Mathematical Framework for Seven-Layer Feasibility Checking [STAGE - 4]

TEAM - LUMEN [TEAM-ID: 93912]

Contents

1 Introduction

Feasibility checking in combinatorial timetabling detects fundamental impossibilities before optimization, saving computational effort. We present a comprehensive, layered mathematical framework wherein each stage utilizes progressively stronger inference to prune infeasible instances with maximal efficiency. Each layer is justified theoretically and accompanied by proof of necessity for its constraints.

2 Layer 1: Data Completeness Schema Consistency

2.1 Formal Statement

Verify that all tuples satisfy declared schemas, unique primary keys, null constraints, and all functional dependencies in the dataset.

2.2 Algorithmic Procedure

Given table set $\mathcal{T}$, for each $T \in \mathcal{T}$:

- Check $\forall$ record $t \in T$, $\forall$ key attribute k: $t[k] \neq \varnothing$ (no null keys)
- Assert |keys| =unique(keys)
- For every FD $X \to Y$, $\forall$ group g with same X-value, Y is unique

2.3 Mathematical Properties

Lemma: The accepted instance is in Boyce–Codd Normal Form (BCNF) with respect to declared FDs.

2.4 Detectable Infeasibility

Schema errors, missing critical data, FD violations (e.g., multiple names for one course).
When caught: Always, since optimization requires unambiguous and complete input.

3 Layer 2: Relational Integrity Cardinality

3.1 Formal Statement

Model the schema as a directed multigraph of tables; each directed edge $(A \to B)$ denotes a FK from A to B. Additionally, each FK may carry a cardinality constraint (ℓ, u).

3.2 Algorithmic Procedure

- Detect cycles of mandatory FKs (where nulls not allowed): perform topological sort; failure implies cycle.
- For every relationship, for all $a \in A$, count c_{ab} children in B: check $\ell \leq c_{ab} \leq u$

3.3 Mathematical Properties

Theorem 3.1. *If the FK digraph contains a strongly connected component with only non-nullable edges, the instance is infeasible.*

Proof: No finite order permits insertions of records because each node is a precondition for all others in the cycle.
Cardinality: If c_{ab} is outside the allowed interval $[\ell, u]$, existence of the instance is impossible as some entity lacks mandatory connections.
Complexity: $O(|V| + |E|)$ for cycle detection; linear for counting.

4 Layer 3: Resource Capacity Bounds

4.1 Formal Statement

For each type r of fundamental resource (rooms, faculty hours, equipment, etc.), sum total demand and check against aggregate supply.

4.2 Algorithmic Model

Let D_r = total demand of resource r, S_r = supply.– Feasibility requires $D_r \leq S_r$ for all r.

4.3 Mathematical Properties

Theorem 4.1. *If there exists r such that $D_r > S_r$, the instance is infeasible.*

Proof: No assignment of events can be completed, as some demand cannot be assigned any available resource.
Detection: $O(N)$ linear in dataset size per resource type.

5 Layer 4: Temporal Window Analysis

5.1 Formal Statement

For each scheduling entity e (faculty, batch, course), verify that their total time demand (teaching hours, meetings, etc.) fits within their union of available timeslot windows.

5.2 Algorithmic Procedure

- For entity e, calculate d_e (hours required), a_e (availability, as time units).
- If $d_e > a_e$, infeasibility is immediate.

5.3 Mathematical Model

Demand > supply (pigeonhole principle): scheduling with d_e required events in a_e available slots is impossible if $d_e > a_e$.

5.4 Key Extensions

- Different window types: soft constraints (can be relaxed) vs hard constraints (cannot).
- For batches with only soft window conflicts, flag but allow to continue.

6 Layer 5: Competency, Eligibility, Availability

6.1 Formal Model

Construct bipartite graphs $G_F = (F, C, E_F)$ (faculty to courses), $G_R = (R, C, E_R)$ (rooms to courses).

6.2 Algorithmic Test

- For every course c, check $deg_{G_F}(c) > 0$ and $deg_{G_R}(c) > 0$
- Globally, check if matching size can cover C

6.3 Mathematical Properties

Theorem 6.1 (Hall's Theorem, necessary version)**.** *If for any subset $S \subseteq C$, $|N(S)| < |S|$ in either bipartite graph, then a matching does not exist, so instance is infeasible.*

Proof: Direct corollary of matching theory.
When caught: When the set of available/eligible resources is empty for any event.

7 Layer 6: Conflict Graph Sparsity and Chromatic Feasibility

7.1 Framework

Construct the conflict (incompatibility) graph G_C: vertices are event assignments (c, b), edges connect assignments in temporal conflict (shared batch, faculty, room).

7.2 Criteria

- Compute maximal degree Δ; if $\Delta + 1 > |T|$, not $|T|$-colorable $\rightarrow$ infeasible (Brook's theorem).
- Find cliques $K_{|T|+1}$ (fully connected $(|T|+1)$-sets); their existence proves infeasibility for $|T|$ slots.
- Analyze for chromatic number $\chi(G_C)$: infeasible if $\chi(G_C) > |T|$.

7.3 Mathematical Details

Lemma 7.1. *Any k-clique requires at least k distinct timeslots to schedule without conflict.*

Proof: Each vertex must be assigned a unique color.

7.4 Complexity Considerations

Clique and coloring detection is $\mathcal{NP}$-hard; maximum-degree/trivial clique-based checks are efficient and catch most practical infeasibilities in $O(n^2)$.

8 Layer 7: Global Constraint-Satisfaction and Propagation

8.1 Framework

Attempt constraint propagation in the reduced constraint system (unary, binary, n-ary) after all above layers.

8.2 Algorithmic Theoretical Definition

- Apply forward-checking, propagate all deducible implications.
- If any variable domain is empty during propagation, instance is infeasible.

8.3 Mathematical Proof

Arc-consistency (AC) preserves global feasibility: if propagation eliminates all possible values for a variable, the overall CSP has no solution.

8.4 Enhanced Propagation

Hyper-arc and higher-order constraint reasoning (e.g., violation projection for aggregates), as feasible for tractable subsets or resource capacity constraints on subsets.

9 Layer Interactions and Cross-Layer Factors

9.1 Aggregate Load Ratio

$$\rho = \frac{\sum_c h_c}{|T|} \quad]$$

If $\rho > |R|$, infeasibility follows immediately.

9.2 Window Tightness Index

$$\tau = \max_v \frac{d_v}{|W_v|} \;] Can predict tightness before chromatic or propagation checks.$$

9.3 Conflict Density

Proportion of possible assignment pairs that are conflicted: $\delta = \frac{|E_C|}{\binom{n}{2}}$.

10 Why Each Layer is Necessary

Each layer exploits a different abstraction:

- 1: Semantic validity of source data.
- 2: Relational semantics and linkage structure.
- 3: Resource contiguity/floor constraints.
- 4: Temporal fit for indivisible demand/supply instances.
- 5: Combinatorial eligibility (existence of qualified assignments).
- 6: Feasibility of colorability, a known hard core in scheduling.
- 7: Nonlocal inferencing (propagation) for configurations undetected by above.

11 Extensions and Further Refinement

Possible extensions include multi-resource generalized assignment checks, tight kernelization via advanced graph decomposition, and entropy-based infeasibility scoring for deeper analysis on borderline instances.

12 Conclusion

A rigorous, mathematically grounded seven-layer feasibility framework pre-empts the computational burden of NP optimization by probabilistically and structurally detecting most sources of infeasibility with $\mathcal{O}(n^2)$ (or better) complexity per layer. Each layer is optimal for its class of infeasibility, and only true hard instances pass to the solver. This modular approach is theoretically optimal and empirically effective for real-world educational scheduling.