MCDA

Interactive Data Visualization Tool

## Project Documentation

Table of Contents

1. Project Overview

2. System Architecture

3. User Interface Layout

4. Core Function Modules

5. MCDA Methods Detailed

6. Technical Implementation

7. User Guide

8. File Structure

# 1. Project Overview

1.1 Project Introduction

MCDA Interactive Data Visualization Tool is a web-based Multi-Criteria Decision Analysis (MCDA) interactive data visualization tool. This tool can read MCDA data in Excel format and provides real-time data adjustment, weight optimization, ranking calculation, and multiple visualization display functions.

1.2 Main Features

- Multi-file Support: Automatically detects and supports multiple Excel data files

- Real-time Interaction: Supports slider and input box real-time adjustment of data and weights

- Multiple MCDA Methods: Supports three classic MCDA methods: Weighted Sum, Compromise Programming (CP), and TOPSIS

- Data Locking: Supports locking specific data items to prevent accidental modification

- Multi-chart Display: Provides three visualization methods: bar charts, line charts, and radar charts

- Pareto Dominance Analysis: Automatically calculates and displays Pareto optimal solutions

- Python Script Integration: Supports running Python scripts to generate scatter plots and tornado diagrams

# 2. System Architecture

Technology Stack

- Frontend: React.js, D3.js, XLSX.js, Tailwind CSS

- Backend: Node.js, Express.js

- Data Processing: JavaScript, Python

- File Format: Excel (.xlsx, .xls)

# 3. User Interface Layout

3.1 Detailed Layout Description

Top Title Bar

- Project Title: MCDA Interactive Data Visualization

- Gradient background design, highlighting project professionalism

3.2 Left Control Panel (50% width)

File Selector Module

- File dropdown selection box (no suffix display)

- Refresh file button

- File status display

MCDA Method Selection Module

- Three method card-style selection

- Method parameter configuration

- Method description

Interactive Data Table Module

- Dynamic data table

- Slider and input box controls

- Data locking functionality

- Real-time ranking display

Weight Control Panel

- Weight slider and input box

- Weight locking functionality

- Total weight display

- Weight reset functionality

Pareto Dominance Analysis Module

- Pareto optimal solution display

- Dominance relationship analysis

- Advantage explanation

3.3 Right Chart Panel (50% width)

Project Value Visualization Module

- Individual project charts

- Three chart type switching

- Pareto dominance information display

Comprehensive Analysis Chart Module

- Stacked bar/line/radar charts

- Legend display

- Method-specific labels

Analysis Image Module

- Scatter plot display

- Tornado chart display

- Image zoom functionality

# 4. Core Function Modules

4.1 File Management Module

Function Description: Automatically detects and loads Excel data files

Core Features:

- Automatically scans Excel files in the data folder

- Supports .xlsx and .xls formats

- File selection dropdown (no suffix display)

- File refresh functionality

Data Reading Logic:

// Excel file structure reading

- Row A3: Column names (indicator names)

- Rows 4 to second-to-last: Data rows (indicator values for each solution)

- Last row: Weight row (weights for each indicator)

- Last column: Initial ranking

4.2 Data Interaction Module

Function Description: Provides real-time data adjustment and weight optimization functionality

Core Features:

- Slider control (0-100 range)

- Numerical input box

- Data locking functionality

- Real-time ranking updates

Interaction Logic:

// Data adjustment process

1. User adjusts slider or inputs value

2. System validates numerical range (0-100)

3. Updates corresponding data item

4. Recalculates ranking

5. Updates chart display

4.3 Weight Management Module

Function Description: Provides weight adjustment and optimization functionality

Core Features:

- Weight slider control (0-1 range)

- Weight input box

- Weight locking functionality

- Total weight constraint (sum equals 1)

- Automatic weight distribution

Weight Constraint Logic:

// Weight adjustment algorithm

1. Check locked weight sum

2. Calculate remaining available weight

3. Proportionally distribute unlocked weights

4. Ensure total weight equals 1

5. Update display and calculations

4.4 Chart Visualization Module

Function Description: Provides multiple data visualization methods

Chart Types:

- Bar Chart: Suitable for discrete data comparison

- Line Chart: Suitable for trends and continuity

- Radar Chart: Suitable for multi-dimensional data display

Chart Features:

- Responsive design

- Interactive legend

- Dynamic data updates

- Method-specific labels

4.5 Pareto Dominance Analysis Module

Function Description: Automatically calculates and displays Pareto optimal solutions

Analysis Logic:

// Pareto dominance judgment

For solutions A and B:

- If A is no worse than B in all indicators, and strictly better in at least one indicator

- Then A dominates B

- Pareto optimal solution: solution not dominated by any other solution

# 5. MCDA Methods Detailed

5.1 Weighted Sum Method

5.1.1 Mathematical Principle

The Weighted Sum Method is the most classic multi-criteria decision analysis method, calculating comprehensive scores through linear combination of indicator values.

Mathematical Formula:

S\_i = Σ(w\_j × x\_ij)

Where:

- S\_i: Comprehensive score of solution i

- w\_j: Weight of indicator j

- x\_ij: Value of solution i on indicator j

5.1.2 Implementation Logic

const calculateWeightedScore = (projectValues, weights) => {

let totalScore = 0;

Object.keys(projectValues).forEach(projectName => {

const weight = weights[projectName] || 0;

const value = projectValues[projectName] || 0;

totalScore += value \* weight;

});

return totalScore;

};

5.1.3 Characteristics

- Advantages: Simple and intuitive, high computational efficiency

- Disadvantages: Assumes linear relationships between indicators, may ignore indicator interactions

- Applicable Scenarios: Relatively independent indicators, decision makers prefer simple methods

5.2 Compromise Programming (CP)

5.2.1 Mathematical Principle

Compromise Programming is based on distance concepts, evaluating solutions by minimizing distance to ideal solution and maximizing distance to negative ideal solution.

Mathematical Formula:

CP\_i = D\_i^+ / (D\_i^+ + D\_i^-)

Where:

- D\_i^+: Distance from solution i to ideal solution

- D\_i^-: Distance from solution i to negative ideal solution

- p: Distance metric parameter (usually p=2, Euclidean distance)

Distance Calculation:

D\_i^+ = [Σ(w\_j × |x\_ij - x\_j^+|^p)]^(1/p)

D\_i^- = [Σ(w\_j × |x\_ij - x\_j^-|^p)]^(1/p)

5.2.2 Implementation Logic

const calculateCPScore = (projectValues, weights, data, p = 2) => {

let distanceToIdeal = 0;

let distanceToNegativeIdeal = 0;

projectNames.forEach(projectName => {

const weight = weights[projectName] || 0;

const value = projectValues[projectName] || 0;

// Fixed ideal point at 100, negative ideal point at 0

const ideal = 100;

const negativeIdeal = 0;

// Normalization

const normalizedValue = (value - negativeIdeal) / (ideal - negativeIdeal);

const normalizedIdeal = 1;

const normalizedNegativeIdeal = 0;

distanceToIdeal += weight \* Math.pow(Math.abs(normalizedValue - normalizedIdeal), p);

distanceToNegativeIdeal += weight \* Math.pow(Math.abs(normalizedValue - normalizedNegativeIdeal), p);

});

// CP score (lower is better)

const cpScore = distanceToIdeal / (distanceToIdeal + distanceToNegativeIdeal);

return cpScore;

};

5.2.3 Characteristics

- Advantages: Considers ideal and negative ideal solutions, more robust results

- Disadvantages: Sensitive to choice of ideal and negative ideal solutions

- Applicable Scenarios: Decision makers focus on proximity to ideal solution

5.3 TOPSIS Method (Technique for Order Preference by Similarity to an Ideal Solution)

5.3.1 Mathematical Principle

TOPSIS method ranks solutions by calculating relative closeness to ideal and negative ideal solutions.

Mathematical Formula:

C\_i = D\_i^- / (D\_i^+ + D\_i^-)

Where:

- C\_i: Relative closeness of solution i

- D\_i^+: Distance from solution i to ideal solution

- D\_i^-: Distance from solution i to negative ideal solution

5.3.2 Calculation Steps

1. Build decision matrix

2. Normalize decision matrix

3. Calculate weighted normalized matrix

4. Determine ideal and negative ideal solutions

5. Calculate distances

6. Calculate relative closeness

5.3.3 Implementation Logic

const calculateTopsScore = (projectValues, weights, data, idealType = 'benefit') => {

// Build decision matrix

const decisionMatrix = data.map(item =>

projectNames.map(projectName => item.projectValues[projectName] || 0)

);

// Normalize decision matrix (relative to fixed ideal point 100)

const normalizedMatrix = decisionMatrix.map(row => {

return row.map(val => val / 100);

});

// Calculate weighted normalized matrix

const weightedMatrix = normalizedMatrix.map(row =>

row.map((val, index) => val \* (weights[projectNames[index]] || 0))

);

// Determine ideal and negative ideal solutions

const idealSolution = [];

const negativeIdealSolution = [];

for (let j = 0; j < projectNames.length; j++) {

if (idealType === 'benefit') {

idealSolution[j] = weights[projectNames[j]] || 0;

negativeIdealSolution[j] = 0;

} else {

idealSolution[j] = 0;

negativeIdealSolution[j] = weights[projectNames[j]] || 0;

}

}

// Calculate distances

const currentIndex = data.findIndex(item =>

Object.keys(item.projectValues).every(key =>

item.projectValues[key] === projectValues[key]

)

);

if (currentIndex === -1) return 0;

const currentRow = weightedMatrix[currentIndex];

let distanceToIdeal = 0;

let distanceToNegativeIdeal = 0;

for (let j = 0; j < projectNames.length; j++) {

distanceToIdeal += Math.pow(currentRow[j] - idealSolution[j], 2);

distanceToNegativeIdeal += Math.pow(currentRow[j] - negativeIdealSolution[j], 2);

}

distanceToIdeal = Math.sqrt(distanceToIdeal);

distanceToNegativeIdeal = Math.sqrt(distanceToNegativeIdeal);

// Calculate TOPSIS score (higher is better)

const topsScore = distanceToNegativeIdeal / (distanceToIdeal + distanceToNegativeIdeal);

return topsScore;

};

5.3.4 Characteristics

- Advantages: Considers both ideal and negative ideal solutions, more comprehensive results

- Disadvantages: Relatively complex calculations, sensitive to data normalization methods

- Applicable Scenarios: Need to comprehensively consider optimal and worst-case decisions

5.4 Method Comparison

Method Computational Complexity Result Stability Applicable Scenarios Main Features

Weighted Sum Low Medium Simple decisions Intuitive and easy, simple calculation

CP Method Medium High Robust decisions Considers ideal solution distance

TOPSIS High High Complex decisions Comprehensive consideration of optimal and worst cases

# 6. Technical Implementation

6.1 Frontend Technology Stack

6.1.1 React.js

- State Management: Uses React Hooks to manage complex state

- Component Design: Modular component structure

- Responsive Updates: Real-time data updates and interface refresh

6.1.2 D3.js

- Chart Rendering: Dynamically generates various chart types

- Interactive Features: Mouse hover, click, and other interactions

- Data Binding: Data-driven chart updates

6.1.3 XLSX.js

- File Parsing: Parses Excel file formats

- Data Extraction: Extracts table data

- Format Support: Supports .xlsx and .xls formats

6.2 Backend Technology Stack

6.2.1 Node.js + Express.js

- Static File Service: Provides frontend resources

- API Interface: Provides data interfaces

- File Management: Manages Excel files and image files

6.2.2 Python Integration

- Script Execution: Runs Python data analysis scripts

- Image Generation: Generates scatter plots and tornado charts

- Data Processing: Complex data analysis tasks

# 7. User Guide

7.1 Environment Preparation

- Install Node.js (version 14.0 or higher)

- Install Python (for script execution)

- Prepare Excel data files

7.2 Application Startup

# Install dependencies

npm install

# Start server

npm start

# Access application

http://localhost:3001

7.3 Data Preparation

Excel file format requirements:

- Row A3: Column names (indicator names)

- Rows 4 to second-to-last: Data rows (indicator values for each solution)

- Last row: Weight row (weights for each indicator)

- Last column: Initial ranking

7.4 Operation Process

1. Select Data File: Choose Excel file to analyze from dropdown

2. Select MCDA Method: Choose appropriate decision analysis method

3. Adjust Parameters: Adjust relevant parameters based on method characteristics

4. Interactive Adjustment: Use sliders and input boxes to adjust data and weights

5. View Results: Observe ranking changes and chart updates

6. Analysis Images: View generated scatter plots and tornado charts

# 8. File Structure

MCDA-Tool-1.0/

├── index.html # Main page file

├── app.js # Frontend application logic (2265 lines)

├── server.js # Express server (201 lines)

├── package.json # Project configuration and dependencies

├── README.txt # Project documentation

├── data/ # Excel data file directory

│ ├── MCDA ELT V7\_Econ\_G\_Top1.xlsx

│ ├── MCDA ELT V7\_Econ\_L\_Top1.xlsx

│ └── ...

├── image/ # Generated image files

│ ├── scatter\_plots/ # Scatter plots

│ └── tornado\_diagrams/ # Tornado charts

├── src/ # Python scripts

│ ├── Scatter.py # Scatter plot generation script

│ ├── Tornado.py # Tornado chart generation script

│ └── PercentageChang.py # Percentage change script

├── static/ # Static resources

│ └── index.html # Backup main page file

└── node\_modules/ # Node.js dependency packages

# Summary

MCDA Interactive Data Visualization Tool is a comprehensive multi-criteria decision analysis tool with the following advantages:

Technical Advantages

- Modern Technology Stack: Uses React, D3.js and other modern web technologies

- Modular Design: Clear code structure and componentized design

- Strong Extensibility: Easy to add new MCDA methods and features

Functional Advantages

- Multiple MCDA Methods: Supports three classic decision analysis methods

- Real-time Interaction: Provides intuitive data adjustment interface

- Rich Visualization: Multiple chart types and display methods

- Intelligent Analysis: Automatic Pareto dominance analysis

Application Value

- Decision Support: Provides scientific basis for complex multi-criteria decisions

- Teaching Tool: Suitable for MCDA method teaching and demonstration

- Research Platform: Provides experimental platform for related research

This tool provides a modern, user-friendly, and powerful solution for multi-criteria decision analysis, effectively supporting various complex decision scenarios.