of assembly time and is driven by connector/join choice and accessibility. For end-of-life scenarios it can further depend on product condition. Accordingly, DfD methods use systematic time- or effort-based metrics and target-oriented disassembly planning, favoring non-destructive, reversible connections, where feasible [8,9]. Empirical studies show that disassembly time and reversibility depend strongly on the joining method, motivating simplified but method-dependent models (e.g. reversible fasteners vs. non-reversible bonds) for early-stage assessment [10]. This line of work supports modelling disassembly as target-state attainment and treating undoability as a joining-method-dependent feasibility predicate rather than assuming universal reversibility of every joining operation.

In process planning, manufacturing process knowledge (e.g. tooling and fixturing capabilities, resource models) is commonly represented using structured rule bases and ontology/graph-based formalisms. Recent surveys emphasize knowledge-graph-based approaches for representing and operationalizing process knowledge, covering extraction, graph construction, validation, and knowledge graph -driven process generation with planning pipelines [11]. Complementary work uses ontology-based manufacturability analysis to encode capability and constraint models that map design features to feasible manufacturing operations [12]. Within assembly contexts, knowledge-graph approaches have also been proposed to organize assembly resources and link product, process, and resource knowledge for retrieval and decision support [13]. These directions motivate a framework interface in which admissible substitutions and hard constraints are provided by a dedicated knowledge module, independent of the optimization method.

Optimizing product or process choices when evaluation requires simulation or scheduling is commonly framed as simulation optimization, where objective and constraints are observable only through computational evaluation. Surveys cover algorithmic families and application patterns, including settings with expensive evaluation and multiple outputs [14]. When precedence structure includes AND/OR conditions, dedicated scheduling formulations exist to capture alternative precedence requirements, providing a basis for integrating product graph structure into production-level evaluation [15, 16].

For multi-objective decision-making in sequential settings, multi-objective reinforcement learning (MORL) surveys formalize vector-valued objectives, scalarization choices, and solution concepts beyond single-objective policies, motivating the use of scalarization for training while retaining vectors for explicit decision support across objectives [17]. In constrained action spaces, invalid action masking is a practical mechanism to restrict exploration to admissible actions. Analyses show that masking can be important when invalid actions are prevalent and formalize implications for policy-gradient learning [18].

Existing work provides (i) graph representations that compactly encode alternative (dis)assembly sequences, (ii) DfD insights that disassembly feasibility and effort are method-dependent and target-state oriented, (iii) structured knowledge representations for manufacturability constraints and admissible process choices, and (iv) optimization paradigms that combine simulation/scheduling evaluation with multi-objective sequential decision-making. However, these elements are rarely integrated into a modular loop that (a) restricts design edits to graph-valid operator instances inferred from a product graph, (b) enforces manufacturability and disassembly feasibility through an explicit knowledge interface, and (c) evaluates design candidates as scenario-indexed makespan vectors to expose assembly-disassembly trade-offs under a shared production model.

3 Framework Concept

In this chapter, we illustrate the concept of the framework to improve the product's assemblability and disassemblability in the early design stage. The framework consists of three modules: The *knowledge module* that encodes manufacturability knowledge and is used to obtain alternative joining operations for the product, the *evaluator module*, where the product and all candidate alternatives are evaluated in multiple scenarios and the *optimizer module*, where changes on the product are being performed, based on the results from the evaluator module. The product is represented by an AOG over its components and joining operations. This graph is optimized in the following way (Figure 1):

Start by inserting the initial product AOG G_0 into the evaluator module and set it as the current AOG G_t (the subscript t denotes the t-th iteration of evaluation and optimization) (0). The initial product AOG will be evaluated once, and its results will be stored as the baseline. Using the disassembly AOG generator, create the disassembly AOG G_t^{-1}(1). Create a set of assembly and disassembly scenarios S in the scenario generator (2). Simulate the set of scenarios in the production simulation yielding a vector of simulation results $\boldsymbol{y}(G_t, S)$ (e.g., makespans) (3). Aggregate the simulation results into a scalar reward $r(G_t)$ (4). In the enhanced AOG generator, create the enhanced AOG G_t^e using the alternatives encoded in the knowledge module (5). Pass the simulation result vector, the reward and the enhanced AOG to the optimizer module (6). Extract the available actions from the enhanced AOG using the action generator (7). Use an optimizer algorithm to select the next action given the reward and vector of simulation results (8). Apply the chosen action to create the (next) candidate AOG (9). Pass the candidate AOG to the evaluator module for evaluation where it is set as the current AOG (10). Repeat steps (1)-(10) until the edit budget is exhausted or optimality is reached. Output the final product AOG G_f (11).

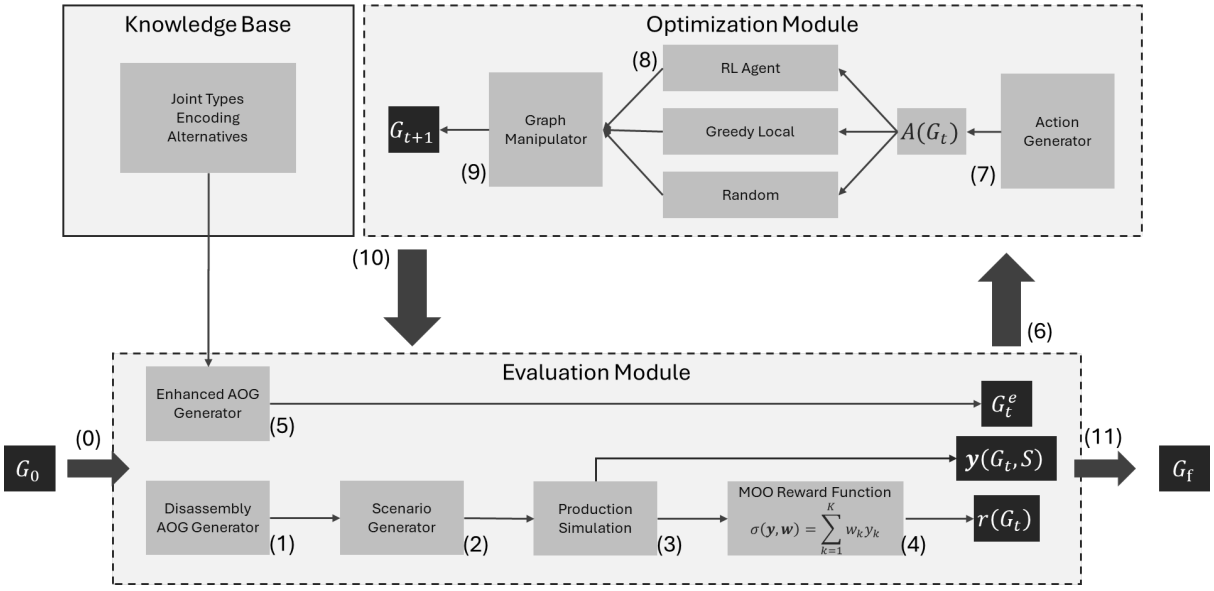

Figure 1: Conceptualization of the framework. G_t is the AOG after step t, G_t^{-1} is the disassembly AOG to G_t, G_t^e is the enhanced AOG to G_t, $\boldsymbol{y}(G_t, S)$ is the vector of performance indicators for the set of scenarios S and the AOG G_t and $r(G_t)$ is the reward function computed from $\boldsymbol{y}(G_t, S)$

In the following sections, the different components of the framework are discussed in detail. We start by introducing AOGs as the main data structure to represent the product. Afterwards, we describe the purpose and possible structure of the knowledge module, before illustrating the different components of the evaluator module. We close this chapter with the discussion of the optimization module and its position within an RL framework, while describing the two naïve baselines that will later be used for comparison.

3.1 Product Representation: AND/OR Assembly Graphs

The product is encoded as an AOG assembly graph G. AND relations encode mandatory composition and precedence relations, while OR relations encode alternative realizations at defined choice points. Node and edge types cover at least: parts, joints and operations. As an extension to the usual AOG definition, edges corresponding to joints are marked as either editable or uneditable. For further clarification, we use the term joint to refer to an edge in the AOG, while the term joining operation refers to the underlying operation. Editable joints might be changed during the optimization process, while uneditable joints remain fixed. A candidate design is defined by changing the joining-operation behind at least one joint.

The AOG is treated as the primary structural representation. Disassembly is not assumed to be a mere reversal of assembly. Instead, disassembly feasibility and required disassembly actions depend on selected joining processes (e.g. reversible vs. destructive operations). Therefore, disassembly-relevant structure is derived from the AOG together with joining-method selections and feasibility predicates. This avoids the implication that every joining operation has an inverse disassembly operation.

The representation must support two operations: (i) extraction of editable sites (joints where the underlying joining operation might be changed) and (ii) application of edits that preserve the feasibility of the corresponding disassembly. The AOG is not required to fully enumerate assembly sequences. It encodes enough structure to ensure edits remain consistent with the product's assembly logic and target part removal constraints.

3.2 Manufacturability Knowledge Module

To obtain alternatives for the editable joints, external knowledge on alternative joining methods is needed. The manufacturability knowledge module provides the rules and data required to infer admissible alternatives at a given editable site. It can be implemented as a knowledge graph or a structured knowledge base with predicates, depending on the complexity of the encoded information.

The module must encode the alternative definitions of joining operations, including process families and admissible substitutions per site type and compatibility constraints linking joining processes and part properties (e.g. materials, geometry classes). The alternative joining operations $j_1, \dots, j_k$ are assessed through the alternatives function

$$alts(G, l) \rightarrow \{j_1, \dots, j_k\}$$

That returns admissible alternatives at site l for AOG G. In this concept, operator instantiation uses $alts$ to construct the constrained action space.

3.3 Evaluator Module

The evaluator module has two main functions: On the one hand, it evaluates AOG in the context of its assemblability and disassemblability for a given production system. On the other hand, it is used to enhance the AOG with the possible changes through simulations that could be made in the next step, while ensuring feasibility of the disassembly of the target part.

3.3.1 Disassembly AOG Generator

Before the product can be evaluated, the disassembly-relevant structure of the AOG has to be computed (see Figure 2 for two examples of AOGs and their disassembly counterparts). This disassembly AOG is derived from the AOG together with joining-method selections and feasibility predicates. This avoids the implication that every joining operation has an inverse disassembly operation. The disassembly AOG generation component of the evaluator module serves two functions. On the one hand it performs a hard feasibility check:

$$feasible(G, p) \rightarrow \{true, false\},$$

which determines whether for a given AOG G and a target product p, there exists a sequence to disassembly that product. On the other hand, it calculates the full disassembly AOG:

$$disassemble(G, p) \rightarrow G_p^{-1}(G, p),$$

encoding all sequences to remove target part p from the final product in G.

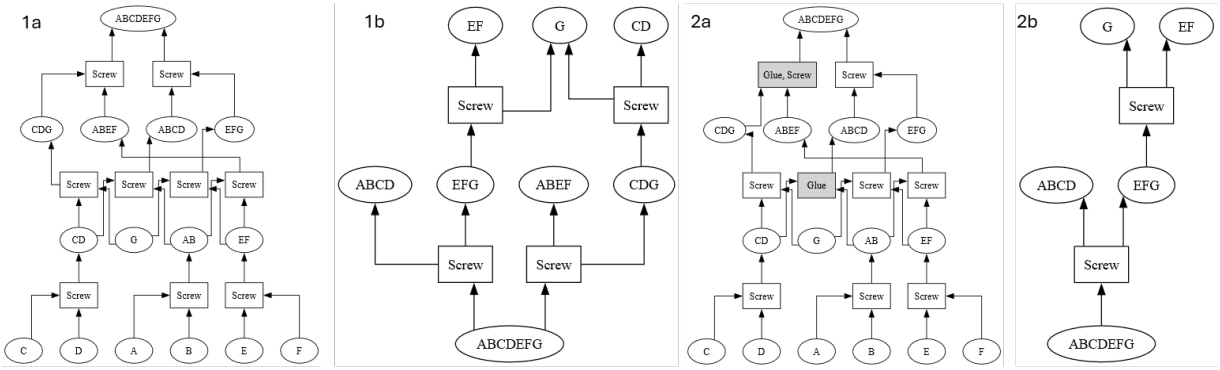

Figure 2: Comparison of a fully undoable AOG (1a) and the associated disassembly AOG for target part G (1b) with an alternative candidate (2a) where one of the disassembly paths of G is blocked by a non-undoable operation (grey) and the associated disassembly AOG (2b)

3.3.2 Scenario Generator

To access the differences of assembly and disassembly, we evaluate each candidate AOG under a set of scenarios that systematically varies the production context. The scenario generation component maps the current AOG G_t and the disassembly AOG $G_t^{-1}(G, p)$ of a candidate design to a set of scenarios in $S = \{s_1, \dots, s_n\}$, where each scenario s_i encodes a specific setting of the production system configuration, the number of target products to be produced (assemblies) and target parts to be removed (disassemblies) and the background products (AOGs and number of different products to be produced).

While the production system configuration and the background products remain constant and the total product load $n_{assembly} + n_{disassembly}$ remain constant, the ratio of disassembly to assembly tasks is varied to reveal trade-offs between the two tasks. A joining method that reduces assembly time may increase makespan in disassembly-heavy scenarios due to destructive operations, longer removal steps, or reduced accessibility.

3.3.3 Production Simulation

The production simulation component is used for the main analysis of the product. It takes the set of scenarios S and simulates each scenario returning a vector

$$\boldsymbol{y}(G, S) = \left[M\left(G, G_p^{-1}, s_1\right), \dots, M\left(G, G_p^{-1}, s_K\right)\right]^T$$

of makespans $M(G, G_p^{-1}, s_k)$, one for each scenario s_k. Inputs include: the candidate design state (G, G_p^{-1}), a production system model (resources, capabilities, setup rules), and the scenario specification (workload including assembly and disassembly tasks). The scenario-indexed makespans form the multi-objective vector used for training and analysis.

3.3.4 MOO Reward Function

To enable training with standard single-objective RL algorithms while preserving the multi-objective character of the problem, the vector $\boldsymbol{y}(G)$ of scenario results is scalarized. The MOO reward function computes the reward as the scalar weighting of the different scenario results

$$r(G) = -\sigma(\boldsymbol{y}(G), \boldsymbol{w}), \quad \sigma(\boldsymbol{y}, \boldsymbol{w}) = \sum_{k=1}^{K} w_k y_k$$

with a non-negative weight-vector $\boldsymbol{w} \in \mathbb{R}^K$.

3.3.5 Enhanced AOG Generator

The enhanced AOG generation component computes the different alternative designs that could be obtained within the next step. It iterates through all the editable joints in G using $alts(G, l, s)$ to obtain different alternatives. For each joint, it is determined whether the alternatives are feasible or not. Alternatives that do not change the undoability of a joint or change it from not undoable to undoable are directly included, while alternatives that remove the undoability of a joint the corresponding AOG is computed and its disassembly feasibility is computed using $feasible(G, p)$. Afterwards, the enhanced AOG G^e is computed by including the alternatives in the metadata of the joints.

3.4 Optimization Module

In the optimization module, the next change on the product AOG is selected. It consists of three components. On the one hand, the action generation extracts the joint changes as actions from the enhanced AOG. On the other hand, the optimization algorithms select the next action to be evaluated. Lastly, the manipulation component implements this change on the AOG.

3.4.1 Action Generation

The action generation component parses the enhanced AOG G^e to extract the possible actions $a_1, \dots, a_n$ that can be made in the next step of the optimization

$$actions(G^e) \rightarrow A = \{a_1, \dots, a_n\}$$

The set of actions is then passed to the optimization algorithms, to select the next action.

3.4.2 Optimization Algorithms

In the next step, the next change of the AOG is selected. While in the main framework, this selection is done by a reinforcement learning agent, we also use two different baselines, a random and a greedy local optimization algorithm for comparison. The optimization is carried out under a given edit budget, that defines how many optimization-evaluation iterations can be performed in total.

Reinforcement-learning implementation: The optimization problem is formulated as a finite-horizon Markov decision process. A state encodes the current design G_t, the results from the production simulation and the remaining edit budget. An action selects a graph-valid edit operator instance from the constrained set inferred from the enhanced AOG. Training uses the scalar reward $r(G_t)$ derived from the evaluation of the current design. Candidate evaluation is performed after each edit step. Invalid edits with respect to structural or manufacturability hard constraints are prevented by the given set of actions.

Random feasible edits: Starting from G_0 repeatedly sample an admissible edit from $alts(G_t, l)$ and apply it if invariants hold. Evaluate each resulting candidate across all scenarios and track the best candidate under the scalarized objective. This baseline serves as a reference to whether constraint pruning alone yields improvements without informed search.

Greedy local improvement: At each step, enumerate a neighborhood of feasible single-step edits at the current design, evaluate candidates across all scenarios under the scalarized objective, and apply the best improving edit. Terminate when no improvement of edit exists or when the edit budget is exhausted. This baseline tests whether myopic optimization suffices in the constrained space.

3.4.3 Graph Manipulator

After an action is selected by the chosen optimization algorithm, the Graph Manipulator includes this action in a new candidate AOG by changing the associated joint. Afterwards the candidate AOG is passed to the Evaluation Module for evaluation.

4 Toy use-case dataset and prototype instantiation

This section describes a current prototype that instantiates a restricted subset of the framework to verify core integration points and identify constraints and the associated toy use-case and training dataset. The prototype optimizes joining process selections in a product represented as an AOG, where each editable joining operation has two alternative methods that alter processing time and thus schedule quality. The evaluator is a scheduling solver that computes a multi-order schedule on a fixed assembly system, and the RL agent proposes method selections to reduce makespan.

4.1 Toy use-case and training dataset creation

We use the ASAP training dataset [7] as a basis to build the toy use-case and training dataset. We use all 4-to-6-part assemblies contained in the dataset. For each assembly, ASAP was used to enumerate the full geometrically feasible assembly tree and the corresponding set of feasible assembly sequences. These sequences were merged into a product-level AOG.

For the creation of the toy use-case we defined two operations: Glueing that represents a fast operation with a timespan of 2 time units, that cannot be undone later and screwing, that is slower method with a time span of 5 time units but can be undone. The training dataset generation yields 50 training instances. For each of these instances, four products are sampled from the product pool and joining methods for their joining operations are assigned randomly. One of the four sampled products is selected as the target product, and one of its components is selected uniformly at random as the disassembly target. This target component is fixed within the instance. Disassembly is defined as the removal of this target component from the assembled product. To ensure that disassembly is feasible in the toy benchmark, all joining operations that directly affect the target component are set to the undoable method and it is checked whether at least one valid disassembly sequence exists to remove the target component. For each of the three non-target products in the scenario set, an assembly demand is sampled uniformly between 1 and 10 units.

4.2 Prototype Architecture

To verify the feasibility of the framework concept, we realized it in a prototype. The prototype instantiates each of the framework components in a minimal or close-to minimal setting. Its purpose is to verify the feasibility of the architecture.

Along the lines of the AOG creation in 4.2 the knowledge module encodes the two different operations glueing and screwing that are assumed to be exchangeable if the joint in the AOG is marked as editable. The components of the disassembly and enhanced AOG generators follow along, by naively reversing the AOG, only with respect to the undoability of joining operations and by the only restriction to an editable joint being if the resulting AOG remains feasible for disassembly. In the scenario generation, four different combinations of assembly and disassembly are created (Table 1). Afterwards, the scenarios are passed to the production simulation represented by a CP-SAT-based multi-order scheduler [19], that

computes a schedule for each of the scenarios. The corresponding makespans are combined, by averaging over the entries of the makespan vector relative to the baseline makespan of the initial AOG. The enhanced AOG, the makespan vector and the reward are passed to the optimization module. The action generator extracts the actions from the enhanced AOG by scanning its joints metadata and enumerating one candidate for each editable joint and each feasible alternative method. Each action is encoded as a pair of joint and alternative method and accessed by the RL agent via a stable key. The RL agent is implemented as a tabular Q-learning agent. At each step, the stored values are updated based on the reward obtained from the last evaluation. Afterwards, the next substitution is selected from the currently available actions using an ε-greedy policy. In the graph manipulator, the enhanced AOG is reduced to a usual AOG, by changing the joint associated with the chosen action and removing the encoded alternatives. The resulting candidate AOG is passed back to the evaluator module.

Table 1: The parameters for the set of scenarios

Scenario	Assemblies	Disassemblies
1	10	0
2	8	2
3	6	4
4	4	6

The prototype was tested on the training dataset, with a limited edit budget of 10 edits and an evaluation budget of 30 seconds. While the described training dataset in combination with the prototype implementation yielded feasible candidate AOGs and suitable changes being selected by the optimizer modules, no improvements over the baselines or learning behavior could be observed during training. This is most likely due to the complex interplay between joints and makespans in the combination with a high variation of the simulated makespans under the limited budget. For a naïve setting with a trivial optimum (both operations being undoable while one being faster than the other), a learning behavior could be observed.

5 Conclusion and Outlook

This paper specified a framework concept for product prototype optimization, that constrains design exploration to graph-valid edits on an AND/OR assembly graph and evaluates candidates under assembly and disassembly scenario-based workloads. Multi-objective evaluation is realized as a vector of scenario makespans, which is scalarized during training. An MVP was presented that implements this concept for joining-method selection for two alternative joining methods, with method-dependent undoability, disassembly modeling via an inverse AOG with blocked undo operations, and scheduling-based evaluation. While the MVP produced feasible solutions, it could be seen that the state representation and limited scheduling budget were insufficient to learn a reliable policy for systematic product improvement.

Future work will build up on this basis. On the one hand, we plan to improve the current MVPs performance on the simple benchmark problem by further evaluation with longer simulation times to reduce the variability of the scheduler results and increase the quality of evaluation, whilst also improving the state space representation of the scheduler results and the associated action generation to enable the RL agent to find better policies. On the other hand, the problem and scenario definition will be enhanced and pushed closer to reality, e.g. by encoding a more realistic set of alternatives in the knowledge module and by widening the disassembly definition from a naïve binary to a more elaborate view, separating assembly and disassembly times, or by including more complex products.

In reference to the research question, we have shown that the presented framework succeeds in performing manufacturable design edits, however, the implemented prototype is not yet able to outperform heuristic baselines. To achieve this goal, further improvements both in training effort, as well as the different concept modules is needed.

Acknowledgement: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – HUMAN-DECISION (543081196), part of SPP 2443: Hybride Entscheidungsunterstützung in der Produktentstehung (521148212)

CRediT Author Statement: Sven Münker (Conceptualization, Writing), Hans Aoyang Zhou (Review, Supervision, Funding Acquisition), Anas Abdelrazeq (Review, Supervision, Funding Acquisition), Julian Haller (Review), Robert H. Schmitt (Review, Supervision, Funding Acquisition, Resources)

References

[1] Regulation (EU) 2024/1781 Establishing a Framework for the Setting of Ecodesign Requirements for Sustainable Products, https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=OJ%3AL_202401781, Accessed 26.08.2025

[2] Michael, J., Cleophas, L., Zschaler, S., Clark, T., Combemale, B., Godfrey, T., ... & Vangheluwe, H. Model-Driven Engineering for Digital Twins: Opportunities and Challenges. *Systems Engineering*.(2025).

[3] Nielsen, Christian Petersson. "Matrix-structured manufacturing systems: From design to operations." (2023).

[4] Sanderson, Arthur C., Luiz S. Homem de Mello, and Hui Zhang. "Assembly sequence planning." *AI Magazine* 11.1: 62-62. (1990).

[5] de Mello, Luiz S. Homem, and Arthur C. Sanderson. "And or graph representation of assembly plans." (1986).

[6] Münker, Sören, and Robert H. Schmitt. "CAD-based and/or graph generation algorithms in (dis) assembly sequence planning of complex products." *Procedia CIRP* 106: 144-149.(2022)

[7] Tian, Yunsheng, et al. "Asap: Automated sequence planning for complex robotic assembly with physical feasibility." *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, (2024).

[8] Mandolini, Marco, et al. "Time-based disassembly method: how to assess the best disassembly sequence and time of target components in complex products." *The International Journal of Advanced Manufacturing Technology* 95.1: 409-430.(2018).

[9] Vanegas Peña, Paul Fernando, et al. "Ease of disassembly of products to support circular economy strategies." (2018).

[10] Kondo, Yasuo, et al. "Reversibility and disassembly time of part connection." *Resources, Conservation and Recycling* 38.3: 175-184. (2003).

[11] Xiao, Youzi, et al. "Knowledge graph-based manufacturing process planning: A state-of-the-art review." *Journal of Manufacturing Systems* 70: 417-435.(2023).

[12] Cao, Jianpeng, et al. "Ontology-based manufacturability analysis automation for industrialized construction." *Automation in Construction* 139: 104277.(2022).

[13] Shi, Xiaolin, et al. "Knowledge graph-based assembly resource knowledge reuse towards complex product assembly process." *Sustainability* 14.23: 15541.(2022).

[14] Amaran, Satyajith, et al. "Simulation optimization: a review of algorithms and applications." *Annals of Operations Research* 240.1: 351-380.(2016).

[15] Möhring, Rolf H., Martin Skutella, and Frederik Stork. "Scheduling with AND/OR precedence constraints." *A 2-page abstract appeared in Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'00), 235-236.*. (2000).

[16] Guo, Daqiang, et al. "A roadmap for Assembly 4.0: self-configuration of fixed-position assembly islands under Graduation Intelligent Manufacturing System." *International Journal of Production Research* 58.15: 4631-4646.(2020).

[17] Roijers, Diederik M., et al. "A survey of multi-objective sequential decision-making." *Journal of Artificial Intelligence Research* 48 (2013): 67-113.(2013).

[18] Huang, Shengyi, and Santiago Ontañón. "A closer look at invalid action masking in policy gradient algorithms." *arXiv preprint arXiv:2006.14171* (2020).

[19] Davies, Toby O., Frédéric Didier, and Laurent Perron. "Violationls: constraint-based local search in cp-sat." *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Cham: Springer Nature Switzerland, (2024).