Timeline Layout Algorithm: Complete Technical Specification

Antigravity Implementation Team

January 2026

Core Concept: "Tetris with Rubber Bands"
A constrained force-directed layout algorithm that arranges team nodes on horizontal "swim-lanes" while minimizing visual crossings and maintaining family proximity.

Contents

1 Fundamental Definitions

1.1 What is a Node?

A **Node** represents a single cycling team with properties defining its lifecycle and identity:

```
{
  id: "LPR",                    // Unique identifier
  founding_year: 2004,          // Year the team was founded
  dissolution_year: 2009,       // Year the team dissolved (or null if still active)
  eras: [                       // Array of yearly "slices"
    { year: 2004, name: "LPR Brakes" },
    { year: 2005, name: "LPR Brakes" },
    // ... one entry per year
  ]
}
```

Visual Representation: A node is drawn as a horizontal bar spanning from $founding\_year$ to $dissolution\_year + 1$ (inclusive rendering).

Example: LPR (2004–2009) visually occupies the space from year 2004 through the end of year 2009.

1.2 What is Temporal Overlap?

Two nodes **temporally overlap** if they exist at the same time. This is determined using **inclusive** year boundaries:

$$Overlap(A, B) \iff (A_{start} \leq B_{end}) \wedge (B_{start} \leq A_{end}) \tag{1}$$

Critical Edge Case - "Touching" Nodes:

- LPR ends 2009, Utensilnord starts 2010 $\implies$ **NO OVERLAP** ($2009 < 2010$)
- Sanson ends 1980, Famcucine starts 1980 $\implies$ **OVERLAP** ($1980 \leq 1980$)

Why this matters: Nodes that overlap cannot share the same horizontal lane (swimlane) because they would visually collide.

1.3 What is a Chain?

A **Chain** is a linear sequence of nodes connected by 1-to-1 relationships with **no temporal overlap**.

```
{
  id: "chain-0",
  nodes: [LPR, Utensilnord, Katusha],  // Array of node objects
  startTime: 2004,                      // founding_year of first node
  endTime: 2019,                        // dissolution_year of last node
  yIndex: 0                             // Current lane assignment
}
```

Chain Formation Rules:

1. **Linear Topology**: Each node has exactly 1 predecessor and 1 successor (except endpoints).
2. **No Temporal Overlap**: Consecutive nodes must NOT overlap.
3. **Break on Split/Merge**: If a node has multiple children or multiple parents, the chain breaks.

Example Chain: LPR (2004–2009) → Utensilnord (2010–2015) → Katusha (2016–2019)
These three nodes form ONE chain because they satisfy all rules and do not overlap ($2009 < 2010$ and $2015 < 2016$).

Counter-Example (Chain Break): Sanson (1963–1980) × Famcucine (1980–1981)
These form TWO separate chains because they overlap ($1980 \leq 1980$).

1.4 What is a Link?

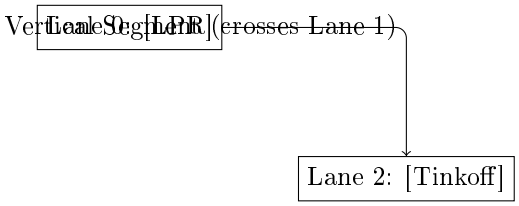
A **Link** represents a lineage connection between two teams:

```
{
  source: "LPR",            // Source node ID
  target: "Tinkoff",        // Target node ID
  type: "LEGAL_TRANSFER",   // Type of connection
  year: 2007                // Year the connection occurred
}
```

Visual Representation: Links are drawn as curved paths connecting the source node to the target node.

1.5 What is a Vertical Segment?

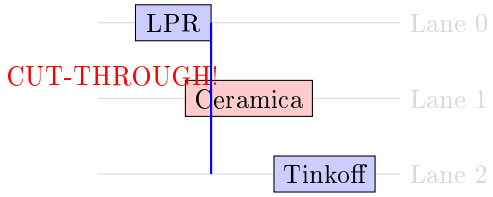
When a link connects nodes in different lanes, it creates a **Vertical Segment** - the portion of the link that crosses intermediate lanes.



Key Property: A vertical segment exists at a specific **year** (the link's year) and spans between two **lanes**.

1.6 What is a Cut-Through?

A **Cut-Through** occurs when a node sits in a lane that a vertical segment passes through.



Detection Logic:

```
// For a node in lane Y, check all vertical segments
isCutThrough = (
  Y > segment.y1 &&            // Node is between the two lanes
  Y < segment.y2 &&
  segment.time >= node.start && // Link occurs during node's lifetime
  segment.time <= node.end + 1  // +1 accounts for inclusive rendering
)
```

Why +1? Since nodes render inclusively to $dissolution\_year + 1$, a node ending in 2009 visually extends to the start of 2010. A link at year 2010 should detect a cut-through with this node.

1.7 What is a Blocker?

A **Blocker** occurs when a node sits on top of a vertical segment, blocking the visual "corridor" between parent and child.

Key Difference from Cut-Through:

- **Cut-Through**: The link slices through the node (node is the victim).
- **Blocker**: The node blocks someone else's link (node is the perpetrator).

1.8 What is Lane Sharing?

Lane Sharing occurs when multiple chains occupy the same horizontal lane.

Strict Stranger Rule: Unrelated chains ("strangers") are **strictly forbidden** from sharing a lane unless there is at least a 1-year gap between them. Since nodes render to $dissolution\_year + 1$:

- Node A ends in 2009 (renders to 2010)
- Node B starts in 2011
- Gap = 2011 - 2010 = 1 year ✓ (allowed)

Family Exception: Parent and child chains can share lanes with temporal overlap.

Distance Decay (Currently Disabled): The penalty formula remains but with weight = 0:

$$Penalty = \frac{W_{SHARE}}{\max(0.5, \Delta T)} \tag{2}$$

2 Math Symbols & Variables Reference

To clarify the formulas used in the optimization phases, here is a reference of the symbols and variables:

Symbol	Name	Description
Y	**Lane Index**	The vertical coordinate of a chain (0 = top).
T	**Time**	An integer year (e.g., 2007).
$\mu_{parents}$	**Mean Parent Lane**	Average Y position of all parent chains connected to current chain.
$\mu_{children}$	**Mean Child Lane**	Average Y position of all child chains connected to current chain.
ΔT	**Temporal Gap**	Years between two nodes. Positive = gap, negative = overlap.
J	**Total Cost**	The "energy" of a specific lane assignment to be minimized.
W_{XXX}	**Weights**	Penalty multipliers defining constraint importance.
P	**Parent Lane**	The Y coordinate of an individual parent chain.
C	**Child Lane**	The Y coordinate of an individual child chain.

Table 1: Math Symbols and Variables

3 Algorithm Overview

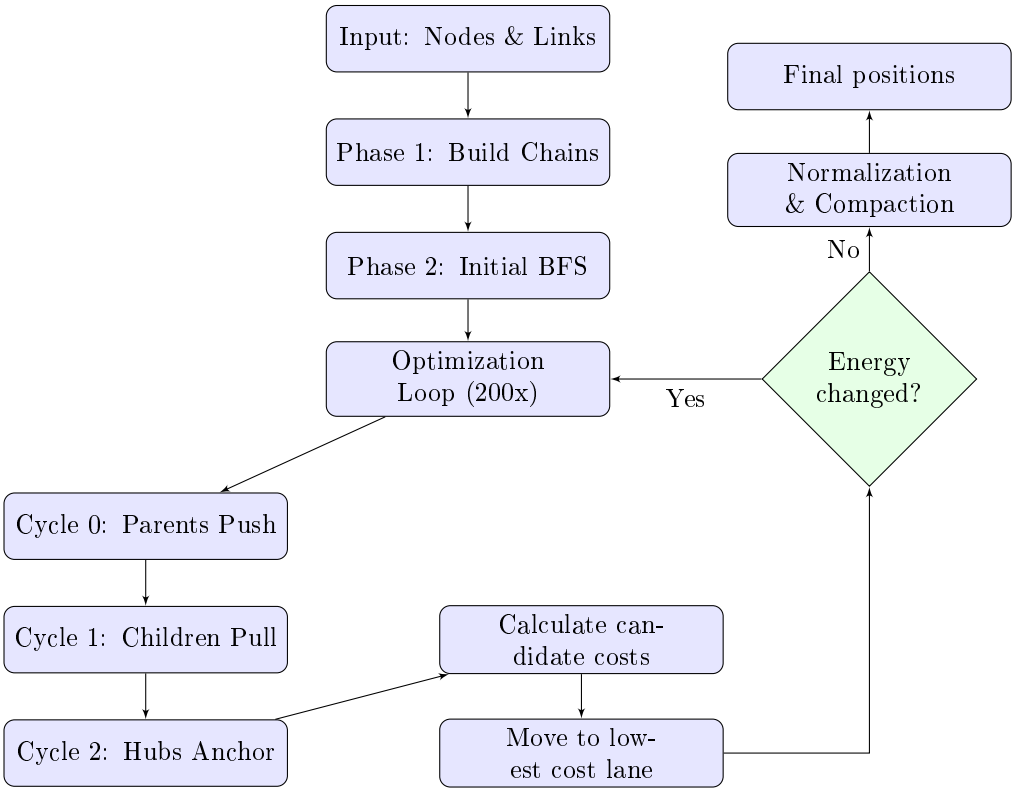


Figure 1: Layout Algorithm Pipeline

4 Phase 1: Chain Decomposition

Goal: Group nodes into rigid linear units that move together.
Algorithm:

1. Build predecessor/successor maps from links.
2. For each unvisited node:
   (a) If it's a chain start (0 or > 1 predecessors):
   - Walk forward following single successors.
   - Stop if: no successor, > 1 successors, or temporal overlap.
   - Create chain from collected nodes.

   (b) Mark all nodes in chain as visited.

5 Phase 2: Initial Placement

Goal: Create a starting layout using Breadth-First Search (BFS). Root chains are placed first (lane 0), then descendants are searched for the first available non-overlapping lane.

6 Phase 3: Optimization Loop

6.1 Iteration Strategy (Tri-State Cycle)

The loop alternates between three sorting strategies every iteration:

- **Cycle 0 - Parents Push**: Sort by *startTime* ASC.

- **Cycle 1 - Children Pull**: Sort by *startTime* DESC.
- **Cycle 2 - Hubs Anchor**: Sort by *degree* DESC.

6.2 Candidate Search Strategy

For each chain at current lane Y, the algorithm searches:

1. **Local Neighborhood**: $Y \pm 50$ lanes.
2. **Parent Vicinity**: $P \pm 10$ lanes.
3. **Child Vicinity**: $C \pm 10$ lanes.

7 Cost Function Details

The total cost J for a chain considering lane Y is:

$$J = C_{ATTR} + C_{CUT} + C_{BLOCK} + C_{SHARE} + C_{YSHAPE} \tag{3}$$

7.1 Attraction Cost (C_{ATTR})

Formula: $C_{ATTR} = W_{ATTR} \times (|Y - \mu_{parents}|^2 + |Y - \mu_{children}|^2)$
Weight: 100
Example: Chain at $Y = 5$ with parents at average lane 3 and children at average lane 8.
$C_{ATTR} = 100 \times ((5-3)^2 + (5-8)^2) = 100 \times (4+9) = 1,300$.

7.2 Cut-Through Cost (C_{CUT})

Formula: $C_{CUT} = W_{CUT} \times$ (Number of cuts)
Weight: 10,000
Example: Chain Ceramica [2005–2010] at $Y = 1$. Segment LPR(0) $\rightarrow$ Tinkoff(2) at 2007.
$Y = 1$ is between 0 and 2; 2007 is within [2005, 2010] $\implies$ 1 cut.
$C_{CUT} = 10,000 \times 1 = 10,000$.

7.3 Blocker Cost (C_{BLOCK})

Formula: $C_{BLOCK} = W_{BLOCK} \times$ (Number of segments blocked)
Weight: 5,000

7.4 Lane Sharing Cost (C_{SHARE})

Formula: $C_{SHARE} = W_{SHARE} / \max(0.5, \Delta T)$
Weight: 0 (disabled)
Strict Rule: Strangers must have $\geq$ 1-year gap or lane is forbidden. Family can overlap.

7.5 Y-Shape Symmetry Cost (C_{YSHAPE})

Formula: $C_{YSHAPE} = W_{YSHAPE} \times$ (Number of violations)
Weight: 150

8 Normalization & Compaction

1. **Normalization**: $Y_{final} = Y - Y_{min}$.
2. **Compaction**: Shift lanes to remove empty horizontal gaps.

9 Configuration Parameters

```
export const LAYOUT_CONFIG = {
  ITERATIONS: { MIN: 50, MAX: 500, MULTIPLIER: 10 },
  SEARCH_RADIUS: 50,
  TARGET_RADIUS: 10,
  WEIGHTS: {
    ATTRACTION: 100.0,
    CUT_THROUGH: 10000.0,
    BLOCKER: 5000.0,
    LANE_SHARING: 0.0,      // Disabled (strict collision handles strangers)
    Y_SHAPE: 150.0
  }
};
```

10 Known Limitations

- **Local Optimization**: Greedy nature can lead to jitter or local minima.
- **Asymmetric Penalties**: Mover pays the cost, potentially causing chains to "flee" each other.
- **Temporal Overlap Edge Cases**: Inclusive comparison (>=) prevents touch points.

11 Debugging Tips

- **Visualizing Costs**: Log output of `calculateCost` for specific node IDs.
- **Tracing Moves**: Log source and target lanes during optimization iterations.

12 Glossary

- **Node**: A single team with founding/dissolution years.
- **Chain**: Linear sequence of nodes with no temporal overlap.
- **Lane**: Horizontal swimlane (Y-coordinate).
- **Link**: Connection between two nodes (parent→child).
- **Vertical Segment**: Portion of a link crossing intermediate lanes.
- **Cut-Through**: Node sitting in a lane crossed by another family's link.
- **Blocker**: Node blocks another family's vertical link.