Article

NetGen 4.0: The browser-based tool for network visualization

WenJun Zhang
School of Life Sciences, Sun Yat-sen University, Guangzhou 510275, China; International Academy of Ecology and Environmental Sciences, Hong Kong
E-mail: wjzhang@iaees.org, zhwj@mail.sysu.edu.cn

Received 30 June 2026; *Accepted 12 July 2026*; *Published online 30 July 2026*; *Published 1 June 2027*

Abstract
Complex network visualization plays a crucial role in network science research, facilitating the understanding of topological structures and the discovery of underlying patterns. This paper presents the design and implementation of a browser-based interactive network visualization tool, aiming to provide researchers with a convenient and efficient platform for network analysis and presentation. The proposed tool utilizes HTML5 Canvas technology for graphical rendering and supports multiple common data formats including TXT, CSV, XLS, XLSX, and JSON. A key feature is its ability to automatically distinguish between weighted networks (four-column data) and unweighted networks (three-column data), enabling intelligent data parsing without manual configuration. The system incorporates six classical layout algorithms: force-directed layout, circular layout, hierarchical layout, grid layout, radial layout, and spectral layout, allowing users to select the optimal visualization scheme based on specific network characteristics. Regarding interaction design, the tool provides node dragging, hover information tooltips, and real-time parameter adjustment capabilities. It supports the visualization of both directed and undirected edges, with edge thickness representing weight information intuitively. Additionally, the system offers image export functionality, enabling users to save visualization results in PNG format for academic publications. The proposed tool will effectively handle small to medium-scale network data with stable algorithm performance and clear visualization output, exhibiting practical value and promising application prospects.

Keywords network visualization; force-directed layout; graph drawing algorithm; complex network analysis; interactive visualization; browser-based application; data format parsing.

Network Biology
ISSN 2220-8879
URL: http://www.iaees.org/publications/journals/nb/online-version.asp
RSS: http://www.iaees.org/publications/journals/nb/rss.xml
E-mail: networkbiology@iaees.org
Editor-in-Chief: WenJun Zhang
Publisher: International Academy of Ecology and Environmental Sciences

1 Introduction

Networks have become a universal language for representing complex relational data across virtually every scientific domain. From protein–protein interaction networks in systems biology and neuronal connectivity in

neuroscience, to social graphs in sociology, food webs in ecology, and citation networks in bibliometrics, the ability to visualise and interactively explore these structures is indispensable. Effective network visualisation not only facilitates pattern recognition and hypothesis generation but also serves as a powerful medium for communicating structural and functional insights to diverse audiences. As the volume and complexity of relational data continue to grow, the demand for intuitive, flexible, and accessible visualisation tools has never been greater.

Over the past two decades, a rich ecosystem of network visualisation software has been developed. Among desktop applications, Cytoscape (Shannon et al., 2003) has become the gold standard for integrating network visualisation with attribute data, particularly in bioinformatics and systems biology, offering extensive plugin support and advanced analytical capabilities. Gephi (Bastian et al., 2009) provides an interactive exploration environment with a wide array of layout algorithms, filtering, and clustering functions, making it popular in social network analysis and digital humanities. For programmatic workflows, NetworkX (Hagberg et al., 2008) offers a comprehensive Python library for network creation, manipulation, and study, while igraph (Csárdi and Nepusz, 2006) delivers high-performance graph operations across multiple programming languages, particularly suited for large-scale networks. In the web ecosystem, Cytoscape.js (Franz et al., 2015) enables rich, interactive graph visualisation directly in browsers, and the vis.js library (Perrone et al., 2020) provides a flexible, lightweight framework for dynamic network rendering. More recently, Gephi Lite (Girard et al., 2025) has emerged as a browser-based, installation-free version of Gephi, lowering the entry barrier for casual users.
Despite these advances, significant gaps remain. Desktop tools like Cytoscape and Gephi often demand substantial memory and processing power, and their installation and configuration can be daunting for non-specialists. Browser-based (or web-based) solutions, while more accessible, may sacrifice functionality, performance, or customisation depth. Moreover, the degree of user control over visual attributes,such as node size and colour, link width, length, and colour, node label appearance, and canvas dimensions, varies widely across platforms, and few tools offer both fully automatic layout algorithms and manual (drag-and-drop) arrangement in an integrated, user-friendly manner. Additionally, support for both weighted and unweighted, directed and undirected networks is not universally available, and many tools lack straightforward import from simple text files without requiring complex data preprocessing.

Recognising these limitations, a series of progressively refined tools has been developed. An early Java-based graph drawing utility (Zhang, 2012) established a foundational framework for rendering graphs. Subsequently, a web-based interactive network generator (Zhang, 2021) expanded accessibility to browser environments, enabling users to create interactive visualisations without local installation. A Matlab implementation (Zhang, 2024a) followed, providing an alternative platform for researchers accustomed to the Matlab ecosystem. To further streamline the workflow, a standalone executable software (Zhang, 2024b) eliminated dependency on any programming environment, and an executable Java version (Zhang, 2024c) offered a lightweight, cross-platform solution.

netGen 3.0 (Zhang, 2024d) is an executable Java software for network visualisation that incorporates substantial improvements over its predecessors. It allows users to generate and interactively visualise a network directly from a plain-text file, supporting both unweighted and weighted edges as well as undirected and directed graphs. Users can flexibly specify node size and colour, link width, length, and colour, node name colour, and canvas size. Furthermore, the software offers two complementary layout modes, artificial (manual) arrangement, which gives users full control over node positioning, and automatic layout, which computes optimal placements using built-in algorithms. This dual-mode design caters to diverse visualisation needs, from exploratory analysis to publication-quality figure preparation. Both the executable software and demonstration data files are provided to ensure immediate usability.

Building upon netGen 3.0 (Zhang, 2024d), NetGen 4.0, developed in present study, is a browser-based network visualization tool packaged as a self-contained HTML document and built using HTML5, CSS, and JavaScript. It allows users to import, generate, explore, style, and export node-link network diagrams without requiring a backend server.

2 Network Visualization Tool NetGen 4.0: Technical Report and User Guide

2.1 Summary

Building on the foundation laid by its predecessor netGen 3.0 (Zhang, 2024d), NetGen 4.0 is a fully self-contained, browser-based network visualisation tool implemented as a single HTML document using HTML5, CSS, and JavaScript. It enables users to import, generate, explore, customise, and export node-link network diagrams entirely client-side, with no backend server or external dependencies.

The application supports:

- Directed and undirected networks
- Weighted and unweighted edges
- CSV, TXT, Excel, and JSON input
- Force-directed, circular, and grid layouts
- Interactive zooming, panning, node dragging, searching, and highlighting
- Network statistics such as node count, edge count, average degree, and density
- Customizable node, edge, label, and canvas appearance
- PNG image export
- Built-in example networks, including Zachary's Karate Club network

The visualization is rendered directly on an HTML5 <canvas>. A custom force-directed algorithm computes node positions using repulsive, spring, centering, and damping forces.

2.2 Purpose and Intended Use

The main purpose of NetGen 4.0 is to provide an accessible visual environment for examining relationships among entities. A network consists of:

- **Nodes**, representing entities such as people, systems, documents, or organizations
- **Edges**, representing relationships, interactions, flows, or connections between entities

Typical applications include:

- Social network analysis
- Communication network inspection
- Citation and knowledge graph visualization
- Organizational relationship mapping
- Dependency analysis
- Transportation or infrastructure network exploration
- Classroom demonstrations of graph concepts
- Preliminary inspection of graph datasets

Because the application runs entirely in the browser, it is suitable for lightweight and local analysis. Imported data is processed on the client side and is not intentionally uploaded to a server by this code.

2.3 Technology Stack

2.3.1 Core technologies

The program uses:

- **HTML5** for application structure
- **CSS3** for layout, styling, gradients, responsive behavior, and animations

- **JavaScript** for file processing, graph construction, layout, interaction, statistics, and rendering
- **HTML5 Canvas API** for drawing nodes, edges, labels, arrowheads, and weights

2.3.2 External dependency

The only external JavaScript library is SheetJS:

html

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.18.5/xlsx.full.min.js"></script>
```

SheetJS is used to parse .xls and .xlsx files. CSV, TXT, and JSON files are handled through built-in browser APIs.

2.3.3 Deployment model

The application has no backend dependency. It can be:

1. Saved as an .html file
2. Opened directly in a modern browser
3. Hosted on any static web server

An internet connection is needed to load SheetJS from the CDN unless that library is downloaded and referenced locally.

2.4 User Interface Structure

The application is divided into two main areas.

2.4.1 Control panel

The left-hand control panel contains:

- Application title
- Node and edge counters
- Node search field
- Network type controls
- Layout controls
- Node appearance settings
- Edge appearance settings
- Force-layout parameters
- Canvas appearance settings
- File-opening and image-saving controls
- Animation and clearing controls
- Example-data buttons

The panel is scrollable when the available display height is insufficient.

2.4.2 Visualization area

The main area contains:

- The network canvas
- A network-information panel
- A floating visualization toolbar
- A node-detail popup
- Hover tooltips
- A loading overlay

The canvas occupies all remaining space after the control panel.

2.5 Principal Features

2.5.1 Multiple file formats

The application accepts:

- .csv
- .txt
- .xls
- .xlsx
- .json

Text files may use commas, tab characters, or semicolons as separators.

2.5.2 Directed and undirected display

Users may switch between:

- **Directed network**, with arrowheads
- **Undirected network**, without arrowheads

For directed data, the third column is interpreted as an edge-direction indicator:

- 1: source → target
- -1: target → source

Other values do not produce an arrowhead under the current drawing logic.

2.5.3 Weighted networks

A dataset is treated as weighted when the first processed row contains at least four fields and its fourth field can be parsed as a number.

Weights can be displayed around the midpoint of each edge.

2.5.4 Layout methods

Three layout algorithms are available:

1. **Force-directed layout**
2. **Circular layout**
3. **Grid layout**

The application also distinguishes between:

- **Automatic layout**, in which the force simulation may run continuously
- **Manual layout**, in which automatic force updates are disabled

2.5.5 Interactive navigation

Users can:

- Drag nodes
- Pan the canvas
- Zoom with the mouse wheel
- Zoom using toolbar buttons
- Fit the network to the screen
- Center the network
- Show or hide labels
- Show or hide edge weights

2.5.6 Search and neighborhood inspection

The search box highlights nodes whose labels contain the query.

Double-clicking a node highlights:

- The selected node
- Adjacent nodes
- Incident edges

Double-clicking empty canvas space removes the highlighting.

2.5.7 Statistics

The interface displays:

- Number of nodes
- Number of edges
- Network type
- Whether the network is weighted
- Average degree
- Network density

2.5.8 Visual customization

Users can adjust:

- Node size
- Node color
- Label color
- Label size
- Edge width
- Edge color
- Ideal edge length
- Background color
- Force parameters

2.5.9 Image export

The current canvas can be exported as a PNG image.

2.5.10 Built-in examples

The application provides:

- A random 10-node network
- A random 30-node network
- Zachary's Karate Club network

2.6 Internal Data Model

The graph is represented through three primary global variables:

javascript

```
let nodes = [];
let edges = [];
let nodeMap = new Map();
```

2.6.1 Node representation

Each node is created by createNode():

javascript

```
function createNode(label) {
    return {
        id: label,
        label: label,
        x: Math.random() * canvas.width,
        y: Math.random() * canvas.height,
        vx: 0,
        vy: 0,
```

```
        fx: null,
        fy: null,
        degree: 0,
        inDegree: 0,
        outDegree: 0,
        fixed: false,
        highlighted: false
    };
}
```

Important properties include:

Property	Meaning
id	Node identifier
label	Display label
x, y	Position in graph coordinates
vx, vy	Current velocity
fx, fy	Accumulated force
degree	Total incident-edge count
inDegree	Incoming-edge count
outDegree	Outgoing-edge count
fixed	Whether simulation movement is disabled
highlighted	Whether the node should be emphasized

The nodeMap provides efficient lookup by node label and prevents duplicate node objects.

2.6.2 Edge representation

An edge is stored as:

javascript

```
{
    source: nodeMap.get(source),
    target: nodeMap.get(target),
    type: type,
    weight: weight
}
```

The source and target fields reference node objects directly rather than storing only identifiers.

2.7 File Input Formats

2.7.1 CSV and TXT

A typical weighted directed file is:

text

```
Alice,Bob,1,2.5
Bob,Carol,1,1.2
Carol,Alice,-1,3.0
Alice,David,1,0.8
```

The expected columns are:

Column	Description
1	Source node
2	Target node
3	Direction/type indicator
4	Optional weight

The text reader accepts comma, tab, and semicolon delimiters:

javascript

```
const parts = line.split(/[,\t;]/).map(p => p.trim());
```

Only lines with at least three fields are accepted.

2.7.2 Excel

The program reads the first worksheet and converts it to an array of rows:

javascript

```
const workbook = XLSX.read(data, { type: 'array' });
const firstSheet = workbook.Sheets[workbook.SheetNames[0]];
const jsonData = XLSX.utils.sheet_to_json(firstSheet, { header: 1 });
```

The spreadsheet should follow the same column structure as a CSV file.

2.7.3 JSON

One supported JSON structure is:

json

```
{
  "edges": [
    {
      "source": "Alice",
      "target": "Bob",
      "type": 1,
      "weight": 2.5
    },
    {
      "source": "Bob",
      "target": "Carol",
      "type": 1,
      "weight": 1.2
    }
  ]
}
```

Alternative property names are accepted:

- from instead of source
- to instead of target

- direction instead of type
- value instead of weight

The application also accepts a JSON array compatible with its row-based input format:

json

```
[
  ["Alice", "Bob", 1, 2.5],
  ["Bob", "Carol", 1, 1.2]
]
```

2.7.4 Header handling

The code does not explicitly detect or remove CSV or Excel header rows. A header such as:

text

```
source,target,type,weight
```

may therefore be interpreted as graph data. Users should remove headers before import, or the program should be enhanced with header detection.

2.8 Data Processing Algorithm

The central data-processing function is processData().

Its workflow is:

1. Clear the existing graph
2. Detect whether the data appears weighted
3. Read each row
4. Convert source and target values to strings
5. Create missing nodes
6. Create an edge between the referenced node objects
7. Calculate node degrees
8. Apply an initial layout
9. Update statistics
10. Fit the graph to the screen

Representative code:

javascript

```
function processData(data) {
    nodes = [];
    edges = [];
    nodeMap.clear();

    isWeighted =
        data.length > 0 &&
        data[0].length >= 4 &&
        !isNaN(parseFloat(data[0][3]));

    for (const row of data) {
        const source = String(row[0]).trim();
        const target = String(row[1]).trim();
```

```
            const type = parseInt(row[2]) || 1;
            const weight = isWeighted ? parseFloat(row[3]) || 1 : 1;

            if (!nodeMap.has(source)) {
                const node = createNode(source);
                nodes.push(node);
                nodeMap.set(source, node);
            }

            if (!nodeMap.has(target)) {
                const node = createNode(target);
                nodes.push(node);
                nodeMap.set(target, node);
            }

            edges.push({
                source: nodeMap.get(source),
                target: nodeMap.get(target),
                type,
                weight
            });
        }

        calculateDegrees();
        // Apply selected layout and refresh the interface.
}
```

This approach automatically derives the node set from the edge list.

2.9 Layout Algorithms

2.9.1 Random initialization

The force-directed layout begins with random positions:

javascript

```
function applyRandomLayout() {
    const padding = 100;

    nodes.forEach(node => {
        node.x = padding + Math.random() * (canvas.width - 2 * padding);
        node.y = padding + Math.random() * (canvas.height - 2 * padding);
        node.vx = 0;
        node.vy = 0;
    });
}
```

Random initialization reduces complete overlap and gives the simulation a starting configuration.

2.9.2 Circular layout

Nodes are evenly distributed around a circle.

For node i among n nodes:

$\theta_i=2\pi i/n$

$x_i=c_x+r\cos(\theta_i)$

$y_i=c_y+r\sin(\theta_i)$

Representative code:

javascript

```
function applyCircleLayout() {
    const centerX = canvas.width / 2;
    const centerY = canvas.height / 2;
    const radius = Math.min(canvas.width, canvas.height) * 0.35;

    nodes.forEach((node, i) => {
        const angle = (2 * Math.PI * i) / nodes.length;
        node.x = centerX + radius * Math.cos(angle);
        node.y = centerY + radius * Math.sin(angle);
        node.vx = 0;
        node.vy = 0;
    });
}
```

This layout is deterministic with respect to node ordering and works well for displaying overall connectivity without simulation.

2.9.3 Grid layout

The grid layout calculates an approximately square number of columns:

$columns=\lceil\sqrt{n}\rceil$

The number of rows is:

$rows=\lceil n/columns\rceil$

Nodes are assigned cells according to their array index.

javascript

```
function applyGridLayout() {
    const cols = Math.ceil(Math.sqrt(nodes.length));
    const rows = Math.ceil(nodes.length / cols);
    const cellWidth = (canvas.width - 200) / cols;
    const cellHeight = (canvas.height - 200) / rows;

    nodes.forEach((node, i) => {
        const col = i % cols;
        const row = Math.floor(i / cols);

        node.x = 100 + col * cellWidth + cellWidth / 2;
        node.y = 100 + row * cellHeight + cellHeight / 2;
        node.vx = 0;
```

```
        node.vy = 0;
    });
}
```

This layout is useful when uniform spacing and predictable placement are preferred.

2.10 Force-Directed Layout Algorithm

The force-directed layout models the graph as a physical system. Nodes repel one another, while edges act as springs.

The implementation includes:

1. Pairwise repulsion
2. Edge attraction
3. Centering force
4. Velocity damping
5. Maximum-speed control
6. Boundary constraints

2.10.1 Force initialization

At every simulation frame, accumulated force is reset:

javascript

```
nodes.forEach(node => {
    node.fx = 0;
    node.fy = 0;
});
```

2.10.2 Pairwise repulsion

For every pair of nodes, the displacement is calculated:

$\Delta x = x_j - x_i$

$\Delta y = y_j - y_i$

$d = \sqrt{(\Delta x)^2 + (\Delta y)^2}$

The repulsive-force magnitude is:

$F_r = K_r / d^2$

where K_r is the configurable repulsion strength.

The force components are:

$F_x = \Delta x F_r / d$

$F_y = \Delta y F_r / d$

Equal and opposite forces are applied to the two nodes.

javascript

```
const force = config.repulsion / (dist * dist);
const fx = (dx / dist) * force;
const fy = (dy / dist) * force;

nodes[i].fx -= fx;
nodes[i].fy -= fy;
nodes[j].fx += fx;
nodes[j].fy += fy;
```

Complexity

Because every node pair is evaluated, this stage has time complexity:

$O(n^2)$

The source-code comment mentions Barnes–Hut approximation, but no quadtree or Barnes–Hut implementation is actually present. The current implementation is exact pairwise repulsion.

2.10.3 Edge attraction

Each edge behaves like a spring. The displacement from the ideal length is:

$\Delta d = d - L$

where L is config.linkLength.

The spring-force magnitude is:

$F_a = K_a(d - L)$

where K_a is the attraction coefficient.

javascript

```
const displacement = dist - config.linkLength;
const force = config.attraction * displacement;
const fx = (dx / dist) * force;
const fy = (dy / dist) * force;

edge.source.fx += fx;
edge.source.fy += fy;
edge.target.fx -= fx;
edge.target.fy -= fy;
```

This stage has complexity:

$O(m)$

where m is the number of edges.

2.10.4 Centering force

A weak force pulls nodes toward the center of the canvas:

javascript

```
node.fx += (centerX - node.x) * centerForce;
node.fy += (centerY - node.y) * centerForce;
```

This prevents disconnected components and weakly connected nodes from drifting indefinitely.

2.10.5 Velocity integration and damping

Velocity is updated from force and then multiplied by the damping coefficient:

javascript

```
node.vx = (node.vx + node.fx) * config.damping;
node.vy = (node.vy + node.fy) * config.damping;
```

Damping reduces oscillation. A coefficient close to 1 preserves motion longer, while a lower coefficient slows the system more quickly.

2.10.6 Speed limiting

The velocity magnitude is capped at 10:

```
javascript
const maxSpeed = 10;
const speed = Math.sqrt(node.vx * node.vx + node.vy * node.vy);

if (speed > maxSpeed) {
    node.vx = (node.vx / speed) * maxSpeed;
    node.vy = (node.vy / speed) * maxSpeed;
}
```

This reduces instability when forces become large.

2.10.7 Position update and boundaries

The new position is:

$x \leftarrow x + v_x$

$y \leftarrow y + v_y$

Nodes are then constrained to remain inside the canvas.

2.10.8 Overall complexity

Each force-layout iteration has approximate complexity:

$O(n^2+m)$

For small and medium networks, this is practical. For large networks, the pairwise repulsion stage is the principal bottleneck.

2.11 Rendering Algorithm

The renderer clears the canvas, applies the current camera transformation, draws edges, and then draws nodes.

```
javascript
function draw() {
    ctx.fillStyle = config.bgColor;
    ctx.fillRect(0, 0, canvas.width, canvas.height);

    ctx.save();
    ctx.translate(offsetX, offsetY);
    ctx.scale(scale, scale);

    drawEdges();
    drawNodes();

    ctx.restore();
}
```

Drawing edges before nodes allows nodes to appear above the lines.

2.11.1 Camera transformation

The view is controlled by:

- scale
- offsetX
- offsetY

World coordinates are converted to screen coordinates using:

$x_s = x_w \cdot scale + offsetX$

$y_s = y_w \cdot scale + offsetY$

The inverse conversion is:

$x_w = (x_s - offsetX)/scale$

$y_w = (y_s - offsetY)/scale$

Representative code:

```
javascript
function screenToWorld(x, y) {
    return {
        x: (x - offsetX) / scale,
        y: (y - offsetY) / scale
    };
}
```

This transformation is important for correctly selecting and dragging nodes while zoomed or panned.

2.11.2 Edge drawing

Each edge is rendered as a straight line from source to target.

If the network is directed, arrowheads are added according to the edge type. If the network is weighted and weight display is enabled, the weight is drawn near the line midpoint.

2.11.3 Arrowhead drawing

The arrow angle is obtained with:

```
javascript
const angle = Math.atan2(dy, dx);
```

The endpoint is moved backward from the node center so that the arrow touches the node boundary rather than extending through the node.

2.11.4 Node drawing

Nodes are rendered as circles. Highlighted or hovered nodes receive an outer glow. Node colors vary according to state:

- Red for the currently selected node
- Yellow for highlighted nodes
- Configured node color otherwise

Labels are drawn below each node.

2.12 Animation Loop

The application uses requestAnimationFrame():

```
javascript
function animate() {
    const layoutType =
        document.querySelector('input[name="layoutType"]:checked')?.value || 'auto';

    const algorithm =
        document.querySelector('input[name="layoutAlgorithm"]:checked')?.value || 'force';

    if (layoutType === 'auto' && algorithm === 'force') {
```

```
        applyForceLayout();
    }

    draw();
    animationId = requestAnimationFrame(animate);
}
```

This synchronizes drawing with browser rendering and normally produces smooth interaction.

The **Pause** button changes animationRunning, which stops force calculations. However, rendering through requestAnimationFrame() continues. This is useful because panning, zooming, highlighting, and styling still update visually while the physical layout is paused.

2.13 Interaction Design

2.13.1 Node dragging

When the user presses the left mouse button over a node:

- The node becomes selected
- Dragging mode starts
- A detail popup appears

As the mouse moves, the selected node's position is updated in world coordinates, and its velocity is reset.

2.13.2 Canvas panning

When the user presses the left mouse button on empty space:

- Panning mode begins
- Mouse movement modifies offsetX and offsetY

2.13.3 Zooming

The mouse wheel changes the scale:

javascript

```
const zoomFactor = e.deltaY > 0 ? 0.9 : 1.1;
scale *= zoomFactor;
scale = Math.max(0.1, Math.min(5, scale));
```

The implementation also adjusts offsets to keep the position under the pointer approximately fixed during zooming.

2.13.4 Hover inspection

When the pointer moves over a node, the application displays:

- Node label
- Degree
- In-degree
- Out-degree

The cursor changes to indicate that the node is interactive.

2.13.5 Double-click neighborhood highlighting

The function below highlights incident edges and their endpoint nodes:

javascript

```
function highlightNeighbors(node) {
    clearHighlights();
    node.highlighted = true;
```

```
    edges.forEach(edge => {
        if (edge.source === node || edge.target === node) {
            edge.highlighted = true;
            edge.source.highlighted = true;
            edge.target.highlighted = true;
        }
    });
}
```

This treats all incident edges as neighbors, regardless of direction.

2.13.6 Search

Search is case-insensitive and uses substring matching:

```
javascript
if (node.label.toLowerCase().includes(query)) {
    node.highlighted = true;
}
```

All matching nodes are highlighted.

2.14. Network Statistics

2.14.1 Node and edge counts

The counts are taken directly from the array lengths:

```
javascript
nodes.length
edges.length
```

2.14.2 Degree calculation

For each edge:

```
javascript
edge.source.outDegree++;
edge.target.inDegree++;
edge.source.degree++;
edge.target.degree++;
```

Thus, total degree counts each edge once at each endpoint.

2.14.3 Average degree

The displayed average degree is calculated as:

$\bar{k}=2m/n$

This is standard for undirected graphs and represents average total incidence. In a directed graph, average in-degree and average out-degree are each m/n, while average total degree is $2m/n$. Therefore, the displayed value should be interpreted as average total degree for directed data.

2.14.4 Density

For a directed network:

$D = m/n(n-1)$

For an undirected network:

$D = m/[n(n-1)/2]$

Representative code:

javascript

```
const maxEdges = isDirected
      ? nodes.length * (nodes.length - 1)
      : nodes.length * (nodes.length - 1) / 2;

const density = (edges.length / maxEdges).toFixed(4);
```

This formula assumes a simple graph without self-loops or duplicate edges. If such edges are present, the value may not have the intended interpretation and can potentially exceed one.

2.15 Configuration Parameters

The visual and physical settings are stored in one object:

javascript

```
let config = {
      nodeSize: 20,
      nodeColor: '#00d4ff',
      labelColor: '#ffffff',
      labelSize: 12,
      linkWidth: 2,
      linkColor: '#4a90d9',
      linkLength: 150,
      repulsion: 1000,
      attraction: 0.01,
      damping: 0.9,
      bgColor: '#0a0a0f'
};
```

Parameter interpretation

Parameter	Description
nodeSize	Node diameter used by the drawing code
nodeColor	Default fill color
labelColor	Node-label and weight color
labelSize	Label font size
linkWidth	Edge width
linkColor	Default edge and arrow color
linkLength	Desired spring length
repulsion	Node-to-node repulsive strength
attraction	Edge spring strength
damping	Velocity retention coefficient

Parameter	Description
bgColor	Canvas background

The sliders use transforms for attraction and damping:

javascript
transform: v => v / 1000

and:

javascript
transform: v => v / 100

This permits convenient integer slider values while maintaining fractional physical parameters.

2.16 User Guide

2.16.1 Starting the application

1. Save the code as an HTML file, for example netVisual-Improved.html.
2. Open it in a current browser such as Chrome, Edge, Firefox, or Safari.
3. Ensure internet access if Excel support is required through the external SheetJS CDN.

If the supplied file has a .txt extension, rename it to .html before opening it as a web page.

2.16.2 Loading a network file

1. Click **Open File**.
2. Select a supported CSV, TXT, XLS, XLSX, or JSON file.
3. Wait for the loading overlay to disappear.
4. The network is parsed, laid out, and fitted to the available screen.

For reliable import:

- Use at least three columns
- Do not include a header row
- Ensure the fourth column is numeric if weights are required
- Use unique, meaningful node labels
- Avoid blank source or target values

2.16.3 Loading sample data

Choose one of the sample buttons:

- **Small Network (10 nodes)**
- **Medium Network (30 nodes)**
- **Karate Club**

The random examples are weighted and may differ each time they are generated.

2.16.4 Selecting network type

Use the **Network Type** section:

- Choose **Directed Network** to display arrowheads
- Choose **Undirected Network** to hide arrowheads

This selection changes visualization and density calculation. It does not reconstruct or reverse stored edges.

2.16.5 Selecting a layout

Force-directed

1. Choose **Automatic Layout**.
2. Select **Force Directed**.
3. Allow the nodes to move until the graph stabilizes.

4. Use **Pause** to stop physical movement when a satisfactory arrangement is reached.

Circular

Select **Circular** to place nodes evenly around a circle.

Grid

Select **Grid** to arrange nodes in rows and columns.

Manual mode

Choose **Manual Layout** to stop automatic force updates. Nodes can then be dragged manually.

2.16.6 Adjusting force parameters

- Increase **Repulsion Strength** to push nodes farther apart.
- Increase **Attraction Strength** to make connected nodes respond more strongly to their ideal edge length.
- Increase **Ideal Edge Length** to produce longer connections.
- Lower **Damping** to reduce oscillation more quickly.
- Raise **Damping** to preserve movement longer.

Large repulsion or attraction values may create unstable or rapidly moving layouts.

2.16.7 Navigating the network

- **Drag a node:** press and move the left mouse button over a node
- **Pan:** press and move the left mouse button over empty canvas
- **Zoom:** use the mouse wheel
- **Toolbar plus:** zoom in
- **Toolbar minus:** zoom out
- **Fit button:** fit all nodes into the viewport
- **Center button:** move the network centroid to the center

2.16.8 Inspecting nodes

- Hover over a node to view its degree information
- Click and hold a node to display the detail popup
- Double-click a node to highlight its neighbors and incident edges
- Double-click empty space to clear neighborhood highlighting

2.16.9 Searching for nodes

1. Enter text in the search field.
2. Matching node labels are highlighted.
3. Matching is case-insensitive and partial.
4. Clear the search field to remove search highlighting.

2.16.10 Showing labels and weights

Use the toolbar buttons:

- **Tag icon:** show or hide node labels
- **Scale icon:** show or hide edge weights

Weights are displayed only if the imported data was detected as weighted.

2.16.11 Styling the visualization

Use the controls to adjust:

- Node size and color
- Label color and size
- Edge width and color
- Background color

Changes are reflected in the continuously rendered canvas.

2.16.12 Saving an image

1. Arrange and style the graph.
2. Click **Save Image**.
3. The browser downloads network.png.

The exported image contains only the canvas. It does not include the control panel, toolbar, information panel, tooltips, or detail popup.

2.16.13 Clearing the network

Click **Clear** to remove all nodes and edges.

2.17 Representative JavaScript Components

2.17.1 File dispatch

javascript

```
switch (extension) {
    case 'csv':
    case 'txt':
        data = await readTextFile(file);
        break;
    case 'xls':
    case 'xlsx':
        data = await readExcelFile(file);
        break;
    case 'json':
        data = await readJSONFile(file);
        break;
    default:
        throw new Error('Unsupported file format');
}
```

This cleanly separates file-format detection from parsing.

2.17.2 Degree calculation

javascript

```
function calculateDegrees() {
    nodes.forEach(node => {
        node.degree = 0;
        node.inDegree = 0;
        node.outDegree = 0;
    });

    edges.forEach(edge => {
        edge.source.outDegree++;
        edge.target.inDegree++;
        edge.source.degree++;
        edge.target.degree++;
    });
```

```
}
```

This function recomputes structural node statistics after data import.

2.17.3 Fit-to-screen behavior

```
javascript
scale = Math.min(
    canvas.width / graphWidth,
    canvas.height / graphHeight
);

const centerX = (minX + maxX) / 2;
const centerY = (minY + maxY) / 2;

offsetX = canvas.width / 2 - centerX * scale;
offsetY = canvas.height / 2 - centerY * scale;
```

The function calculates graph bounds and chooses a scale that keeps the full graph visible.

2.17.4 PNG export

```
javascript
function saveImage() {
    const link = document.createElement('a');
    link.download = 'network.png';
    link.href = canvas.toDataURL('image/png');
    link.click();
}
```

This uses the built-in canvas export capability and a temporary download link.

2.18 Strengths of the Implementation

2.18.1 Self-contained architecture

Most functionality is contained in one document. This makes the application easy to distribute, inspect, customize, and deploy.

2.18.2 No server-side processing

Network files are parsed in the browser, providing convenience and a degree of local-data privacy.

2.18.3 Broad input support

Support for text, spreadsheet, and JSON formats makes the program adaptable to common graph-data workflows.

2.18.4 Effective interaction set

The implementation includes the main controls expected in a network viewer:

- Zoom
- Pan
- Drag
- Search
- Hover details
- Neighborhood highlighting

- Layout selection
- Visual customization

2.18.5 Clear separation of responsibilities

The JavaScript is divided into logical sections:

- Global state
- Configuration
- Initialization
- Event binding
- File processing
- Data processing
- Layout
- Rendering
- Interaction
- UI utilities
- Sample data

2.18.6 Responsive visual design

The interface uses a modern dark theme, translucent panels, gradients, responsive layout rules, and clear status displays.

2.19 Limitations and Technical Observations

2.19.1 Force-layout scalability

The repulsion algorithm is $O(n^2)$. Performance will deteriorate as the number of nodes grows.

The comment:

javascript

// Repulsion Calculation (Accelerated using Barnes-Hut Approximation)

is inaccurate because no Barnes–Hut quadtree approximation is implemented.

Recommended improvement: implement a quadtree-based Barnes–Hut method, use a spatial grid, or use a mature force-layout library.

2.19.2 Header rows are not recognized

CSV and Excel header rows may become graph nodes.

Recommended improvement: provide a “first row is a header” option or automatically detect common header names.

2.19.3 Weighted detection uses only the first row

The weighted status is inferred from data[0][3]. If the first row lacks a valid weight but later rows contain weights, the graph is treated as unweighted.

Recommended improvement: inspect all rows or allow users to explicitly select the data schema.

2.19.4 Fallback handling changes valid zero values

The code uses:

javascript

parseInt(row[2]) || 1

parseFloat(row[3]) || 1

A valid value of 0 is therefore replaced with 1.

Recommended improvement:

javascript
const parsedWeight = Number.parseFloat(row[3]);
const weight = Number.isFinite(parsedWeight) ? parsedWeight : 1;

2.19.5 Direction and degree interpretation

Degree calculations always treat the stored source as outgoing and the stored target as incoming. An edge with type === -1 is visually reversed, but its degree contribution is not reversed.

Thus, for a row:

text
A,B,-1

the arrow is displayed as B → A, but the degree calculation still increments:

- A.outDegree
- B.inDegree

This is inconsistent.

Recommended improvement: normalize edge direction during import or account for type in calculateDegrees().

2.19.6 Network-type switching is visual and statistical only

Changing from directed to undirected does not modify stored data or recalculate degrees. It changes arrow rendering and density denominator.

2.19.7 Canvas width and height fields are not connected

The UI contains canvasWidth and canvasHeight inputs, but no event handlers use them. Canvas dimensions are always based on the container:

javascript
canvas.width = container.clientWidth;
canvas.height = container.clientHeight;

Recommended improvement: remove the unused fields or implement export-resolution and canvas-resizing behavior.

2.19.8 Node fixation is incomplete

Nodes contain fixed, fx, and fy fields, but there is no user-facing action that permanently fixes a node. During dragging, simulation movement is skipped only because the node equals selectedNode. After release, it moves again.

2.19.9 Arrowhead styling inconsistency

Highlighted edges are drawn in red, but arrowheads always use config.linkColor.

Recommended improvement: pass the active edge color into drawArrow().

2.19.10 Arrowhead scaling

The arrowhead length is divided by scale, while the node radius used to position the arrow endpoint is not similarly adjusted. Because the drawing context is already scaled, this may produce inconsistent visual sizing at different zoom levels.

2.19.11 Resize behavior

When the window is resized, the canvas dimensions change but graph positions and camera fitting are not automatically recalculated. Nodes may become clipped or constrained unexpectedly by the next force update.

2.19.12 Density edge cases

For zero or one node, the maximum number of possible edges is zero. If updateNetworkInfo() is called in this state, density calculation can produce NaN or an invalid result.

2.19.13 Duplicate edges and self-loops

The importer does not remove duplicate edges or specially handle self-loops. A self-loop is drawn as a zero-length line and may also cause division-related drawing issues in arrow calculations.

2.19.14 Mobile interaction

The CSS is responsive, but the canvas interaction code uses mouse events only. Touch dragging, pinch zooming, and touch selection are not implemented.

2.19.15 Accessibility

Radio inputs are visually hidden and canvas content has no semantic alternative. Toolbar buttons rely mainly on symbols and title attributes.

Potential improvements include:

- ARIA labels
- Keyboard navigation
- Focus styles
- A tabular alternative for graph data
- Screen-reader status messages

2.19.16 Security and privacy considerations

The application processes files locally, which is beneficial. However:

- The SheetJS library is loaded from a third-party CDN
- No subresource integrity attribute is supplied
- Very large or malformed files may consume excessive memory or block the main thread

For controlled environments, the library should be hosted locally and file-size limits should be added.

2.20 Recommended Enhancements

2.20.1 Data import improvements

Add:

- Header detection
- Column mapping
- Delimiter selection
- Encoding selection
- Row-level validation
- Error reporting with line numbers
- Duplicate-edge handling
- Self-loop options
- Explicit directed/weighted schema settings

2.20.2 Layout improvements

Possible enhancements include:

- Barnes–Hut acceleration
- Collision avoidance based on node radius
- Hierarchical layout
- Radial layout
- Community-based layout
- Component separation
- Simulation temperature or convergence threshold

- "Re-run layout" button

2.20.3 Analysis functions

Useful metrics could include:

- Connected components
- Strongly connected components
- Betweenness centrality
- Closeness centrality
- PageRank
- Clustering coefficient
- Community detection
- Shortest paths
- Degree distribution

2.20.4 Rendering improvements

Recommended additions:

- Curved edges
- Parallel-edge separation
- Self-loop rendering
- Edge labels
- Node shapes and icons
- Degree-scaled node size
- Weight-scaled edge width
- Community-based colors
- High-DPI canvas support
- SVG export

2.20.5 Interaction improvements

Possible additions include:

- Multi-node selection
- Box selection
- Context menus
- Permanent node pinning
- Undo and redo
- Keyboard shortcuts
- Touch and pointer events
- Node and edge editing
- Export of edited graph data

2.20.6 Software architecture

For long-term maintenance, the script could be split into modules:

- GraphModel
- FileParser
- LayoutEngine
- CanvasRenderer
- InteractionController
- StatisticsService
- UIController

This would reduce reliance on global state and improve testing.

2.21 Suggested Corrective Code Examples

2.21.1 Robust numeric parsing

javascript

```
function parseNumber(value, fallback) {
    const number = Number(value);
    return Number.isFinite(number) ? number : fallback;
}

const type = parseNumber(row[2], 1);
const weight = isWeighted ? parseNumber(row[3], 1) : 1;
```

2.21.2 Degree calculation respecting reverse direction

javascript

```
function calculateDegrees() {
    nodes.forEach(node => {
        node.degree = 0;
        node.inDegree = 0;
        node.outDegree = 0;
    });

    edges.forEach(edge => {
        let source = edge.source;
        let target = edge.target;

        if (isDirected && edge.type === -1) {
            [source, target] = [target, source];
        }

        source.degree++;
        target.degree++;

        if (isDirected) {
            source.outDegree++;
            target.inDegree++;
        } else {
            source.inDegree++;
            source.outDegree++;
            target.inDegree++;
            target.outDegree++;
        }
    });
}
```

2.21.3 Safe density calculation

javascript

```
function calculateDensity() {
    const n = nodes.length;
    const m = edges.length;

    if (n < 2) return 0;

    const maxEdges = isDirected
        ? n * (n - 1)
        : n * (n - 1) / 2;

    return maxEdges > 0 ? m / maxEdges : 0;
}
```

2.21.4 Local SheetJS deployment

Instead of a remote CDN reference:

html

```
<script src="./vendor/xlsx.full.min.js"></script>
```

This enables offline Excel import and reduces third-party dependency risk.

2.22 Testing Recommendations

A practical test plan should include the following categories.

2.22.1 Import tests

Test:

- Comma-separated data
- Tab-separated data
- Semicolon-separated data
- XLS and XLSX workbooks
- Edge-object JSON
- Row-array JSON
- Empty files
- Header-only files
- Missing columns
- Non-numeric weights
- Unicode node labels
- Duplicate edges
- Self-loops

2.22.2 Graph tests

Test:

- Empty graph
- One-node graph
- Two-node graph
- Disconnected components

- Dense graph
- Directed acyclic graph
- Bidirectional edges
- Negative or zero weights
- Reverse edge types

2.22.3 Interaction tests

Verify:

- Dragging at different zoom levels
- Panning
- Mouse-wheel zoom
- Fit-to-screen
- Search highlighting
- Neighborhood highlighting
- Popup positioning
- Pause and resume behavior
- Label and weight toggles

2.22.4 Performance tests

Measure responsiveness with increasing graph sizes, for example:

- 50 nodes
- 100 nodes
- 500 nodes
- 1,000 nodes

The force-directed mode is expected to become the limiting component due to its quadratic repulsion calculation.

2.22.5 Browser tests

Test on:

- Chrome
- Microsoft Edge
- Firefox
- Safari
- Mobile browsers if touch support is later added

2.23 Summary

NetGen 4.0 is a capable lightweight network visualization application with an attractive interface and a useful combination of file import, layouts, styling controls, statistics, and interactive exploration. Its client-side implementation makes it convenient for demonstrations, teaching, prototyping, and inspection of small to medium graph datasets.

The strongest aspects are its broad input support, straightforward canvas renderer, interactive camera controls, built-in force simulation, and self-contained deployment model. Its main technical limitation is the quadratic force calculation, and several details, such as header handling, reverse-direction degree calculation, unused canvas-size controls, self-loop handling, and large-network performance, would benefit from refinement.

With improved data validation, a genuinely accelerated force algorithm, richer graph analysis, and enhanced accessibility and touch support, the application could evolve from a compact visualizer into a more robust general-purpose network analysis tool.

3 JS/HTML Codes

The following are the JavaScript/HTML codes of the browser-based tool (http://www.iaees.org/publications/journals/nb/articles/2027-17(2)/netGen4.0.htm):

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>NetGen 4.0: The browser-based tool for network visualization</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background: linear-gradient(135deg, #1a1a2e 0%, #16213e 50%, #0f3460 100%);
            min-height: 100vh;
            color: #fff;
        }

        .container {
            display: flex;
            height: 100vh;
        }

        /* 控制面板样式 */
        .control-panel {
            width: 320px;
            background: rgba(255, 255, 255, 0.05);
            backdrop-filter: blur(10px);
            padding: 20px;
            overflow-y: auto;
            border-right: 1px solid rgba(255, 255, 255, 0.1);
        }

        .control-panel h1 {
            font-size: 1.5rem;
            margin-bottom: 5px;
            background: linear-gradient(90deg, #00d4ff, #7c3aed);
            -webkit-background-clip: text;
            -webkit-text-fill-color: transparent;
            background-clip: text;
```

```
}

.control-panel .subtitle {
    font-size: 0.75rem;
    color: #888;
    margin-bottom: 20px;
}

.section {
    margin-bottom: 20px;
    padding: 15px;
    background: rgba(255, 255, 255, 0.05);
    border-radius: 10px;
    border: 1px solid rgba(255, 255, 255, 0.1);
}

.section-title {
    font-size: 0.9rem;
    font-weight: 600;
    color: #00d4ff;
    margin-bottom: 12px;
    display: flex;
    align-items: center;
    gap: 8px;
}

.section-title::before {
    content: '';
    width: 4px;
    height: 16px;
    background: linear-gradient(180deg, #00d4ff, #7c3aed);
    border-radius: 2px;
}

.radio-group {
    display: flex;
    gap: 10px;
    flex-wrap: wrap;
}

.radio-option {
    flex: 1;
    min-width: 120px;
}

.radio-option input {
```

```
        display: none;
}

.radio-option label {
        display: block;
        padding: 10px 15px;
        background: rgba(255, 255, 255, 0.05);
        border: 2px solid rgba(255, 255, 255, 0.1);
        border-radius: 8px;
        cursor: pointer;
        text-align: center;
        font-size: 0.85rem;
        transition: all 0.3s ease;
}

.radio-option input:checked + label {
        background: rgba(0, 212, 255, 0.2);
        border-color: #00d4ff;
        color: #00d4ff;
}

.radio-option label:hover {
        background: rgba(255, 255, 255, 0.1);
}

.input-group {
        margin-bottom: 12px;
}

.input-group label {
        display: block;
        font-size: 0.8rem;
        color: #aaa;
        margin-bottom: 5px;
}

.input-group input[type="number"],
.input-group input[type="text"] {
        width: 100%;
        padding: 10px 12px;
        background: rgba(0, 0, 0, 0.3);
        border: 1px solid rgba(255, 255, 255, 0.1);
        border-radius: 6px;
        color: #fff;
        font-size: 0.9rem;
        transition: all 0.3s ease;
```

```
}

.input-group input:focus {
    outline: none;
    border-color: #00d4ff;
    box-shadow: 0 0 10px rgba(0, 212, 255, 0.3);
}

.input-row {
    display: flex;
    gap: 10px;
}

.input-row .input-group {
    flex: 1;
}

.color-picker-group {
    display: flex;
    align-items: center;
    gap: 10px;
    margin-bottom: 10px;
}

.color-picker-group label {
    flex: 1;
    font-size: 0.8rem;
    color: #aaa;
}

.color-picker-group input[type="color"] {
    width: 50px;
    height: 35px;
    border: none;
    border-radius: 6px;
    cursor: pointer;
    background: none;
}

.btn {
    width: 100%;
    padding: 12px 20px;
    border: none;
    border-radius: 8px;
    font-size: 0.9rem;
    font-weight: 600;
```

```
        cursor: pointer;
        transition: all 0.3s ease;
        margin-bottom: 10px;
    }

    .btn-primary {
        background: linear-gradient(135deg, #00d4ff, #7c3aed);
        color: #fff;
    }

    .btn-primary:hover {
        transform: translateY(-2px);
        box-shadow: 0 5px 20px rgba(0, 212, 255, 0.4);
    }

    .btn-secondary {
        background: rgba(255, 255, 255, 0.1);
        color: #fff;
        border: 1px solid rgba(255, 255, 255, 0.2);
    }

    .btn-secondary:hover {
        background: rgba(255, 255, 255, 0.2);
    }

    .btn-success {
        background: linear-gradient(135deg, #10b981, #059669);
        color: #fff;
    }

    .btn-danger {
        background: linear-gradient(135deg, #ef4444, #dc2626);
        color: #fff;
    }

    .btn-group {
        display: flex;
        gap: 10px;
    }

    .btn-group .btn {
        flex: 1;
    }

    /* 画布区域 */
    .canvas-container {
```

```
    flex: 1;
    position: relative;
    background: #0a0a0f;
    overflow: hidden;
}

.canvas-scroll {
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    overflow: auto;              /* 同时出现水平和垂直滚动条 */
}

.canvas-scroll::-webkit-scrollbar {
    width: 12px;
    height: 12px;
}

.canvas-scroll::-webkit-scrollbar-track {
    background: rgba(255, 255, 255, 0.05);
}

.canvas-scroll::-webkit-scrollbar-thumb {
    background: rgba(0, 212, 255, 0.5);
    border-radius: 6px;
}

.canvas-scroll::-webkit-scrollbar-thumb:hover {
    background: rgba(0, 212, 255, 0.8);
}

.canvas-scroll::-webkit-scrollbar-corner {
    background: rgba(255, 255, 255, 0.05);
}

#networkCanvas {
    display: block;
    cursor: grab;
}

#networkCanvas:active {
    cursor: grabbing;
}
```

```
/* 信息面板 */
.info-panel {
    position: absolute;
    top: 20px;
    right: 20px;
    background: rgba(0, 0, 0, 0.7);
    backdrop-filter: blur(10px);
    padding: 15px 20px;
    border-radius: 10px;
    border: 1px solid rgba(255, 255, 255, 0.1);
    font-size: 0.85rem;
    min-width: 200px;
}

.info-panel h3 {
    color: #00d4ff;
    margin-bottom: 10px;
    font-size: 0.9rem;
}

.info-item {
    display: flex;
    justify-content: space-between;
    padding: 5px 0;
    border-bottom: 1px solid rgba(255, 255, 255, 0.1);
}

.info-item:last-child {
    border-bottom: none;
}

.info-item .label {
    color: #888;
}

.info-item .value {
    color: #fff;
    font-weight: 600;
}

/* 工具栏 */
.toolbar {
    position: absolute;
    bottom: 20px;
    left: 50%;
    transform: translateX(-50%);
```

```
        display: flex;
        gap: 10px;
        background: rgba(0, 0, 0, 0.7);
        backdrop-filter: blur(10px);
        padding: 10px 20px;
        border-radius: 30px;
        border: 1px solid rgba(255, 255, 255, 0.1);
    }

    .toolbar-btn {
        width: 40px;
        height: 40px;
        border: none;
        border-radius: 50%;
        background: rgba(255, 255, 255, 0.1);
        color: #fff;
        cursor: pointer;
        transition: all 0.3s ease;
        display: flex;
        align-items: center;
        justify-content: center;
        font-size: 1.2rem;
    }

    .toolbar-btn:hover {
        background: rgba(0, 212, 255, 0.3);
        transform: scale(1.1);
    }

    .toolbar-btn.active {
        background: #00d4ff;
        color: #000;
    }

    /* 文件输入隐藏 */
    #fileInput {
        display: none;
    }

    /* 提示框 */
    .tooltip {
        position: absolute;
        background: rgba(0, 0, 0, 0.9);
        color: #fff;
        padding: 8px 12px;
        border-radius: 6px;
```

```
        font-size: 0.8rem;
        pointer-events: none;
        z-index: 1000;
        display: none;
        max-width: 250px;
    }

    /* 节点详情弹窗 */
    .node-detail {
        position: absolute;
        background: rgba(0, 0, 0, 0.9);
        backdrop-filter: blur(10px);
        padding: 15px;
        border-radius: 10px;
        border: 1px solid #00d4ff;
        display: none;
        z-index: 100;
        min-width: 180px;
    }

    .node-detail h4 {
        color: #00d4ff;
        margin-bottom: 10px;
    }

    .node-detail p {
        font-size: 0.85rem;
        color: #ccc;
        margin: 5px 0;
    }

    /* 加载动画 */
    .loading-overlay {
        position: absolute;
        top: 0;
        left: 0;
        right: 0;
        bottom: 0;
        background: rgba(0, 0, 0, 0.8);
        display: none;
        align-items: center;
        justify-content: center;
        z-index: 1000;
    }

    .loading-spinner {
```

```
        width: 50px;
        height: 50px;
        border: 3px solid rgba(255, 255, 255, 0.1);
        border-top-color: #00d4ff;
        border-radius: 50%;
        animation: spin 1s linear infinite;
}

@keyframes spin {
        to { transform: rotate(360deg); }
}

/* 滚动条样式 */
.control-panel::-webkit-scrollbar {
        width: 6px;
}

.control-panel::-webkit-scrollbar-track {
        background: rgba(255, 255, 255, 0.05);
}

.control-panel::-webkit-scrollbar-thumb {
        background: rgba(255, 255, 255, 0.2);
        border-radius: 3px;
}

/* 搜索框 */
.search-box {
        position: relative;
        margin-bottom: 15px;
}

.search-box input {
        width: 100%;
        padding: 10px 15px 10px 40px;
        background: rgba(0, 0, 0, 0.3);
        border: 1px solid rgba(255, 255, 255, 0.1);
        border-radius: 20px;
        color: #fff;
        font-size: 0.9rem;
}

.search-box::before {
        content: '□';
        position: absolute;
        left: 12px;
```

```
        top: 50%;
        transform: translateY(-50%);
        font-size: 1rem;
    }

    /* 统计卡片 */
    .stats-cards {
        display: grid;
        grid-template-columns: repeat(2, 1fr);
        gap: 10px;
        margin-bottom: 15px;
    }

    .stat-card {
        background: rgba(255, 255, 255, 0.05);
        padding: 12px;
        border-radius: 8px;
        text-align: center;
    }

    .stat-card .number {
        font-size: 1.5rem;
        font-weight: 700;
        background: linear-gradient(90deg, #00d4ff, #7c3aed);
        -webkit-background-clip: text;
        -webkit-text-fill-color: transparent;
        background-clip: text;
    }

    .stat-card .label {
        font-size: 0.75rem;
        color: #888;
        margin-top: 5px;
    }

    /* 图例 */
    .legend {
        display: flex;
        flex-wrap: wrap;
        gap: 10px;
        margin-top: 10px;
    }

    .legend-item {
        display: flex;
        align-items: center;
```

```
        gap: 5px;
        font-size: 0.8rem;
        color: #aaa;
    }

    .legend-color {
        width: 12px;
        height: 12px;
        border-radius: 3px;
    }

    /* 滑块样式 */
    input[type="range"] {
        width: 100%;
        height: 6px;
        border-radius: 3px;
        background: rgba(255, 255, 255, 0.1);
        appearance: none;
        outline: none;
    }

    input[type="range"]::-webkit-slider-thumb {
        appearance: none;
        width: 16px;
        height: 16px;
        border-radius: 50%;
        background: #00d4ff;
        cursor: pointer;
    }

    /* 响应式 */
    @media (max-width: 768px) {
        .container {
            flex-direction: column;
        }

        .control-panel {
            width: 100%;
            height: auto;
            max-height: 50vh;
        }

        .canvas-container {
            height: 50vh;
        }
    }
```

```
    </style>
</head>
<body>
    <div class="container">
        <!-- 控制面板 -->
        <div class="control-panel">
            <h1>NetGen 4.0</h1>
            <p class="subtitle">A browser-based tool for network visualization<br>
            <a href="http://www.iaees.org/publications/journals/nb/articles/2027-17(2)/2-Zhang-Abstract.asp" style="color: gray;">Zhang WJ. 2027. NetGen 4.0: The browser-based tool for network visualization. Network Biology, 17(2): 468-537</a>
            </p>

            <!-- 统计卡片 -->
            <div class="stats-cards">
                <div class="stat-card">
                    <div class="number" id="nodeCount">0</div>
                    <div class="label">Number of nodes</div>
                </div>
                <div class="stat-card">
                    <div class="number" id="edgeCount">0</div>
                    <div class="label">Number of edges</div>
                </div>
            </div>

            <!-- 搜索框 -->
            <div class="search-box">
                <input type="text" id="searchNode" placeholder="Search nodes...">
            </div>

            <!-- 网络类型 -->
            <div class="section">
                <div class="section-title">Type of network</div>
                <div class="radio-group">
                    <div class="radio-option">
                        <input type="radio" name="networkType" id="directed" value="directed" checked>
                        <label for="directed">Directed network</label>
                    </div>
                    <div class="radio-option">
                        <input type="radio" name="networkType" id="undirected" value="undirected">
                        <label for="undirected">Undirected network</label>
                    </div>
                </div>
            </div>

            <!-- 布局方式 -->
            <div class="section">
```

```
        <div class="section-title">Type of layout</div>
        <div class="radio-group">
            <div class="radio-option">
                <input type="radio" name="layoutType" id="manual" value="manual">
                <label for="manual">Manual layout</label>
            </div>
            <div class="radio-option">
                <input type="radio" name="layoutType" id="auto" value="auto" checked>
                <label for="auto">Automatic layout</label>
            </div>
        </div>
        <div style="margin-top: 10px;">
            <div class="radio-group">
                <div class="radio-option">
                    <input type="radio" name="layoutAlgorithm" id="forceLayout" value="force" checked>
                    <label for="forceLayout">Force directed layout</label>
                </div>
                <div class="radio-option">
                    <input type="radio" name="layoutAlgorithm" id="circleLayout" value="circle">
                    <label for="circleLayout">Circle layout</label>
                </div>
                <div class="radio-option">
                    <input type="radio" name="layoutAlgorithm" id="gridLayout" value="grid">
                    <label for="gridLayout">Grid layout</label>
                </div>
            </div>
        </div>
    </div>

    <!-- 节点设置 -->
    <div class="section">
        <div class="section-title">Node settings</div>
        <div class="input-group">
            <label>Node size: <span id="nodeSizeValue">20</span></label>
            <input type="range" id="nodeSize" min="5" max="50" value="20">
        </div>
        <div class="color-picker-group">
            <label>Node color</label>
            <input type="color" id="nodeColor" value="#00d4ff">
        </div>
        <div class="color-picker-group">
            <label>Label color</label>
            <input type="color" id="labelColor" value="#ffffff">
        </div>
        <div class="input-group">
            <label>Label size: <span id="labelSizeValue">12</span></label>
```

```
            <input type="range" id="labelSize" min="8" max="24" value="12">
        </div>
    </div>

    <!-- 连边设置 -->
    <div class="section">
        <div class="section-title">Link settings</div>
        <div class="input-group">
            <label>Link width: <span id="linkWidthValue">2</span></label>
            <input type="range" id="linkWidth" min="1" max="10" value="2">
        </div>
        <div class="color-picker-group">
            <label>Link color</label>
            <input type="color" id="linkColor" value="#4a90d9">
        </div>
        <div class="input-group">
            <label>Link length: <span id="linkLengthValue">150</span></label>
            <input type="range" id="linkLength" min="50" max="400" value="150">
        </div>
    </div>

    <!-- 力导向参数 -->
    <div class="section" id="forceSettings">
        <div class="section-title">Force settings</div>
        <div class="input-group">
            <label>Repulsion: <span id="repulsionValue">1000</span></label>
            <input type="range" id="repulsion" min="100" max="5000" value="1000">
        </div>
        <div class="input-group">
            <label>Attraction: <span id="attractionValue">0.01</span></label>
            <input type="range" id="attraction" min="1" max="100" value="10">
        </div>
        <div class="input-group">
            <label>Damping: <span id="dampingValue">0.9</span></label>
            <input type="range" id="damping" min="50" max="99" value="90">
        </div>
    </div>

    <!-- 画布设置 -->
    <div class="section">
        <div class="section-title">Canvas settings</div>
        <div class="color-picker-group">
            <label>Background color</label>
            <input type="color" id="bgColor" value="#0a0a0f">
        </div>
        <div class="input-row">
```

```
                    <div class="input-group">
                        <label>Canvas width</label>
                        <input type="number" id="canvasWidth" value="1200">
                    </div>
                    <div class="input-group">
                        <label>Canvas height</label>
                        <input type="number" id="canvasHeight" value="800">
                    </div>
                </div>
            </div>

            <!-- 操作按钮 -->
            <div class="section">
                <div class="section-title">Action</div>
                <input type="file" id="fileInput" accept=".csv,.txt,.xls,.xlsx,.json">
                <button class="btn btn-primary" onclick="document.getElementById('fileInput').click()">
                    □ Load file
                </button>
            <font size=2>(<a href="http://www.iaees.org/publications/journals/nb/articles/2027-17(2)/SampleDataFiles.zip" style="color: yellow;">Download</a> the package for sample data files)</font><br><br>
                <div class="btn-group">
                    <button class="btn btn-success" onclick="saveImage()">□ Save image</button>
                    <button class="btn btn-secondary" onclick="resetView()">□ Reset view</button>
                </div>
                <div class="btn-group">
                    <button class="btn btn-secondary" onclick="toggleAnimation()">
                        <span id="animBtnText">□□ Toggle animation</span>
                    </button>
                    <button class="btn btn-danger" onclick="clearNetwork()">□□ Clear network</button>
                </div>
            </div>

            <!-- 示例数据 -->
            <div class="section">
                <div class="section-title">Sample network</div>
                <button class="btn btn-secondary" onclick="loadSampleData('small')">Small network (10 nodes)</button>
                <button class="btn btn-secondary" onclick="loadSampleData('medium')">Medium network (30 nodes)</button>
                <button class="btn btn-secondary" onclick="loadSampleData('karate')">Karate network</button>
            </div>
        </div>

        <!-- 画布区域 -->
        <div class="canvas-container">
            <div class="canvas-scroll" id="canvasScroll">
                <canvas id="networkCanvas"></canvas>
```

```
        </div>

        <!-- 信息面板 -->
        <div class="info-panel" id="infoPanel">
            <h3>Network information</h3>
            <div class="info-item">
                <span class="label">Type of network</span>
                <span class="value" id="infoType">-</span>
            </div>
            <div class="info-item">
                <span class="label">Weighted</span>
                <span class="value" id="infoWeighted">-</span>
            </div>
            <div class="info-item">
                <span class="label">Average degree</span>
                <span class="value" id="infoAvgDegree">-</span>
            </div>
            <div class="info-item">
                <span class="label">Density</span>
                <span class="value" id="infoDensity">-</span>
            </div>
        </div>

        <!-- 工具栏 -->
        <div class="toolbar">
            <button class="toolbar-btn" onclick="zoomIn()" title="Zoom in">□</button>
            <button class="toolbar-btn" onclick="zoomOut()" title="Zoom out">□</button>
            <button class="toolbar-btn" onclick="fitToScreen()" title="Fit to screen">□</button>
            <button class="toolbar-btn" onclick="centerNetwork()" title="Center">□</button>
            <button class="toolbar-btn" id="toggleLabels" onclick="toggleLabels()" title="Toggle labels">□□</button>
            <button class="toolbar-btn" id="toggleWeights" onclick="toggleWeights()" title="Toggle
weights">□□ </button>
        </div>

        <!-- 节点详情 -->
        <div class="node-detail" id="nodeDetail">
            <h4 id="detailName">Name of node</h4>
            <p>Degree: <span id="detailDegree">0</span></p>
            <p>In-degree: <span id="detailInDegree">0</span></p>
            <p>Out-degree: <span id="detailOutDegree">0</span></p>
        </div>

        <!-- 提示框 -->
        <div class="tooltip" id="tooltip"></div>

        <!-- 加载动画 -->
```

```
            <div class="loading-overlay" id="loadingOverlay">
                <div class="loading-spinner"></div>
            </div>
        </div>
    </div>

    <!-- 引入 SheetJS 用于读取 Excel 文件 -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.18.5/xlsx.full.min.js"></script>

    <script>
        // ==================== 全局变量 ====================
        const canvas = document.getElementById('networkCanvas');
        const ctx = canvas.getContext('2d');

        let nodes = [];
        let edges = [];
        let nodeMap = new Map();

        // 视图变换
        let scale = 1;
        let offsetX = 0;
        let offsetY = 0;

        // 交互状态
        let isDragging = false;
        let isPanning = false;
        let selectedNode = null;
        let hoveredNode = null;
        let lastMousePos = { x: 0, y: 0 };

        // 动画状态
        let animationRunning = true;
        let animationId = null;

        // 显示选项
        let showLabels = true;
        let showWeights = true;

        // 网络属性
        let isDirected = true;
        let isWeighted = false;

        // ==================== 配置参数 ====================
        let config = {
            nodeSize: 20,
            nodeColor: '#00d4ff',
```

```
        labelColor: '#ffffff',
        labelSize: 12,
        linkWidth: 2,
        linkColor: '#4a90d9',
        linkLength: 150,
        repulsion: 1000,
        attraction: 0.01,
        damping: 0.9,
        bgColor: '#0a0a0f'
    };

    // ==================== 初始化 ====================
    function init() {
        resizeCanvas();
        bindEvents();
        updateConfigFromUI();
        animate();
    }

    function resizeCanvas() {
        const container = document.querySelector('.canvas-container');
        const w = parseInt(document.getElementById('canvasWidth').value) || 0;
        const h = parseInt(document.getElementById('canvasHeight').value) || 0;
        canvas.width = Math.max(container.clientWidth, w);
        canvas.height = Math.max(container.clientHeight, h);
    }

    // ==================== 事件绑定 ====================
    function bindEvents() {
        // 窗口大小改变
        window.addEventListener('resize', resizeCanvas);
        document.getElementById('canvasWidth').addEventListener('input', resizeCanvas);
        document.getElementById('canvasHeight').addEventListener('input', resizeCanvas);

        // 文件输入
        document.getElementById('fileInput').addEventListener('change', handleFileUpload);

        // 鼠标事件
        canvas.addEventListener('mousedown', onMouseDown);
        canvas.addEventListener('mousemove', onMouseMove);
        canvas.addEventListener('mouseup', onMouseUp);
        canvas.addEventListener('wheel', onWheel);
        canvas.addEventListener('dblclick', onDoubleClick);
        canvas.addEventListener('contextmenu', (e) => e.preventDefault());

        // UI 控件事件
```

```
    bindUIEvents();

    // 搜索功能
    document.getElementById('searchNode').addEventListener('input', searchNodes);
}

function bindUIEvents() {
    // 滑块更新
    const sliders = [
        { id: 'nodeSize', config: 'nodeSize', display: 'nodeSizeValue' },
        { id: 'labelSize', config: 'labelSize', display: 'labelSizeValue' },
        { id: 'linkWidth', config: 'linkWidth', display: 'linkWidthValue' },
        { id: 'linkLength', config: 'linkLength', display: 'linkLengthValue' },
        { id: 'repulsion', config: 'repulsion', display: 'repulsionValue' },
        { id: 'attraction', config: null, display: 'attractionValue', transform: v => v / 1000 },
        { id: 'damping', config: null, display: 'dampingValue', transform: v => v / 100 }
    ];

    sliders.forEach(slider => {
        const elem = document.getElementById(slider.id);
        elem.addEventListener('input', () => {
            const value = slider.transform ? slider.transform(elem.value) : elem.value;
            document.getElementById(slider.display).textContent = value;
            if (slider.config) {
                config[slider.config] = parseFloat(elem.value);
            } else if (slider.id === 'attraction') {
                config.attraction = value;
            } else if (slider.id === 'damping') {
                config.damping = value;
            }
        });
    });

    // 颜色选择器
    const colorPickers = [
        { id: 'nodeColor', config: 'nodeColor' },
        { id: 'labelColor', config: 'labelColor' },
        { id: 'linkColor', config: 'linkColor' },
        { id: 'bgColor', config: 'bgColor' }
    ];

    colorPickers.forEach(picker => {
        document.getElementById(picker.id).addEventListener('input', (e) => {
            config[picker.config] = e.target.value;
        });
    });
```

```
        // 网络类型
        document.querySelectorAll('input[name="networkType"]').forEach(radio => {
            radio.addEventListener('change', (e) => {
                isDirected = e.target.value === 'directed';
                updateNetworkInfo();
            });
        });

        // 布局算法
        document.querySelectorAll('input[name="layoutAlgorithm"]').forEach(radio => {
            radio.addEventListener('change', (e) => {
                if (e.target.value === 'circle') {
                    applyCircleLayout();
                } else if (e.target.value === 'grid') {
                    applyGridLayout();
                }
            });
        });
    }

    function updateConfigFromUI() {
        config.nodeSize = parseInt(document.getElementById('nodeSize').value);
        config.nodeColor = document.getElementById('nodeColor').value;
        config.labelColor = document.getElementById('labelColor').value;
        config.labelSize = parseInt(document.getElementById('labelSize').value);
        config.linkWidth = parseInt(document.getElementById('linkWidth').value);
        config.linkColor = document.getElementById('linkColor').value;
        config.linkLength = parseInt(document.getElementById('linkLength').value);
        config.repulsion = parseInt(document.getElementById('repulsion').value);
        config.attraction = parseInt(document.getElementById('attraction').value) / 1000;
        config.damping = parseInt(document.getElementById('damping').value) / 100;
        config.bgColor = document.getElementById('bgColor').value;
    }

    // ==================== 文件处理 ====================
    async function handleFileUpload(event) {
        const file = event.target.files[0];
        if (!file) return;

        showLoading(true);

        try {
            const extension = file.name.split('.').pop().toLowerCase();
            let data;
```

```
            switch (extension) {
                case 'csv':
                case 'txt':
                    data = await readTextFile(file);
                    break;
                case 'xls':
                case 'xlsx':
                    data = await readExcelFile(file);
                    break;
                case 'json':
                    data = await readJSONFile(file);
                    break;
                default:
                    throw new Error('不支持的文件格式');
            }

            processData(data);
        } catch (error) {
            alert('File read error: ' + error.message);
        } finally {
            showLoading(false);
            event.target.value = '';
        }
    }

    function readTextFile(file) {
        return new Promise((resolve, reject) => {
            const reader = new FileReader();
            reader.onload = (e) => {
                const content = e.target.result;
                const lines = content.trim().split('\n');
                const data = [];

                for (const line of lines) {
                    if (!line.trim()) continue;
                    // 支持逗号、制表符、分号分隔
                    const parts = line.split(/[,\t;]/).map(p => p.trim());
                    if (parts.length >= 3) {
                        data.push(parts);
                    }
                }

                resolve(data);
            };
            reader.onerror = reject;
            reader.readAsText(file);
```

```
    });
}

function readExcelFile(file) {
    return new Promise((resolve, reject) => {
        const reader = new FileReader();
        reader.onload = (e) => {
            try {
                const data = new Uint8Array(e.target.result);
                const workbook = XLSX.read(data, { type: 'array' });
                const firstSheet = workbook.Sheets[workbook.SheetNames[0]];
                const jsonData = XLSX.utils.sheet_to_json(firstSheet, { header: 1 });

                // 过滤空行
                const filtered = jsonData.filter(row => row.length >= 3 && row[0] !== '');
                resolve(filtered);
            } catch (err) {
                reject(err);
            }
        };
        reader.onerror = reject;
        reader.readAsArrayBuffer(file);
    });
}

function readJSONFile(file) {
    return new Promise((resolve, reject) => {
        const reader = new FileReader();
        reader.onload = (e) => {
            try {
                const json = JSON.parse(e.target.result);
                // 支持多种 JSON 格式
                if (json.edges) {
                    const data = json.edges.map(edge => [
                        edge.source || edge.from,
                        edge.target || edge.to,
                        edge.type || edge.direction || 1,
                        edge.weight || edge.value
                    ]);
                    resolve(data);
                } else if (Array.isArray(json)) {
                    resolve(json);
                } else {
                    reject(new Error('Unsupported JSON format'));
                }
            } catch (err) {
```

```
                reject(err);
            }
        };
        reader.onerror = reject;
        reader.readAsText(file);
    });
}

// ==================== 数据处理 ====================
function processData(data) {
    nodes = [];
    edges = [];
    nodeMap.clear();

    // 自动检测是否为加权网络 (4 列 vs 3 列)
    isWeighted = data.length > 0 && data[0].length >= 4 && !isNaN(parseFloat(data[0][3]));

    for (const row of data) {
        const source = String(row[0]).trim();
        const target = String(row[1]).trim();
        const type = parseInt(row[2]) || 1;
        const weight = isWeighted ? parseFloat(row[3]) || 1 : 1;

        // 添加节点
        if (!nodeMap.has(source)) {
            const node = createNode(source);
            nodes.push(node);
            nodeMap.set(source, node);
        }

        if (!nodeMap.has(target)) {
            const node = createNode(target);
            nodes.push(node);
            nodeMap.set(target, node);
        }

        // 添加边
        edges.push({
            source: nodeMap.get(source),
            target: nodeMap.get(target),
            type: type,
            weight: weight
        });
    }

    // 计算节点度数
```

```
        calculateDegrees();

        // 初始布局
        const layoutType = document.querySelector('input[name="layoutAlgorithm"]:checked').value;
        if (layoutType === 'circle') {
            applyCircleLayout();
        } else if (layoutType === 'grid') {
            applyGridLayout();
        } else {
            applyRandomLayout();
        }

        // 更新 UI
        updateStats();
        updateNetworkInfo();
        fitToScreen();
    }

    function createNode(label) {
        return {
            id: label,
            label: label,
            x: Math.random() * canvas.width,
            y: Math.random() * canvas.height,
            vx: 0,
            vy: 0,
            fx: null,
            fy: null,
            degree: 0,
            inDegree: 0,
            outDegree: 0,
            fixed: false,
            highlighted: false
        };
    }

    function calculateDegrees() {
        // 重置度数
        nodes.forEach(node => {
            node.degree = 0;
            node.inDegree = 0;
            node.outDegree = 0;
        });

        // 计算度数
        edges.forEach(edge => {
```

```
            edge.source.outDegree++;
            edge.target.inDegree++;
            edge.source.degree++;
            edge.target.degree++;
        });
    }

    // ==================== 布局算法 ====================
    function applyRandomLayout() {
        const padding = 100;
        nodes.forEach(node => {
            node.x = padding + Math.random() * (canvas.width - 2 * padding);
            node.y = padding + Math.random() * (canvas.height - 2 * padding);
            node.vx = 0;
            node.vy = 0;
        });
    }

    function applyCircleLayout() {
        const centerX = canvas.width / 2;
        const centerY = canvas.height / 2;
        const radius = Math.min(canvas.width, canvas.height) * 0.35;

        nodes.forEach((node, i) => {
            const angle = (2 * Math.PI * i) / nodes.length;
            node.x = centerX + radius * Math.cos(angle);
            node.y = centerY + radius * Math.sin(angle);
            node.vx = 0;
            node.vy = 0;
        });
    }

    function applyGridLayout() {
        const cols = Math.ceil(Math.sqrt(nodes.length));
        const rows = Math.ceil(nodes.length / cols);
        const cellWidth = (canvas.width - 200) / cols;
        const cellHeight = (canvas.height - 200) / rows;

        nodes.forEach((node, i) => {
            const col = i % cols;
            const row = Math.floor(i / cols);
            node.x = 100 + col * cellWidth + cellWidth / 2;
            node.y = 100 + row * cellHeight + cellHeight / 2;
            node.vx = 0;
            node.vy = 0;
        });
```

```
}

// ==================== 力导向布局 (优化版) ====================
function applyForceLayout() {
    if (!animationRunning || nodes.length === 0) return;

    const n = nodes.length;

    // 初始化力
    nodes.forEach(node => {
        node.fx = 0;
        node.fy = 0;
    });

    // 斥力计算 (使用 Barnes-Hut 近似加速)
    for (let i = 0; i < n; i++) {
        for (let j = i + 1; j < n; j++) {
            const dx = nodes[j].x - nodes[i].x;
            const dy = nodes[j].y - nodes[i].y;
            let dist = Math.sqrt(dx * dx + dy * dy);
            dist = Math.max(dist, 1); // 防止除以零

            // 库仑力 (斥力)
            const force = config.repulsion / (dist * dist);
            const fx = (dx / dist) * force;
            const fy = (dy / dist) * force;

            nodes[i].fx -= fx;
            nodes[i].fy -= fy;
            nodes[j].fx += fx;
            nodes[j].fy += fy;
        }
    }

    // 弹簧力 (引力)
    edges.forEach(edge => {
        const dx = edge.target.x - edge.source.x;
        const dy = edge.target.y - edge.source.y;
        let dist = Math.sqrt(dx * dx + dy * dy);
        dist = Math.max(dist, 1);

        // 胡克定律
        const displacement = dist - config.linkLength;
        const force = config.attraction * displacement;
        const fx = (dx / dist) * force;
        const fy = (dy / dist) * force;
```

```
            edge.source.fx += fx;
            edge.source.fy += fy;
            edge.target.fx -= fx;
            edge.target.fy -= fy;
        });

        // 向心力 (防止节点飘远)
        const centerX = canvas.width / 2;
        const centerY = canvas.height / 2;
        const centerForce = 0.01;

        nodes.forEach(node => {
            node.fx += (centerX - node.x) * centerForce;
            node.fy += (centerY - node.y) * centerForce;
        });

        // 更新位置
        nodes.forEach(node => {
            if (node.fixed || node === selectedNode) return;

            // 应用力
            node.vx = (node.vx + node.fx) * config.damping;
            node.vy = (node.vy + node.fy) * config.damping;

            // 限制速度
            const maxSpeed = 10;
            const speed = Math.sqrt(node.vx * node.vx + node.vy * node.vy);
            if (speed > maxSpeed) {
                node.vx = (node.vx / speed) * maxSpeed;
                node.vy = (node.vy / speed) * maxSpeed;
            }

            // 更新位置
            node.x += node.vx;
            node.y += node.vy;

            // 边界约束
            const padding = config.nodeSize + 20;
            node.x = Math.max(padding, Math.min(canvas.width - padding, node.x));
            node.y = Math.max(padding, Math.min(canvas.height - padding, node.y));
        });
    }

    // ==================== 绘制函数 ====================
    function draw() {
```

```
        // 清空画布
        ctx.fillStyle = config.bgColor;
        ctx.fillRect(0, 0, canvas.width, canvas.height);

        // 保存状态
        ctx.save();

        // 应用视图变换
        ctx.translate(offsetX, offsetY);
        ctx.scale(scale, scale);

        // 绘制边
        drawEdges();

        // 绘制节点
        drawNodes();

        // 恢复状态
        ctx.restore();
    }

    function drawEdges() {
        edges.forEach(edge => {
            const source = edge.source;
            const target = edge.target;

            ctx.beginPath();
            ctx.strokeStyle = edge.highlighted ? '#ff6b6b' : config.linkColor;
            ctx.lineWidth = config.linkWidth / scale;

            // 绘制线条
            ctx.moveTo(source.x, source.y);
            ctx.lineTo(target.x, target.y);
            ctx.stroke();

            // 绘制箭头 (有向图)
            if (isDirected && edge.type === 1) {
                drawArrow(source.x, source.y, target.x, target.y);
            } else if (isDirected && edge.type === -1) {
                drawArrow(target.x, target.y, source.x, source.y);
            }

            // 绘制权重
            if (showWeights && isWeighted) {
                const midX = (source.x + target.x) / 2;
                const midY = (source.y + target.y) / 2;
```

```
                ctx.fillStyle = config.labelColor;
                ctx.font = `${10 / scale}px Arial`;
                ctx.textAlign = 'center';
                ctx.fillText(edge.weight.toFixed(2), midX, midY - 5);
            }
        });
    }

    function drawArrow(fromX, fromY, toX, toY) {
        const headLength = (config.nodeSize + 10) / scale;
        const dx = toX - fromX;
        const dy = toY - fromY;
        const angle = Math.atan2(dy, dx);

        // 计算箭头终点 (节点边缘)
        const dist = Math.sqrt(dx * dx + dy * dy);
        const nodeRadius = config.nodeSize / 2;
        const endX = toX - (dx / dist) * nodeRadius;
        const endY = toY - (dy / dist) * nodeRadius;

        // 绘制箭头
        ctx.beginPath();
        ctx.moveTo(endX, endY);
        ctx.lineTo(
            endX - headLength * Math.cos(angle - Math.PI / 6),
            endY - headLength * Math.sin(angle - Math.PI / 6)
        );
        ctx.lineTo(
            endX - headLength * Math.cos(angle + Math.PI / 6),
            endY - headLength * Math.sin(angle + Math.PI / 6)
        );
        ctx.closePath();
        ctx.fillStyle = config.linkColor;
        ctx.fill();
    }

    function drawNodes() {
        nodes.forEach(node => {
            const radius = config.nodeSize / 2;

            // 节点阴影
            if (node.highlighted || node === hoveredNode) {
                ctx.beginPath();
                ctx.arc(node.x, node.y, radius + 8, 0, Math.PI * 2);
                ctx.fillStyle = 'rgba(0, 212, 255, 0.3)';
                ctx.fill();
```

```
            }

            // 节点圆
            ctx.beginPath();
            ctx.arc(node.x, node.y, radius, 0, Math.PI * 2);

            // 根据度数调整颜色深度
            const nodeColor = node === selectedNode ? '#ff6b6b' :
                              node.highlighted ? '#ffd93d' : config.nodeColor;
            ctx.fillStyle = nodeColor;
            ctx.fill();

            // 节点边框
            ctx.strokeStyle = '#fff';
            ctx.lineWidth = 2 / scale;
            ctx.stroke();

            // 节点标签
            if (showLabels) {
                ctx.fillStyle = config.labelColor;
                ctx.font = `${config.labelSize / scale}px Arial`;
                ctx.textAlign = 'center';
                ctx.textBaseline = 'top';
                ctx.fillText(node.label, node.x, node.y + radius + 5);
            }
        });
    }

    // ==================== 动画循环 ====================
    function animate() {
        const layoutType = document.querySelector('input[name="layoutType"]:checked')?.value || 'auto';
        const algorithm = document.querySelector('input[name="layoutAlgorithm"]:checked')?.value || 'force';

        if (layoutType === 'auto' && algorithm === 'force') {
            applyForceLayout();
        }

        draw();
        animationId = requestAnimationFrame(animate);
    }

    // ==================== 交互处理 ====================
    function onMouseDown(e) {
        const pos = getMousePos(e);
        const worldPos = screenToWorld(pos.x, pos.y);
```

```
        // 查找点击的节点
        const clickedNode = findNodeAt(worldPos.x, worldPos.y);

        if (e.button === 0) { // 左键
            if (clickedNode) {
                selectedNode = clickedNode;
                isDragging = true;
                showNodeDetail(clickedNode, e.clientX, e.clientY);
            } else {
                isPanning = true;
                hideNodeDetail();
            }
        }

        lastMousePos = pos;
    }

    function onMouseMove(e) {
        const pos = getMousePos(e);
        const worldPos = screenToWorld(pos.x, pos.y);

        if (isDragging && selectedNode) {
            selectedNode.x = worldPos.x;
            selectedNode.y = worldPos.y;
            selectedNode.vx = 0;
            selectedNode.vy = 0;
        } else if (isPanning) {
            offsetX += pos.x - lastMousePos.x;
            offsetY += pos.y - lastMousePos.y;
        } else {
            // 悬停检测
            const hovered = findNodeAt(worldPos.x, worldPos.y);
            if (hovered !== hoveredNode) {
                hoveredNode = hovered;
                canvas.style.cursor = hovered ? 'pointer' : 'grab';
                if (hovered) {
                    showTooltip(hovered, e.clientX, e.clientY);
                } else {
                    hideTooltip();
                }
            }
        }

        lastMousePos = pos;
    }
```

```
function onMouseUp(e) {
    isDragging = false;
    isPanning = false;
    selectedNode = null;
}

function onWheel(e) {
    e.preventDefault();
    const pos = getMousePos(e);
    const worldPosBefore = screenToWorld(pos.x, pos.y);

    // 缩放
    const zoomFactor = e.deltaY > 0 ? 0.9 : 1.1;
    scale *= zoomFactor;
    scale = Math.max(0.1, Math.min(5, scale));

    // 保持鼠标位置不变
    const worldPosAfter = screenToWorld(pos.x, pos.y);
    offsetX += (worldPosAfter.x - worldPosBefore.x) * scale;
    offsetY += (worldPosAfter.y - worldPosBefore.y) * scale;
}

function onDoubleClick(e) {
    const pos = getMousePos(e);
    const worldPos = screenToWorld(pos.x, pos.y);
    const clickedNode = findNodeAt(worldPos.x, worldPos.y);

    if (clickedNode) {
        // 高亮相邻节点和边
        highlightNeighbors(clickedNode);
    } else {
        // 清除高亮
        clearHighlights();
    }
}

// ==================== 辅助函数 ====================
function getMousePos(e) {
    const rect = canvas.getBoundingClientRect();
    return {
        x: e.clientX - rect.left,
        y: e.clientY - rect.top
    };
}

function screenToWorld(x, y) {
```

```
        return {
            x: (x - offsetX) / scale,
            y: (y - offsetY) / scale
        };
    }

    function worldToScreen(x, y) {
        return {
            x: x * scale + offsetX,
            y: y * scale + offsetY
        };
    }

    function findNodeAt(x, y) {
        const radius = config.nodeSize / 2 + 5;
        for (let i = nodes.length - 1; i >= 0; i--) {
            const node = nodes[i];
            const dx = x - node.x;
            const dy = y - node.y;
            if (dx * dx + dy * dy < radius * radius) {
                return node;
            }
        }
        return null;
    }

    function highlightNeighbors(node) {
        clearHighlights();
        node.highlighted = true;

        edges.forEach(edge => {
            if (edge.source === node || edge.target === node) {
                edge.highlighted = true;
                edge.source.highlighted = true;
                edge.target.highlighted = true;
            }
        });
    }

    function clearHighlights() {
        nodes.forEach(node => node.highlighted = false);
        edges.forEach(edge => edge.highlighted = false);
    }

    // ==================== UI 功能 ====================
    function showTooltip(node, x, y) {
```

```
        const tooltip = document.getElementById('tooltip');
        tooltip.innerHTML = `
            <strong>${node.label}</strong><br>
            Degree: ${node.degree} | In-degree: ${node.inDegree} | Out-degree: ${node.outDegree}
        `;
        tooltip.style.left = (x + 15) + 'px';
        tooltip.style.top = (y + 15) + 'px';
        tooltip.style.display = 'block';
    }

    function hideTooltip() {
        document.getElementById('tooltip').style.display = 'none';
    }

    function showNodeDetail(node, x, y) {
        const detail = document.getElementById('nodeDetail');
        document.getElementById('detailName').textContent = node.label;
        document.getElementById('detailDegree').textContent = node.degree;
        document.getElementById('detailInDegree').textContent = node.inDegree;
        document.getElementById('detailOutDegree').textContent = node.outDegree;
        detail.style.left = (x + 20) + 'px';
        detail.style.top = (y - 50) + 'px';
        detail.style.display = 'block';
    }

    function hideNodeDetail() {
        document.getElementById('nodeDetail').style.display = 'none';
    }

    function showLoading(show) {
        document.getElementById('loadingOverlay').style.display = show ? 'flex' : 'none';
    }

    function updateStats() {
        document.getElementById('nodeCount').textContent = nodes.length;
        document.getElementById('edgeCount').textContent = edges.length;
    }

    function updateNetworkInfo() {
        document.getElementById('infoType').textContent = isDirected ? 'Directed' : 'Undirected';
        document.getElementById('infoWeighted').textContent = isWeighted ? 'Yes' : 'No';

        if (nodes.length > 0) {
            const avgDegree = (2 * edges.length / nodes.length).toFixed(2);
            const maxEdges = isDirected ? nodes.length * (nodes.length - 1) : nodes.length * (nodes.length - 1) / 2;
            const density = (edges.length / maxEdges).toFixed(4);
```

```
            document.getElementById('infoAvgDegree').textContent = avgDegree;
            document.getElementById('infoDensity').textContent = density;
        }
    }

    function searchNodes(e) {
        const query = e.target.value.toLowerCase();
        clearHighlights();

        if (query) {
            nodes.forEach(node => {
                if (node.label.toLowerCase().includes(query)) {
                    node.highlighted = true;
                }
            });
        }
    }

    // ==================== 工具栏功能 ====================
    function zoomIn() {
        scale *= 1.2;
        scale = Math.min(5, scale);
    }

    function zoomOut() {
        scale *= 0.8;
        scale = Math.max(0.1, scale);
    }

    function fitToScreen() {
        if (nodes.length === 0) return;

        let minX = Infinity, maxX = -Infinity;
        let minY = Infinity, maxY = -Infinity;

        nodes.forEach(node => {
            minX = Math.min(minX, node.x);
            maxX = Math.max(maxX, node.x);
            minY = Math.min(minY, node.y);
            maxY = Math.max(maxY, node.y);
        });

        const padding = 100;
        const graphWidth = maxX - minX + 2 * padding;
        const graphHeight = maxY - minY + 2 * padding;
```

```
        scale = Math.min(
            canvas.width / graphWidth,
            canvas.height / graphHeight
        );
        scale = Math.max(0.1, Math.min(2, scale));

        const centerX = (minX + maxX) / 2;
        const centerY = (minY + maxY) / 2;

        offsetX = canvas.width / 2 - centerX * scale;
        offsetY = canvas.height / 2 - centerY * scale;
    }

    function centerNetwork() {
        if (nodes.length === 0) return;

        let sumX = 0, sumY = 0;
        nodes.forEach(node => {
            sumX += node.x;
            sumY += node.y;
        });

        const centerX = sumX / nodes.length;
        const centerY = sumY / nodes.length;

        offsetX = canvas.width / 2 - centerX * scale;
        offsetY = canvas.height / 2 - centerY * scale;
    }

    function resetView() {
        scale = 1;
        offsetX = 0;
        offsetY = 0;
        fitToScreen();
    }

    function toggleLabels() {
        showLabels = !showLabels;
        document.getElementById('toggleLabels').classList.toggle('active', showLabels);
    }

    function toggleWeights() {
        showWeights = !showWeights;
        document.getElementById('toggleWeights').classList.toggle('active', showWeights);
    }
```

```
function toggleAnimation() {
    animationRunning = !animationRunning;
    document.getElementById('animBtnText').textContent = animationRunning ? '□□ Toggle' : '▶□ Continue';
}

function clearNetwork() {
    nodes = [];
    edges = [];
    nodeMap.clear();
    updateStats();
    document.getElementById('infoAvgDegree').textContent = '-';
    document.getElementById('infoDensity').textContent = '-';
}

function saveImage() {
    const link = document.createElement('a');
    link.download = 'network.png';
    link.href = canvas.toDataURL('image/png');
    link.click();
}

// ==================== 示例数据 ====================
function loadSampleData(type) {
    let data;

    switch (type) {
        case 'small':
            data = generateRandomNetwork(10, 15);
            break;
        case 'medium':
            data = generateRandomNetwork(30, 50);
            break;
        case 'karate':
            data = getKarateClubData();
            break;
        default:
            return;
    }

    processData(data);
}

function generateRandomNetwork(nodeCount, edgeCount) {
    const data = [];
    const nodeNames = [];
```

```
    for (let i = 0; i < nodeCount; i++) {
        nodeNames.push('Node' + (i + 1));
    }

    for (let i = 0; i < edgeCount; i++) {
        const source = nodeNames[Math.floor(Math.random() * nodeCount)];
        let target = nodeNames[Math.floor(Math.random() * nodeCount)];
        while (target === source) {
            target = nodeNames[Math.floor(Math.random() * nodeCount)];
        }
        const type = Math.random() > 0.5 ? 1 : -1;
        const weight = (Math.random() * 10).toFixed(2);
        data.push([source, target, type, weight]);
    }

    return data;
}

function getKarateClubData() {
    // Zachary 空手道俱乐部数据
    const edges = [
        [1,2], [1,3], [1,4], [1,5], [1,6], [1,7], [1,8], [1,9], [1,11], [1,12], [1,13], [1,14], [1,18], [1,20], [1,22], [1,32],
        [2,3], [2,4], [2,8], [2,14], [2,18], [2,20], [2,22], [2,31],
        [3,4], [3,8], [3,9], [3,10], [3,14], [3,28], [3,29], [3,33],
        [4,8], [4,13], [4,14],
        [5,7], [5,11],
        [6,7], [6,11], [6,17],
        [7,17],
        [9,31], [9,33], [9,34],
        [10,34],
        [14,34],
        [15,33], [15,34],
        [16,33], [16,34],
        [19,33], [19,34],
        [20,34],
        [21,33], [21,34],
        [23,33], [23,34],
        [24,26], [24,28], [24,30], [24,33], [24,34],
        [25,26], [25,28], [25,32],
        [26,32],
        [27,30], [27,34],
        [28,34],
        [29,32], [29,34],
        [30,33], [30,34],
        [31,33], [31,34],
```

```
                [32,33], [32,34],
                [33,34]
            ];

            return edges.map(e => ['Member' + e[0], 'Member' + e[1], 1]);
        }

        // ==================== 启动 ====================
        init();
    </script>
</body>
</html>
```

Using the downloaded sample data files (http://www.iaees.org/publications/journals/nb/articles/2027-17(2)/SampleDataFiles.zip), the program can be used to generate sample networks (Fig. 1).

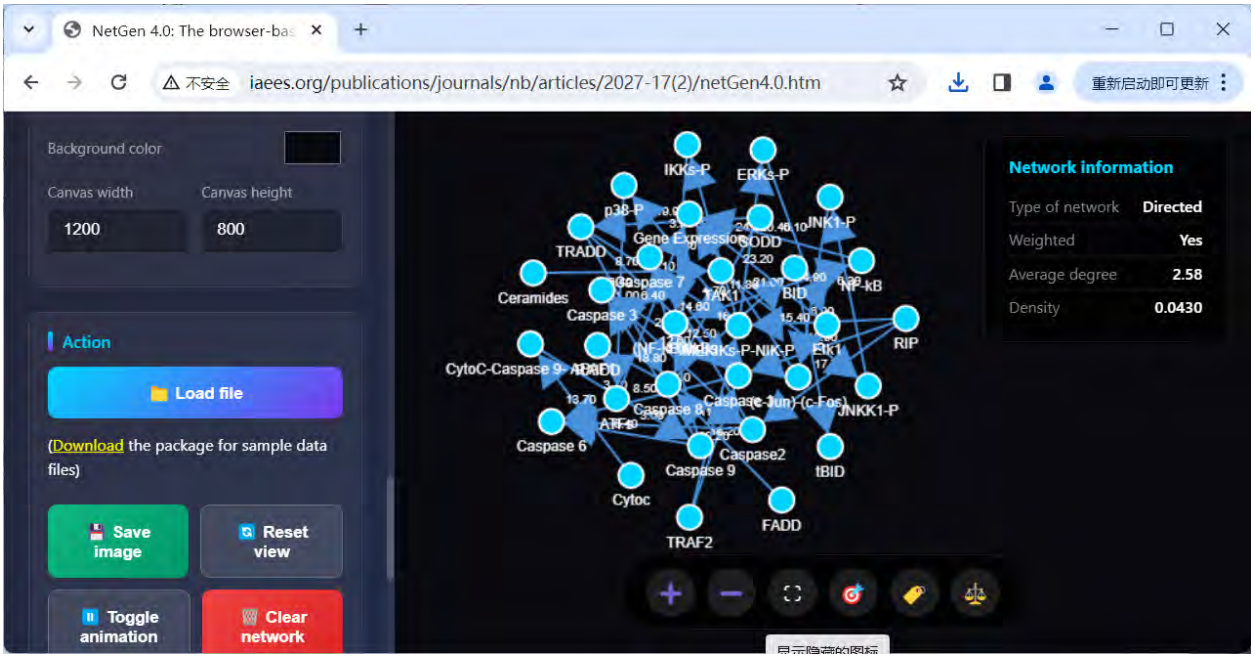

Fig. 1 The browser-based tool pages.

References

Bastian M, Heymann S, Jacomy M. 2009. Gephi: An open source software for exploring and manipulating networks. Proceedings of the International AAAI Conference on Web and Social Media, 3(1). https://ojs.aaai.org/index.php/ICWSM/article/view/13937

Csárdi G, Nepusz T. 2006. The igraph software package for complex network research. InterJournal, Complex Systems, 1695. http://www.necsi.edu/events/iccs6/papers/c1602a3c126ba822d0bc4293371c.pdf

Franz M, Lopes CT, Huck G, Dong Y, Sumer O, Bader GD. 2015. Cytoscape.js: A graph theory library for visualisation and analysis. Bioinformatics, 32(2): 309-311. https://europepmc.org/article/PMC/PMC4708103

Girard P, Jacomy A, Simard B, Jacomy M. 2025. Gephi Lite: A lighter web-based version of Gephi. Digital Humanities *2025*, Lisbon, Portugul. https://lite.gephi.org/v1.0.2/

Hagberg AA, Schult DA, Swart PJ. 2008. Exploring network structure, dynamics, and function using NetworkX. In: Proceedings of the 7th Python in Science Conference (SciPy2008), 11-15. https://proceedings.scipy.org/articles/TCWV9851

Perrone G, Unpingco J, Lu HM. 2020. Network visualizations with Pyvis and VisJS. In: Proceedings of the 19th Python in Science Conference (SciPy 2020), 58-62. https://proceedings.scipy.org/articles/Majora-342d178e-008

Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, Amin N, Schwikowski B, Ideker T. 2003. Cytoscape: A software environment for integrated models of biomolecular interaction networks. Genome Research, 13(11): 2498-2504. https://genome.cshlp.org/content/13/11/2498

Zhang WJ. 2012. A Java software for drawing graphs. Network Biology, 2(1): 38-44. http://www.iaees.org/publications/journals/nb/articles/2012-2(1)/a-Java-software-for-drawing-graphs.pdf

Zhang WJ. 2021. A web tool for generating user-interface interactive networks. Network Biology, 11(4): 247-262. http://www.iaees.org/publications/journals/nb/articles/2021-11(4)/web-tool-for-generating-user-interface-interactive-networks.pdf

Zhang WJ. 2024a. A Matlab software for visualizing user-interface interactive networks. Network Biology, 14(1): 13-19. http://www.iaees.org/publications/journals/nb/articles/2024-14(1)/2-Zhang-Abstract.asp

Zhang WJ. 2024b. A standalone executable software for network visualization. Network Pharmacology, 9(1-2): 1-10. http://www.iaees.org/publications/journals/np/articles/2024-9(1-2)/1-Zhang-Abstract.asp

Zhang WJ. 2024c. An executable Java software for visualizing networks. Network Biology, 14(1): 1-12. http://www.iaees.org/publications/journals/nb/articles/2024-14(1)/1-Zhang-Abstract.asp

Zhang WJ. 2024d. netGen 3.0: The executable Java software for network visualization. Selforganizology, 11(1-2): 1-27. http://www.iaees.org/publications/journals/selforganizology/articles/2024-11(1-2)/1-Zhang-Abstract.asp