Feasibility-Aware Sequential Synthesis of Structural Support Assemblies: a Canonical Formulation for Automated MEP Support Design

Lavinia Pedrollo[1], Torben Graeber[2], and Martin Fischer[3]

[1]Stanford University, Stanford, CA, USA
[2]Hilti AG, Schaan, Liechtenstein
[3]Stanford University, Stanford, CA, USA

laviniap@stanford.edu, torben.graeber@hilti.com, fischer@stanford.edu

Abstract -

Structural Support Assemblies (SSAs) connect Mechanical, Electrical, and Plumbing (MEP) services to building structures under strict geometric, catalog-defined compatibility, and structural constraints. MEP systems account for a substantial share of construction cost and project complexity, yet SSA design remains largely manual even in BIM-enabled workflows, driven by the combinatorial complexity of catalog-constrained component selection and load-path feasibility requirements. Existing tools verify or size support configurations after a topology has been fixed, leaving the upstream synthesis problem, deciding which components to use and in what sequence, handled through practitioner heuristics and iterative manual revision. SSA synthesis is strongly order-dependent: early selections restrict the future design space by inducing geometric exclusions, redistributing loads, and activating catalog-specific compatibility constraints. Engineers frequently encounter situations where locally constraint-valid decisions lead to a partial assembly that cannot be completed. We refer to this irreversible loss of solvability as feasibility collapse. This paper formulates SSA synthesis as a constrained sequential decision-making problem in which partial assemblies are states, component placements are actions, and feasibility collapse is a structural property of the design space rather than a post hoc validation outcome. Two ceiling-mounted examples illustrate component-level branching and topology-level commitment, including feasibility collapse induced by locally valid early decisions. The contribution is conceptual and foundational: no specific algorithm is proposed. Instead, the formulation provides a canonical problem structure for feasibility-aware automation of SSA design and a reusable abstraction for heuristic search, constraint-aware planning, and reinforcement learning methods in construction domains involving sequential assembly decision processes.

Keywords -

Structural Support Assemblies; Sequential Decision Making; Markov Decision Process; Constrained Design; Design Automation; Construction Automation

1 Introduction

Structural Support Assemblies (SSAs) connect Mechanical, Electrical, and Plumbing (MEP) services such as pipes, ducts, and cable trays to the primary building structure [1, 2, 3, 4]. These assemblies form the critical interface between service systems and structural elements, as they must simultaneously satisfy constraints of different natures: (i) geometric constraints, (ii) catalog-defined compatibility rules and the availability of compatible connection interfaces as an assembly is built, and (iii) structural requirements such as capacity limits and continuous load-path connectivity.

MEP systems account for a substantial share of construction cost and project complexity [5], and BIM-based coordination has been shown to reduce clashes, rework, and labor compared to traditional 2D practices [2, 6]. Despite these advances, SSA design in BIM-enabled workflows remains largely performed through manual engineering judgment, driven by the high combinatorial complexity of catalog-constrained component selection and the strict feasibility requirements of load-path construction [1, 7, 6]. While BIM provides accurate geometric representation and clash detection [8], it does not automate the synthesis of support assemblies: designers must still decide which modular components to select and in what sequence to place them to form valid load paths under the aforesaid constraints [4, 9]. This process is time-consuming and error-prone, particularly at scale, motivating the development of automated synthesis methods.

Commercial tools such as Hilti PROFIS provide mature support for structural validation, sizing, and code compliance of support systems [4, 9]. Rule-based systems verify geometric and code constraints, and optimization-based tools refine capacities and dimensions for predefined topologies [4]. However, these approaches assume that a complete support topology has already been specified. As a result, the upstream synthesis problem, deciding which components to use and in what order to assemble them, remains largely unsupported and is instead handled through experience-driven heuristics and trial-and-error [1].

An SSA is composed of modular components drawn from a finite manufacturer catalog [10], including elements such as connectors, hangers, threaded rods, channels, and pipe clamps. Figure 1 provides a representative example of a ceiling-mounted SSA, illustrating the modular components and connection interfaces that form a load path from supported services to the building structure.

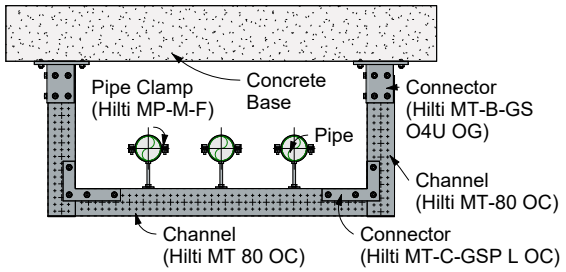

Figure 1. Example ceiling-mounted SSA. Pipes and concrete slab are given as input to the designer, while modular catalog components are selected and assembled to form a continuous load path.

For a given location, designers incrementally construct an assembly that establishes a valid load path to the building structure while maintaining clearance from surrounding services and respecting catalog-defined compatibility rules and the evolving availability of connection interfaces [4]. The design process is inherently sequential and order-dependent: early decisions regarding connector type, geometry, or channel length and orientation irreversibly restrict the set of admissible downstream connections. While such decisions may satisfy all immediately applicable geometric, structural, and catalog-defined compatibility constraints when made, they can still preclude completion of a valid load path at later stages. This irreversible loss of solvability, despite locally valid intermediate decisions, is referred to as *feasibility collapse*.

Static optimization approaches abstract away sequencing effects by assuming fixed topologies, while rule-based systems act only as post hoc validators and cannot reason about latent infeasibility caused by early design choices [11]. Reinforcement Learning (RL) methods in construction planning have focused primarily on high-level task sequencing or scheduling rather than the fine-grained synthesis of physical support components under strict feasibility constraints [12, 13, 14].

Consequently, SSA synthesis currently lacks any formal abstraction within the existing literature. We therefore introduce a decision-theoretic formulation that models SSA synthesis as a constrained sequential decision-making problem [15]. Partial assemblies define the system state; modular component placements define actions; and the governing constraints determine the resulting state transitions. Feasibility is treated as a first-class concern in the state-space representation: certain sequences of locally valid actions lead to states from which no feasible complete assembly can be reached. Preserving the possibility of feasibility throughout the decision sequence therefore becomes a central concern of SSA synthesis.

The contribution of this work is conceptual and foundational. We provide a formal problem formulation that (i) models SSA construction as a sequential decision process, (ii) explicitly represents feasibility collapse as an irreversible state-space phenomenon, and (iii) defines a canonical abstraction suitable for heuristic search, constraint-aware planning, and RL. No algorithms are presented; rather, this work establishes, to our knowledge, the first formal decision-theoretic formulation of SSA synthesis, intended to serve as a rigorous foundation upon which future optimization, search, and RL-based methods for automated SSA design can be developed.

2 Related Work

BIM has transformed MEP coordination by enabling 3D visualization, automated clash detection, and multi-disciplinary collaboration [6, 8]. Teo et al. [2] and Lee et al. [7] show that BIM-based workflows significantly improve coordination and clash resolution, but these methods focus on routing and layout of services rather than the detailed synthesis of individual support assemblies. The automated generation of support structures remains a gap in current BIM workflows [3, 6]. Industrial support systems are assembled from standardized catalogs of modular components governed by strict compatibility and load-bearing rules [10]. Manufacturer software like Hilti PROFIS Engineering enables engineers to validate and optimize support configurations for predefined topologies [9]. However, these tools assume that the designer has already selected the assembly structure and connection sequence. The upstream synthesis problem, deciding which components to use and in what order to assemble them, remains largely unsupported.

From a formal perspective, SSA synthesis relates to sequential decision making, constraint satisfaction, and planning. Markov Decision Processes (MDPs) and absorbing Markov chains model systems in which certain states, once entered, cannot be left [16, 15], aligning with our notion of feasibility collapse. Constraint satisfaction methods use constraint propagation to maintain feasibility [17, 18], but do not naturally capture irreversible, order-dependent assembly sequences. Feasibility-aware decision-focused learning [19] and constrained planning [20] similarly emphasize maintaining feasibility across decision horizons, but have not been applied to catalog-driven support assembly. Related problems have been studied in robotics through Task and Motion Planning (TAMP) and Assembly Sequence Planning (ASP). TAMP integrates sym-

bolic task-level decisions with continuous motion planning [21, 22], and ASP searches for feasible component assembly orders [23, 24, 25, 26]. These methods typically assume known assembly relationships or focus on kinematic feasibility, whereas SSA synthesis requires reasoning over discrete, catalog-defined components whose placement alters future compatibility and feasibility. Reinforcement Learning (RL) has been applied in construction for resource allocation, equipment control, and structural design [12, 13, 27], including crane planning [28] and robotic assembly [29]. However, these applications do not model the fine-grained, catalog-constrained synthesis of support assemblies or the feasibility collapse induced by irreversible design decisions. Taken together, existing BIM tools, modular support software, planning methods, and RL approaches excel at validating or optimizing predefined structures, but do not address the upstream synthesis problem in which topology itself emerges from a sequence of constrained decisions. Our formulation bridges this gap by modeling SSA synthesis as a feasibility-critical sequential decision process, in which irreversible loss of solvability emerges as an intrinsic state-space property rather than a post hoc validation outcome. Section 3 formalizes this problem.

3 Problem Definition

We consider the synthesis of a single SSA at a fixed spatial location, given a predefined local MEP configuration. The objective is to determine a sequence of component placements, drawn from a finite manufacturer catalog, that constructs a complete and feasible support assembly. Feasibility requires satisfaction of geometric constraints, catalog-defined compatibility and interface rules, structural capacity limits, and continuous load-path connectivity from the supported services to the building structure.

The problem inputs are: (i) a local MEP cross-section describing service geometries, (ii) the supporting structural element (e.g., slab, beam, or wall), (iii) a finite set of allowable attachment locations, and (iv) a manufacturer-specific catalog of modular components with prescribed geometry, capacities, and compatibility rules. Clearance requirements, spacing rules, load transfer behavior, service loads, and structural properties are treated as deterministic. A single SSA may support multiple MEP services; however, global MEP routing, interactions between distinct supports, cost optimization, uncertainty, and site-level scheduling are outside the scope of this work.

Synthesis proceeds incrementally: at each step, a catalog component is placed and connected relative to the existing partial assembly, the MEP services, and the supporting structure. These placements are irreversible and restrict future actions through induced geometric, interface, and structural constraints.

We distinguish three related concepts over partial and complete assemblies. Let $\mathcal{F}(s)$ denote the set of complete assemblies reachable from partial assembly s via sequences of constraint-valid placements. A partial assembly is *constraint-valid* if all immediately applicable geometric, compatibility, and structural constraints are satisfied at the current construction stage. It is *solvable* if $\mathcal{F}(s) \neq \emptyset$, i.e., at least one such sequence yields a complete feasible assembly. A complete assembly is *feasible* if it is constraint-valid and establishes a continuous valid load path to the supporting structure. Feasibility implies solvability, and solvability implies constraint-validity, but neither converse holds in general. Because placements are irreversible, a sequence of individually constraint-valid decisions may nonetheless render the partial assembly unsolvable. We refer to this transition, from a state that is both constraint-valid and solvable to one that is constraint-valid but unsolvable, as *feasibility collapse*. Table 1 summarizes these concepts, and Section 4 formalizes the resulting decision process.

Table 1. Key concepts in SSA synthesis. Feasibility implies solvability, and solvability implies constraint-validity, but not vice versa.

Concept	Applies to	Condition
Constraint validity	State s (any)	All geometric, catalog, and structural constraints satisfied
Solvability	State s (non-terminal)	$\mathcal{F}(s) \neq \emptyset$: at least one feasible completion reachable
Feasibility	State s (terminal)	Constraint-valid with a continuous load path to the supporting structure
Feasibility collapse	Transition $s \to s'$	s solvable; s' constraint-valid but $\mathcal{F}(s') = \emptyset$

4 Canonical Formulation of SSA Synthesis

We formalize SSA synthesis as a constrained sequential decision-making problem in which partial assemblies define states, component placements define actions, and feasibility collapse is explicitly represented as an absorbing terminal set. The concepts introduced in Table 1 are carried through this formulation.

4.1 State Space

Let $\mathcal{S}$ denote the set of all possible assembly configurations. Each state $s_t \in \mathcal{S}$ represents the assembly at decision step t and encodes: (i) the decision step index t, (ii) the set of installed components, (iii) their geometric placement and orientation, (iv) component connectivity and induced load-transfer relationships, and (v) all constraints on future placements activated by prior decisions.

States are assumed Markov and fully observable: each s_t contains all information required to evaluate the

constraint-validity and solvability of any candidate placement. The initial state s_0 corresponds to the empty assembly at $t = 0$, which is trivially constraint-valid and solvable. $\mathcal{S}$ is partitioned into three disjoint classes:

- $\mathcal{S}_{\text{valid}} \subset \mathcal{S}$: non-terminal states representing incomplete assemblies in which all geometric, catalog, and structural constraints induced by prior placements are satisfied and at least one action remains available. The initial state $s_0 \in \mathcal{S}_{\text{valid}}$. A valid state may or may not admit a feasible completion; this distinction is a property of the state rather than a class boundary, and is discussed further in Sections 4.3 and 4.4.
- $\mathcal{S}_{\text{success}} \subset \mathcal{S}$: terminal states representing complete, feasible assemblies in which all supported services are connected to the building structure through a continuous valid load path. These states are absorbing under T, as no further placements are required.
- $\mathcal{S}_{\text{fail}} \subset \mathcal{S}$: terminal states from which no successful completion is possible. This class subsumes two distinct failure modes: (a) *constraint violation*, in which a placement action violates one or more geometric, interface, or structural constraints, rendering the state invalid; and (b) *feasibility collapse*, in which the state is constraint-valid but no feasible completion exists, i.e., $\mathcal{F}(s) = \emptyset$. Both failure modes are absorbing under T. Constraint violations are detectable via local checks; feasibility collapse requires global reasoning over $\mathcal{F}(s)$ and is the central planning challenge of SSA synthesis.

4.2 Action Space

Let C denote the finite manufacturer catalog of modular support components. An action consists of selecting a component together with a discrete, catalog-consistent placement parameterization (e.g., location, orientation, and interface), such that $\mathcal{A} \subseteq C \times \mathcal{P}$, where $\mathcal{P}$ is a finite set of allowable placement parameterizations.For a given state $s \in \mathcal{S}_{\text{valid}}$, the available action set $\mathcal{A}(s) \subseteq \mathcal{A}$ is the subset of actions not eliminated by deterministic local checks, including geometric clash detection, interface compatibility, and load capacity verification. Once a terminal state $s \in \mathcal{S}_{\text{fail}} \cup \mathcal{S}_{\text{success}}$ is reached, no further actions are considered. Above all, $\mathcal{A}(s)$ is not guaranteed to contain only safe actions: a locally admissible action may still induce feasibility collapse, transitioning the assembly to $\mathcal{S}_{\text{fail}}$ for reasons undetectable by local checks alone.

4.3 Transition Dynamics

The system evolves deterministically according to a transition function $T : \mathcal{S}_{\text{valid}} \times \mathcal{A} \to \mathcal{S}$. For any action $a_t \in \mathcal{A}(s_t)$, the successor state $s_{t+1} = T(s_t, a_t)$ augments the partial assembly and updates all induced constraints. Transitions are irreversible: once a component is placed, its geometry, connectivity, and load-transfer relationships persist, and component removal or modification is not permitted. A successor state s_{t+1} may belong to any of the three classes: $\mathcal{S}_{\text{valid}}$ if the assembly remains incomplete and constraint-valid, $\mathcal{S}_{\text{success}}$ if a complete valid load path has been established, or $\mathcal{S}_{\text{fail}}$ if feasibility collapse has occurred, i.e., $\mathcal{F}(s_{t+1}) = \emptyset$. Note that $\mathcal{A}(s_t)$ does not guard against feasibility collapse: an action may be locally constraint-valid yet render the assembly globally unsolvable. This failure mode is undetectable by local constraint checking alone and constitutes the central planning challenge of SSA synthesis.

4.4 Objective

The objective is to identify a sequence of actions $(a_0, \ldots, a_{N-1})$ that transitions the system from $s_0 \in \mathcal{S}_{\text{valid}}$ to a terminal state $s_N \in \mathcal{S}_{\text{success}}$ while avoiding $\mathcal{S}_{\text{fail}}$. This objective may be expressed without an explicit reward function, focusing solely on feasibility preservation, or equivalently through a sparse reward formulation in which success yields a positive reward and entry into $\mathcal{S}_{\text{fail}}$ yields a penalty. The magnitude of the penalty may differentiate between constraint violation and feasibility collapse if the distinction is relevant to the solution method. The specific representation is left open, as the purpose of this formulation is to define the problem structure rather than prescribe a solution method.

5 Illustrative Examples

This section presents illustrative examples of SSA synthesis for ceiling-mounted MEP supports using components from the Hilti Modular Support Systems catalog [10]; all loads, dimensions, and capacities are illustrative and not design-approved. The examples illustrate how catalog-defined compatibility constraints, geometry, and irreversible sequencing shape the feasible assembly space. Example 1 illustrates component-level branching within a fixed topology, while Example 2 illustrates multiple feasible topologies and irreversible topology commitments induced by early decisions.

5.1 Example 1: Component-Level Branching

A single pipe is to be supported from a concrete ceiling using components drawn from a finite catalog. The pipe carries a vertical load $P = 200$ lbf and is located at a vertical offset h below the ceiling, with a single ceiling attachment location x_c. The catalog is $C_{\text{Ex1}} = \{c^{\text{rod}}, c^{\text{hanger}}, c^{\text{clamp}}_{250}, c^{\text{clamp}}_{150}\}$, where each component may be placed at most once. Feasible assemblies must respect interface compatibility and capacity limits while preserving valid load transfer. Figure 2 shows the feasibility-pruned state–action tree; branches terminate upon constraint violation. Feasibility collapse is illustrated in Example 2.

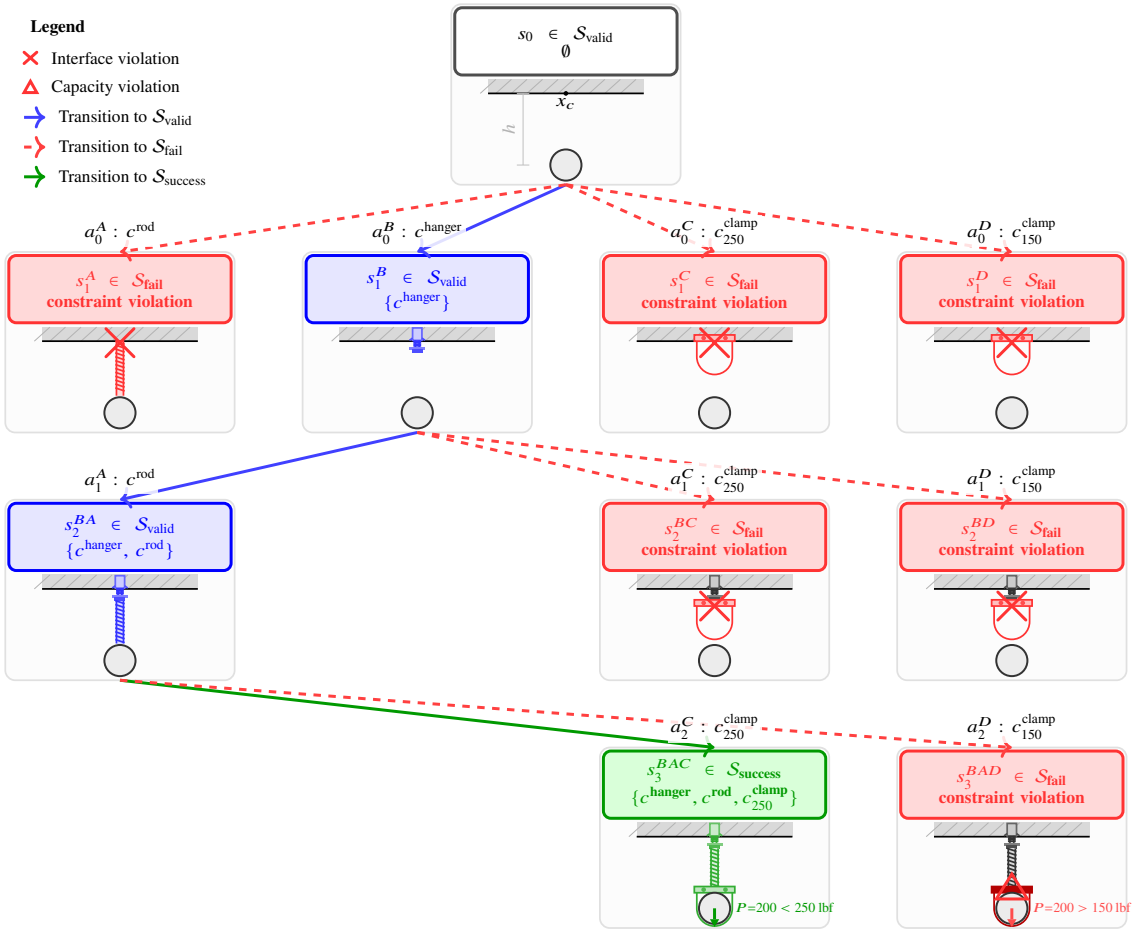

Figure 2. Feasibility-pruned state–action tree for Example 1. Each state represents a partial assembly classified as valid (blue), fail (red), or success (green), where fail states indicate violations of interface compatibility or capacity constraints. Actions a correspond to component placements, and arrows indicate transitions to the next state obtained by taking an action. The superscripts (A, B, C, D) denote the component selected at each step, while subscripts indicate the step index. Any infeasible configuration terminates immediately. Only one sequence yields a complete feasible assembly, while all others end in fail states.

The process begins at $s_0 \in \mathcal{S}_{\text{valid}}$, corresponding to the empty assembly, and at each step an action selects a previously unused component and places it in a catalog-consistent configuration at x_c. Several actions lead to immediate failure due to interface incompatibility, transitioning directly to $\mathcal{S}_{\text{fail}}$. Only one sequence yields a complete assembly that satisfies all constraints and establishes a valid load path, reaching a terminal state in $\mathcal{S}_{\text{success}}$.

This example illustrates how feasibility constraints drastically reduce the effective search space. Although multiple catalog actions are nominally available at early stages, most are eliminated either by immediate interface infeasibility or by capacity violations encountered when loads are finally transferred to the clamp. In this example all failures are detected immediately via local constraint checks, so the feasibility-pruned tree coincides with the constraint-violation-pruned tree; the distinction between local constraint-validity and global solvability becomes consequential in Example 2.

Because each component is used at most once, the number of nominal length-k ordered sequences is the falling factorial $\left(|C_{\text{Ex1}}|\right)_k$. In this example, a complete single-rod assembly requires three components (hanger, rod, and clamp), so $k = 3$. For $|C_{\text{Ex1}}| = 4$, this yields 24 nominal sequences, yet only one reaches $\mathcal{S}_{\text{success}}$, eliminating 95.8% of the search space by feasibility alone.

5.2 Example 2: Topology-Level Branching

As in Example 1, a single pipe carries a vertical load $P = 200$ lbf and is to be supported from a concrete ceiling at a vertical offset h. Here, however, three candidate ceiling attachment points are given, $\mathcal{X} = \{x_1, x_2, x_3\}$ with $x_1 < x_2 < x_3$, where $x_2 = x_c$ coincides with the pipe centerline projection and x_1, x_3 are exterior points enabling a

trapeze-type support. Only ceiling-mounted threaded rod hangers may be installed at locations in $\mathcal{X}$. We restrict the available catalog to one representative component from each family, $\mathcal{C}_{\text{Ex2}} = \{c^{\text{hanger}}, c^{\text{rod}}_{\text{sr}}, c^{\text{rod}}_{\text{tr}}, c^{\text{tr}}, c^{\text{cl}}_{\text{sr}}, c^{\text{cl}}_{\text{tr}}\}$. Here, c^{hanger} denotes a ceiling-mounted threaded rod hanger; $c^{\text{rod}}_{\text{sr}}$ and $c^{\text{rod}}_{\text{tr}}$ are threaded rods used in single-rod and trapeze configurations; c^{tr} is a trapeze channel; and $c^{\text{cl}}_{\text{sr}}$, $c^{\text{cl}}_{\text{tr}}$ are clamps attaching the pipe to a rod or trapeze channel, respectively. Each element represents a component *family*: unlike Example 1, where each catalog entry is a unique component, here each entry represents a family and multiple units may be placed at distinct locations. Two feasible backbone support topologies exist for this configuration, whose spatial layouts are shown in Figure 3.

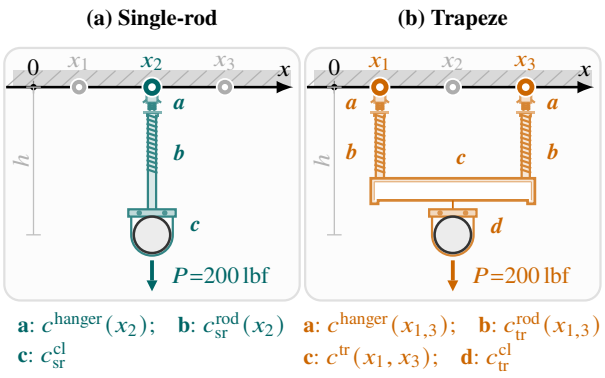

Figure 3. Spatial layout for Example 2. **(a)** Single-rod (teal): hanger, rod, and clamp at x_2 form a direct vertical load path. **(b)** Trapeze (orange): hangers and rods at x_1 and x_3 suspend a channel from which the pipe is clamped at mid-span. Both topologies are feasible from s_0; early placement decisions irreversibly commit to one, and constraint-valid actions may silently eliminate both (feasibility collapse).

Two feasible backbone topologies exist, each reached by a distinct feasibility-preserving action sequence summarized in Table 2. Both sequences terminate in $\mathcal{S}_{\text{success}}$.

Table 2. Action sequences for Example 2.

Step	Action	Component placed
Single-rod topology ($s_0 \to s_3 \in \mathcal{S}_{\text{success}}$)		
0	a_0	c^{hanger} at x_2
1	a_1	$c^{\text{rod}}_{\text{sr}}$ at x_2
2	a_2	$c^{\text{cl}}_{\text{sr}}$ on rod; transfers P
Trapeze topology ($s_0 \to s'_6 \in \mathcal{S}_{\text{success}}$)		
0	a'_0	c^{hanger} at x_1
1	a'_1	$c^{\text{rod}}_{\text{tr}}$ at x_1
2	a'_2	c^{hanger} at x_3
3	a'_3	$c^{\text{rod}}_{\text{tr}}$ at x_3
4	a'_4	c^{tr} spanning (x_1, x_3)
5	a'_5	$c^{\text{cl}}_{\text{tr}}$ on channel; transfers P

Topology commitment emerges from early constraint-valid actions and is irreversible. Suppose c^{hanger} is placed at x_2 followed by $c^{\text{rod}}_{\text{tr}}$ at x_2: both actions satisfy all local interface and geometric constraints, yet the resulting state $s^*_2 = \{c^{\text{hanger}}(x_2), c^{\text{rod}}_{\text{tr}}(x_2)\}$ has $\mathcal{F}(s^*_2) = \emptyset$. The single-rod topology requires $c^{\text{rod}}_{\text{sr}}$ at x_2, now blocked by an incompatible type; the trapeze topology requires $c^{\text{rod}}_{\text{tr}}$ at x_1 and x_3, rendering the placement at x_2 structurally irrelevant. Neither topology can be completed: feasibility collapse has occurred through a locally valid action sequence. More generally, once c^{hanger} and $c^{\text{rod}}_{\text{sr}}$ are placed at x_2 (state s_2), $\mathcal{F}(s_2) \neq \emptyset$ but trapeze completions are permanently excluded: $c^{\text{rod}}_{\text{sr}}$ occupies x_2 and is incompatible with the trapeze channel interface c^{tr}, which requires $c^{\text{rod}}_{\text{tr}}$ at both x_1 and x_3. Conversely, after the trapeze backbone is installed at s'_5, only trapeze completions remain in $\mathcal{F}(s'_5)$; any attempt to realize the single-rod topology transitions to $\mathcal{S}_{\text{fail}}$ under T.

6 Discussion

The canonical formulation presented in this paper exposes the core difficulty of SSA synthesis: feasibility is not a property of individual component choices but of action sequences. The examples in Section 5 demonstrate how feasibility constraints shape the decision space in practice: Example 1 shows how local constraint violations drastically prune the feasible action space, while Example 2 shows how locally constraint-valid placements can irreversibly eliminate all feasible completions, a failure mode that neither rule-based validation nor topology-fixed optimization can anticipate.

This explains why existing approaches are structurally misaligned with the upstream synthesis problem. Rule-based systems verify constraints against a fixed topology but cannot detect feasibility collapse, since no local constraint is violated at the point of introduction. Static optimization methods assume a fixed topology and abstract away sequencing effects, and are therefore unable to detect feasibility collapse arising from the order in which components are placed. Experience-driven heuristics are likewise inadequate, as they offer no guarantee of preserving solvability across the decision sequence. The proposed formulation resolves this misalignment by treating topology as an emergent property of the decision sequence rather than a design input.

The formulation is agnostic to representational choices. Partial assemblies may be encoded as graphs whose nodes correspond to components or interfaces and whose edges encode geometric or load-transfer relationships, as commonly done in assembly and planning formulations [24]. Catalog compatibility rules, geometric constraints, and structural checks can be implemented as state-dependent action masks or feasibility filters [17], allowing rule-based,

geometric, and learned components to coexist within a unified decision-making framework.

Although no algorithms are proposed in this work, the formulation naturally supports multiple computational paradigms. Heuristic search can prioritize states that preserve future solvability [11], constraint-aware planning can propagate catalog and geometric constraints to prune infeasible branches [21, 17], and RL can learn feasibility-preserving policies through interaction with a catalog-based simulator [12, 13]. Hybrid symbolic-learning methods are natural extensions, since the formulation already accommodates symbolic components via action masks and feasibility filters alongside learned policies via RL. Although the theoretical state space is combinatorial, most partial assemblies are eliminated by local constraint violations; sparse representations with dynamic action masking make feasibility filtering tractable even for industrial catalogs [10].

The formulation also provides a foundation for benchmarking future methods. Because the problem structure is now precisely defined, researchers can assess solvers on feasibility success rate, solvability preservation, and solution quality, and can narrow or extend the formulation to specific catalog families, load conditions, or assembly scales without losing the core sequential decision structure. This extensibility is not specific to SSA synthesis: the formulation applies to any sequential design or assembly problem with irreversible feasibility loss, including modular product configuration, robotic assembly, and spatial layout synthesis [30]. In all such domains, feasibility is path-dependent rather than state-local and cannot be captured by static optimization, post hoc rule checking, or experience-driven heuristics.

7 Conclusion

This paper introduces, to our knowledge, the first canonical formulation of SSA synthesis as a constrained sequential decision process with absorbing unsolvable states. By representing partial assemblies as states, catalog component placements as actions, and feasibility collapse as an intrinsic state-space property, the formulation captures why early irreversible decisions can silently eliminate all feasible completions, a failure mode invisible to static optimization, post hoc rule-based validation, and experience-driven heuristics. The formal distinction between constraint-validity, solvability, and feasibility, illustrated through two ceiling-mounted single-pipe examples examining component-level and topology-level branching, clarifies why SSA synthesis cannot be reduced to topology-fixed optimization or post hoc constraint checking. The result is a canonical problem structure that unifies heuristic search, constraint-aware planning, and reinforcement learning under a common abstraction. The formulation is theoretical and proposes no algorithms; computational validation and solver development remain open. Future work includes developing RL solvers that learn feasibility-preserving policies from catalog-based simulation, constructing standardized SSA benchmarks that capture the combinatorial and topological diversity of industrial MEP installations, characterizing how feasibility collapse scales with catalog size and support complexity, and validating automated designs against expert solutions in practice.

References

[1] C. B. Tatum and T. Korman. MEP coordination in building and industrial projects. CIFE Working Paper No. 54, Stanford University. On-line: `https://stacks.stanford.edu/file/druid:vn180wh3959/WP054.pdf`, Accessed: 30/01/2026.

[2] Y. H. Teo, J. H. Yap, H. An, S. C. M. Yu, L. Zhang, J. Chang, and K. H. Cheong. Enhancing the MEP coordination process with BIM technology and management strategies. *Sensors*, 22(13), 2022. doi:10.3390/s22134936.

[3] T. M. Korman, L. Simonian, and E. Speidel. Using building information modeling to improve the mechanical, electrical, and plumbing coordination process for buildings. In *Proceedings of AEI 2008*, pages 1–10, Denver, CO, USA, 2008.

[4] Hilti Corporation. Product technical guides. On-line: `https://www.hilti.com/content/hilti/W1/US/en/business/business/engineering/product-technical-guides.html`, Accessed: 14/01/2026.

[5] W. Wei, Y. Lu, R. Bai, L. Zhong, Y. Chen, Y. Lin, and J. C. P. Cheng. Enhanced MEP construction progress tracking using panoramic mobile positioning and optimized pipeline segmentation. *Automation in Construction*, 180:106487, 2025. doi:10.1016/j.autcon.2025.106487.

[6] A. Khanzode, M. Fischer, and D. Reed. Benefits and lessons learned of implementing building virtual design and construction (VDC) technologies for coordination of mechanical, electrical, and plumbing (MEP) systems on a large healthcare project. *Journal of Information Technology in Construction*, 13: 324–342, 2008.

[7] N. Lee, C. S. Dossick, and S. P. Foley. Comparison of experienced and novice BIM coordinators in performing mechanical, electrical, and plumbing (MEP) coordination tasks. In *Construction Research Congress 2016*, pages 177–186, 2016.

[8] C. M. Eastman, P. M. Teicholz, R. Sacks, and G. Lee. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*. John Wiley & Sons, Hoboken, NJ, USA, 3rd edition, 2018.

[9] Hilti Corporation. PROFIS engineering suite. On-line: `https://www.hilti.com/content/hilti/W1/US/en/business/business/engineering/profis-engineering.html`, Accessed: 14/01/2026.

[10] Hilti, Inc. Modular support systems. On-line: `https://www.hilti.com/c/CLS_MODULAR_SUPPORT_SYSTEM`, Accessed: 14/01/2026.

[11] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, UK, 2006.

[12] Y. Pan and L. Zhang. Reinforcement learning in construction engineering and management: A review. *Journal of Construction Engineering and Management*, 148(11):03122001, 2022. doi:10.1061/(ASCE)CO.1943-7862.0002386.

[13] S. Jeong and H. Jo. Deep reinforcement learning for automated design of reinforced concrete structures. *Computer-Aided Civil and Infrastructure Engineering*, 37(12):1508–1529, 2022. doi:10.1111/mice.12773.

[14] A. De Giorgio, A. Maffei, M. Onori, and L. Wang. Towards online reinforced learning of assembly sequence planning with interactive guidance systems for industry 4.0 adaptive manufacturing. *Journal of Manufacturing Systems*, 60:22–34, 2021. doi:10.1016/j.jmsy.2021.05.001.

[15] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Hoboken, NJ, USA, 2nd edition, 2014.

[16] J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. Springer-Verlag, New York, 1976.

[17] R. Bartak. Constraint propagation and backtracking-based search. Lecture notes, Charles University in Prague, 1998.

[18] P. Brucker and S. Knust. Scheduling and constraint propagation. *Discrete Applied Mathematics*, 123(1–3):227–256, 2002.

[19] X. Wang, W. Hu, and J. Tang. Feasibility-aware decision-focused learning for predicting parameters in the constraints. On-line: `https://arxiv.org/abs/2510.04951`, Accessed: 30/01/2026.

[20] A. Badanidiyuru Varadaraja. Sequential decision making with resource constraints. PhD dissertation, Cornell University, 2014.

[21] C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:265–293, 2021. doi:10.1146/annurev-control-091420-084139.

[22] M. Mansouri, F. Pecora, and I. Scholl. Combining task and motion planning: Challenges and guidelines. *Frontiers in Robotics and AI*, 8:637888, 2021.

[23] P. Jimenez. A survey on assembly sequence planning and assembly line balancing optimisation using soft computing approaches. *The International Journal of Advanced Manufacturing Technology*, 57(5):763–779, 2011.

[24] C. J. Heemskerk, Y. Kassahun, and L. Methnani. Efficient and feasible robotic assembly sequence planning via graph representation learning. In *Proceedings of the 42nd International Symposium on Automation and Robotics in Construction*, Lille, France, 2023.

[25] H. Hashemi and Y. Guo. Effective assembly sequence planning through graph representation and reinforcement learning. *Journal of Manufacturing Systems*, 67:21–35, 2023.

[26] T. Lozano-Pérez and L. P. Kaelbling. A constraint-based method for solving sequential manipulation planning problems. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3684–3691, Tokyo, Japan, 2013.

[27] K. Kim and H. Kim. Reinforcement learning applications in construction: A review. *Automation in Construction*, 136:104177, 2022.

[28] J. Zhao, Z. Liu, and X. Zhou. A reinforcement learning based construction material supply system for robotic crane. *Automation in Construction*, 139: 104280, 2022.

[29] J. Kim and I. Brilakis. Robotic disassembly sequence planning from BIM for automated resource recovery. In *Proceedings of the 42nd International Symposium on Automation and Robotics in Construction*, pages 125–132, Montreal, Canada, 2025.

[30] W. Wan and K. Harada. Integrated assembly and motion planning using regrasp graphs. *Robotics and Biomimetics*, 3(18):1–16, 2016.