HME
Harris Matrix Editor

Version 1.0

Technical Documentation

A comprehensive web-based tool for creating, editing, and visualizing archaeological stratigraphic relationships using the Harris Matrix method

Documentation generated by Claude (Anthropic)
in collaboration with Archaeotux
February 2026

Table of Contents

1. Introduction

The Harris Matrix Editor (HME) is a comprehensive web-based application designed for archaeologists to create, edit, and visualize stratigraphic relationships following the Harris Matrix methodology developed by Dr. Edward C. Harris in 1973. The Harris Matrix is the standard method for recording and analyzing archaeological stratigraphy, representing the temporal sequence of archaeological deposits and interfaces.

HME provides a modern, intuitive interface that replaces traditional paper-based matrix creation with a digital workflow. The application supports the complete archaeological documentation workflow, from initial data entry through validation, visualization, and export for publication or integration with Geographic Information Systems (GIS).

Key Features

- **Stratigraphic Unit Management:** Create and manage stratigraphic units (SU) with five distinct types: Layer, Deposit, Fill, Structure, and Interface
- **Relationship Editing:** Establish stratigraphic relationships through intuitive drag-and-drop connections
- **Phase Management:** Organize units into chronological phases with customizable colors and ordering
- **Object Grouping:** Group related units into archaeological features with convex hull visualization
- **Intelligent Auto-Layout:** Automatic arrangement optimizing visual clarity while respecting stratigraphic principles
- **Comprehensive Validation:** Detect logical errors, phase inconsistencies, and data quality issues
- **Open File Format Support:** Full interoperability with JSON, GraphML, GeoJSON, and SVG formats

2. Open File Formats

HME is built on a foundation of open, well-documented file formats to ensure long-term data preservation and interoperability with other archaeological software systems. All formats are human-readable text formats that can be inspected, edited, and processed with standard tools.

2.1 JSON Format (Save/Load)

The native HME file format uses JSON (JavaScript Object Notation), an industry-standard data interchange format. JSON files preserve all matrix data including node positions, allowing projects to be saved and restored exactly as left.

JSON Structure

The JSON file contains four primary arrays within a versioned container:

Field	Description
version	File format version (currently '1.0')
exportDate	ISO 8601 timestamp of when the file was saved
nodes	Array of stratigraphic units with id, label, type, description, phase, x, y coordinates, and optional geometry
edges	Array of relationships with id, source, and target node references
phases	Array of chronological phases with id, name, and color
objects	Array of feature groups with id, name, color, and nodeIds array

Node Types: Each node has a 'type' field with one of five values: 'layer', 'deposit', 'fill', 'structure', or 'interface'. These correspond to standard archaeological context classifications and determine visual rendering (rectangles for most types, ellipses for interfaces).

2.2 GraphML Format (Import/Export)

GraphML is an XML-based format for graph data, widely supported by graph visualization and analysis tools. HME's GraphML implementation includes yEd-specific extensions for full visual fidelity when opening exported files in yEd Graph Editor.

GraphML Features

- **Hierarchical Grouping:** Phases are represented as top-level group nodes, with objects as nested sub-groups
- **Visual Preservation:** Node shapes, colors, and positions are preserved using yFiles namespace extensions
- **Custom Attributes:** HME-specific metadata (hme_type, hme_description) stored as GraphML data keys
- **Roundtrip Support:** Files edited in yEd or other GraphML editors can be re-imported into HME

The GraphML export is compatible with yEd, Gephi, Cytoscape, and other graph analysis tools. This enables advanced network analysis of stratigraphic relationships beyond HME's built-in capabilities.

2.3 GeoJSON Format (QGIS Integration)

GeoJSON is the standard format for encoding geographic data structures, enabling seamless integration with Geographic Information Systems. HME supports bidirectional GeoJSON workflows: importing spatial data from QGIS and exporting enriched data back.

GeoJSON Import Workflow

When importing GeoJSON files from QGIS, HME performs intelligent attribute matching:

1. **Attribute Detection:** Automatically scans all feature properties to identify potential unit identifier fields
2. **Auto-Selection:** Prioritizes common field names (label, SU_NR, befund, unit, name, id) for automatic matching
3. **Preview:** Displays a preview table showing how GeoJSON values will match existing matrix units
4. **Flexible Matching:** Supports exact matches, prefix-tolerant matches (e.g., 'SU 001' matches '001'), and interface designation preservation
5. **Geometry Association:** Matched units receive polygon/point geometry for display in the integrated map panel

GeoJSON Export Properties

Exported GeoJSON files include comprehensive properties for each stratigraphic unit: id, label, description, type, phase information (id, name, color), object memberships, and stratigraphic relationships (relations_above, relations_below). This enables rich styling and analysis in GIS software.

2.4 SVG Format (Publication Export)

Scalable Vector Graphics (SVG) export produces publication-ready vector images suitable for print or digital publication. The exported SVG preserves all visual elements including phase strips, object hulls, node shapes, and edge routing.

SVG Export Characteristics

- **Vector Format:** Infinitely scalable without quality loss
- **Embedded Fonts:** Uses system fonts (Segoe UI, system-ui) with CSS styling
- **Automatic Bounding:** ViewBox calculated to fit all content with padding
- **Post-Processing Compatible:** Can be edited in Inkscape, Adobe Illustrator, or other vector editors

3. Auto-Layout Algorithm

The auto-layout system is one of HME's most sophisticated features, implementing a multi-pass algorithm that respects archaeological principles while minimizing visual clutter. The algorithm draws on established graph drawing techniques, adapted specifically for Harris Matrix requirements.

3.1 Overview and Design Principles

The layout algorithm is guided by several key principles derived from Harris Matrix conventions:

- **Superposition Principle:** Stratigraphically connected units should share vertical alignment (same X-position) where possible, reflecting their physical relationship in the ground
- **Temporal Ordering:** Younger units appear above older units, with phases organized from top (most recent) to bottom (oldest)
- **Object Coherence:** Units belonging to the same archaeological feature (object) should be grouped together visually
- **Edge Crossing Minimization:** Reduce visual complexity by minimizing edge intersections
- **Long Edge Routing:** Edges spanning multiple phases should be routed to the outer edges of the matrix to avoid cluttering the central area

3.2 Eight-Step Layout Process

The auto-layout algorithm executes in eight distinct steps, each building on the results of previous steps:

Step 1: Topological Ranking

Within each phase, nodes are assigned vertical ranks based on their stratigraphic relationships. Using a modified Kahn's algorithm, nodes with no incoming edges (within the phase) receive rank 0, and each subsequent level receives rank = max(parent ranks) + 1. This establishes the basic vertical structure.

Step 2: Column Structure Building

Nodes are organized into columns. Object groups form dedicated columns containing all their member nodes at their respective ranks. Loose nodes (not belonging to any object) each form individual columns. Columns are initially sorted by the minimum label of their contained nodes for reproducibility.

Step 3: Position Computation

Given a column ordering, horizontal positions are computed. Each column is assigned horizontal space based on the maximum width of nodes it contains across all ranks. Object columns receive additional gap spacing to ensure visual separation. Nodes within a column are centered within their allocated space.

Step 4: Long Edge Identification

Edges spanning two or more phases are identified as 'long edges.' The nodes at both ends of these edges are marked for special treatment during column ordering, as their placement significantly impacts the overall matrix readability.

Step 5: Median-Based Column Sorting

Columns are reordered based on the median X-position of their connected nodes in adjacent phases. This is the barycenter heuristic adapted for column-based layout. Columns with connections to nodes on the left side of the matrix move left; those with connections to the right move right.

Step 6: Long Edge Outer Routing

Columns containing nodes with long edges are moved to the outer positions (left or right) of their phase. The direction is determined by analyzing where the connected nodes are positioned: columns with connections predominantly on the left go to the left edge, and vice versa.

Step 7: Multi-Pass Iteration

Steps 5 and 6 are repeated in four passes, alternating between top-down (using parent positions) and bottom-up (using child positions) ordering. This iterative refinement progressively reduces edge crossings and improves vertical alignment of connected units.

Step 8: Final Position Computation

After all ordering optimizations, final X/Y coordinates are computed for each node. The matrix is centered horizontally, and the view is reset to show all content.

3.3 Barycenter Optimization

The barycenter (or median) heuristic is a well-established technique for edge crossing minimization in layered graph drawing. For each column, the algorithm computes the median of the X-positions of all nodes connected to that column's nodes. Columns are then sorted by this median value.

The key insight is that placing columns near their connected nodes' positions minimizes the horizontal distance edges must travel, which in turn reduces crossings. By alternating between using positions of parent nodes (top-down pass) and child nodes (bottom-up pass), the algorithm converges toward a locally optimal arrangement.

4. Validation System

HME includes a comprehensive validation system that checks the stratigraphic data for logical errors, inconsistencies, and potential problems. Validation results are categorized into three severity levels: errors (must be fixed), warnings (should be reviewed), and informational messages (suggestions for improvement).

4.1 Error Detection

Errors represent fundamental logical problems that violate the principles of stratigraphic recording. These must be addressed before the matrix can be considered valid.

Duplicate Labels

Each stratigraphic unit must have a unique identifier. The validation system performs case-insensitive comparison of all labels and reports any duplicates. This prevents confusion when referencing units in the matrix or in linked documentation.

Cycle Detection

A Harris Matrix must be a Directed Acyclic Graph (DAG). Cycles represent logical impossibilities – a unit cannot be both above and below another unit in the stratigraphic sequence. HME uses depth-first search (DFS) with three-color marking (white/gray/black) to detect cycles. When a cycle is found, the complete cycle path is reported (e.g., 'SU 001 → SU 002 → SU 003 → SU 001').

Phase Direction Consistency

When units are assigned to phases, their stratigraphic relationships must respect the phase ordering. A unit from an older phase cannot lie stratigraphically above a unit from a younger phase. The validation system checks each edge and reports violations with the specific phase names involved.

4.2 Warnings

Warnings indicate potential problems that may or may not require attention depending on the specific excavation context.

Phase-Skipping Edges

When an edge connects units in non-adjacent phases without any intermediate units in the skipped phases, this may indicate missing stratigraphy or incorrect phase assignments. The validation uses breadth-first search to check if any intermediate path exists through the skipped phases; if not, a warning is raised.

Isolated Units

Units with no stratigraphic relationships (neither above nor below any other unit) are flagged. In a complete matrix, every unit should be connected to at least one other unit. Isolated units may indicate incomplete data entry or units that were accidentally disconnected.

Unassigned Phases

Units without phase assignments are reported. While not strictly an error (units can exist without phases during initial data entry), assigned phases are essential for chronological analysis and proper auto-layout behavior.

4.3 Informational Messages

Informational messages highlight structural characteristics that may warrant review but are not necessarily problems.

Dangling Roots and Leaves

Units that have no incoming edges (roots) should typically be in the youngest phase, and units with no outgoing edges (leaves) should be in the oldest phase. Units that violate this expectation are flagged as potentially having missing relationships.

Redundant (Transitive) Edges

An edge is redundant if the same relationship is already implied through other paths. For example, if A→B and B→C exist, then A→C is transitively implied. The validation computes transitive reachability and reports edges that could be removed without losing information. Note that some archaeologists prefer to include these edges for clarity; thus this is informational only.

5. AutoSave System

HME version 1.0 introduces an AutoSave system to prevent data loss from unexpected events such as browser crashes, accidental tab closure, or power failures. The system operates transparently in the background while providing user control and feedback.

AutoSave Behavior

- **Automatic Interval:** Data is automatically saved to browser localStorage every 30 seconds
- **Before Unload Protection:** When closing or refreshing the browser, a confirmation dialog appears if unsaved data exists
- **Session Recovery:** On application load, users are prompted to restore previously autosaved data
- **Visual Feedback:** A brief '✓ Saved' indicator appears in the toolbar after each autosave
- **Toggle Control:** AutoSave can be enabled or disabled via a toolbar button

Storage Details

AutoSave data is stored in browser localStorage under the key 'hme_autosave'. This storage is domain- and port-specific, meaning data saved when running on localhost:3000 is separate from localhost:8000. The storage limit is typically 5-10 MB depending on the browser, which is sufficient for matrices with hundreds of units.

Important: AutoSave is designed as a safety net, not a replacement for explicit saving. Users should still regularly export their work to JSON files for permanent storage and backup purposes.

6. Technical Specifications

Specification	Value
Application Type	Single-Page Application (SPA)
Framework	React 18 with Hooks
Build Tool	Vite 5
Styling	Tailwind CSS 3
Source Code	~4,500 lines (single App.jsx file)
Browser Requirements	Modern browsers: Chrome, Firefox, Safari, Edge
Node.js Version	18.x LTS or later recommended
License	MIT License

— End of Documentation —