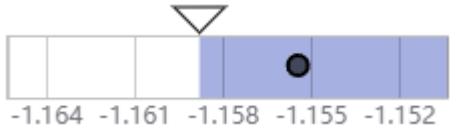
Visual components

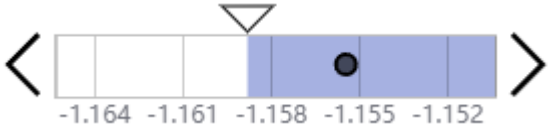
The visual folder contains mainly visualization components, which use Apache ECharts.js and Svelte.js libraries. ECharts handles the chart configuration and generation, Svelte is used to make them easily accessible as commponents.

Available components

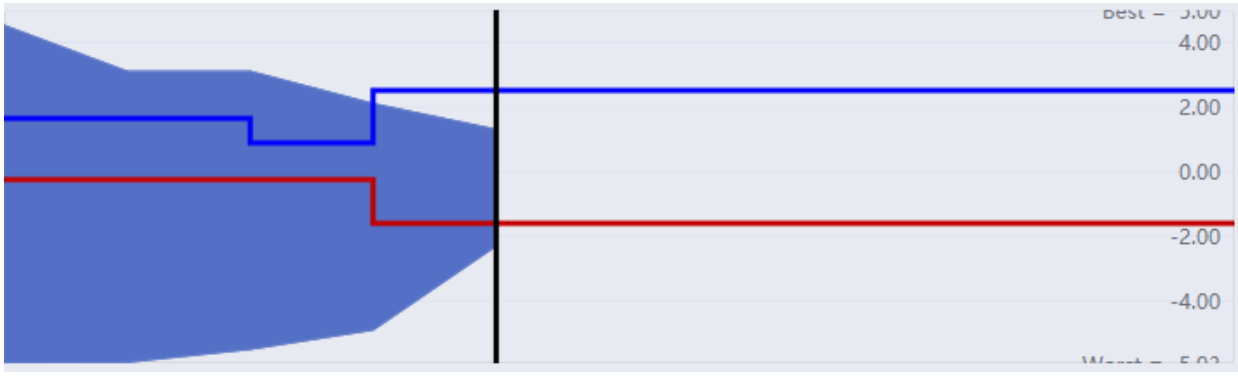
Horizontal Bar



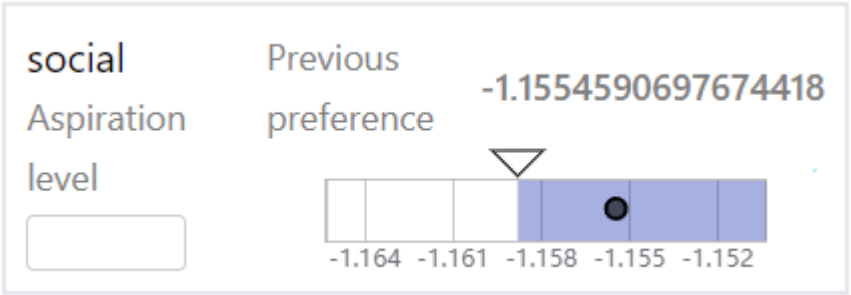
- **With arrows and input validation:**



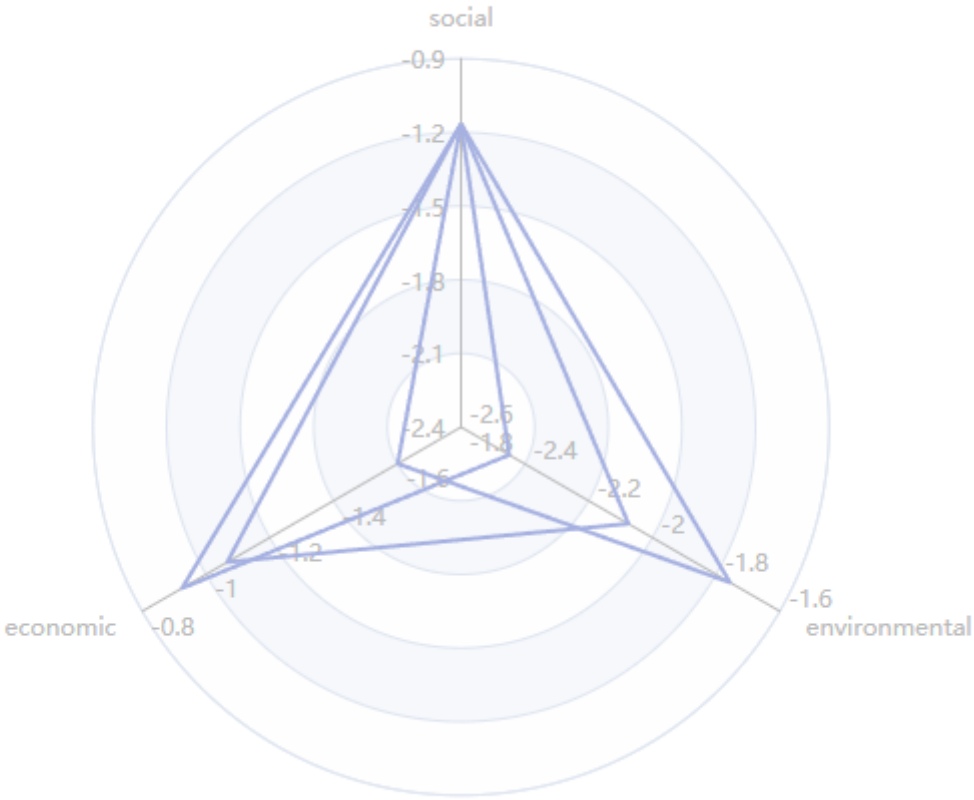
Nautilus Navigation Bar



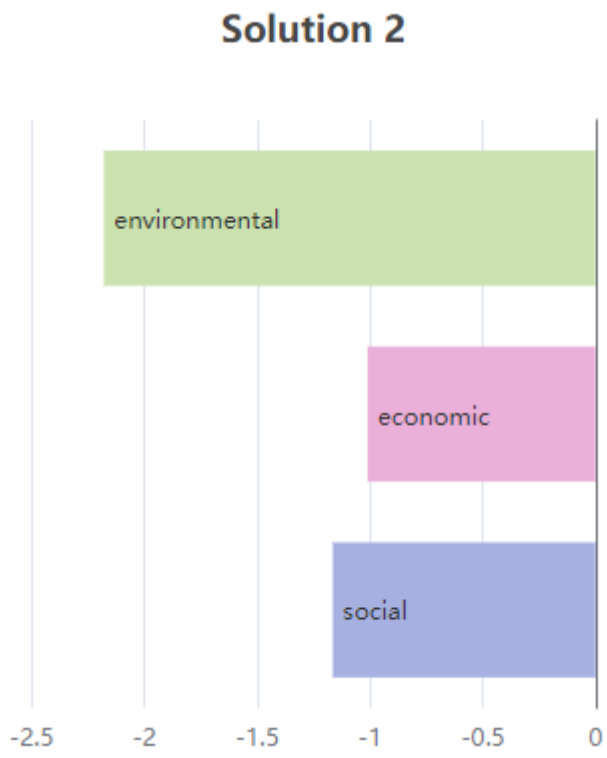
Horizontal and Nautilus bars with inputs

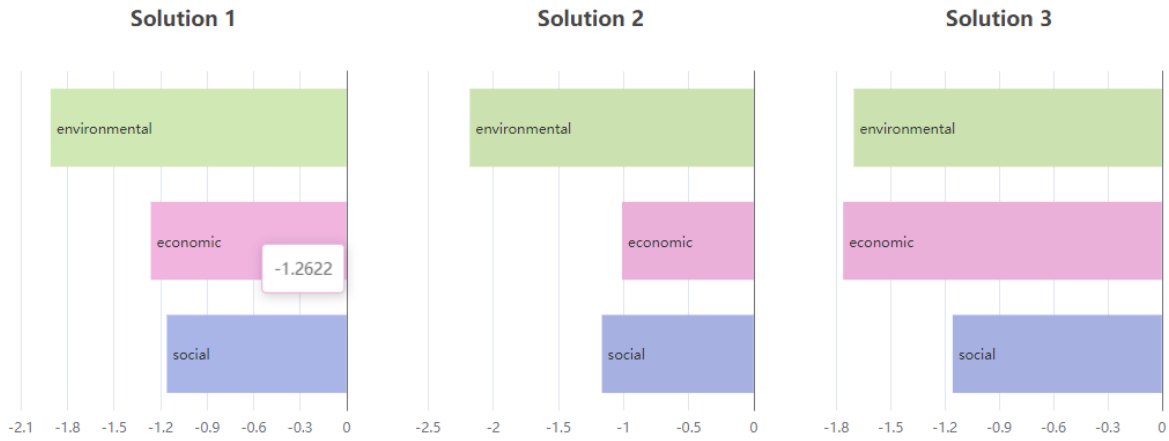


Radar Chart

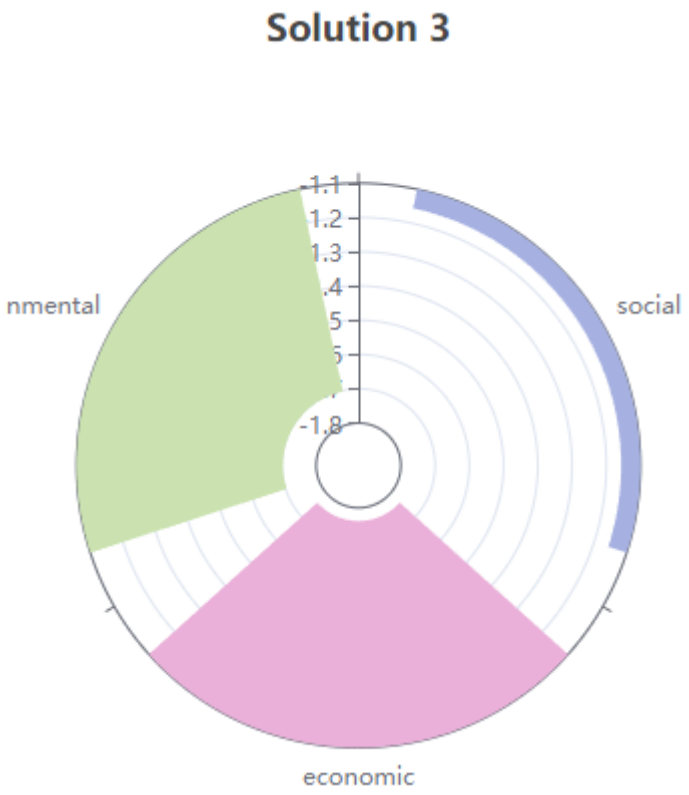

social
-0.9
-1.2
-1.5
-1.8
-2.1
-2.4
-2.6
-1.8
-2.4
-2.2
-2
-1.8
-1.6
environmental
-1.4
-1.2
-1
-0.8
economic

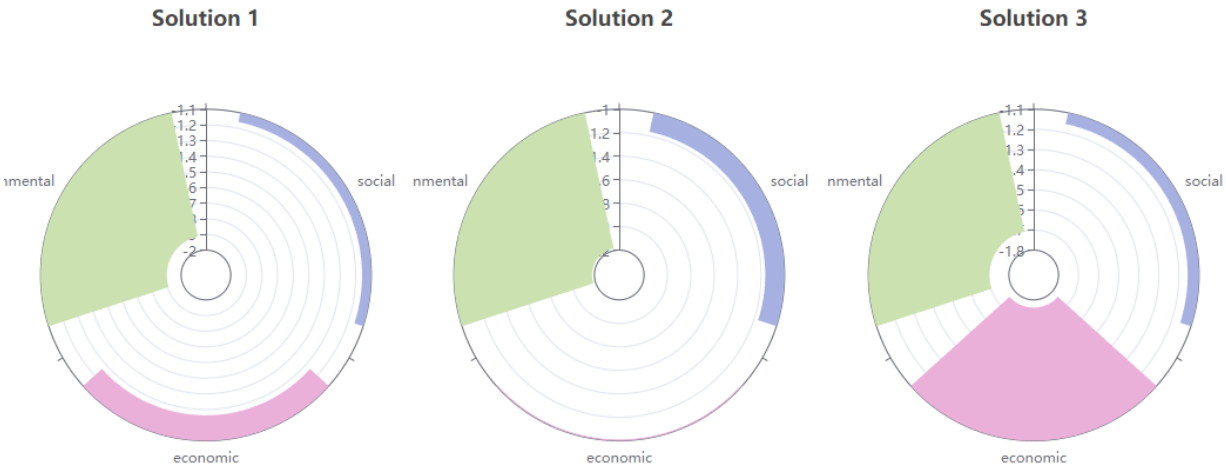
Bar Chart


Solution 2
environmental
economic
social
-2.5
-2
-1.5
-1
-0.5
0

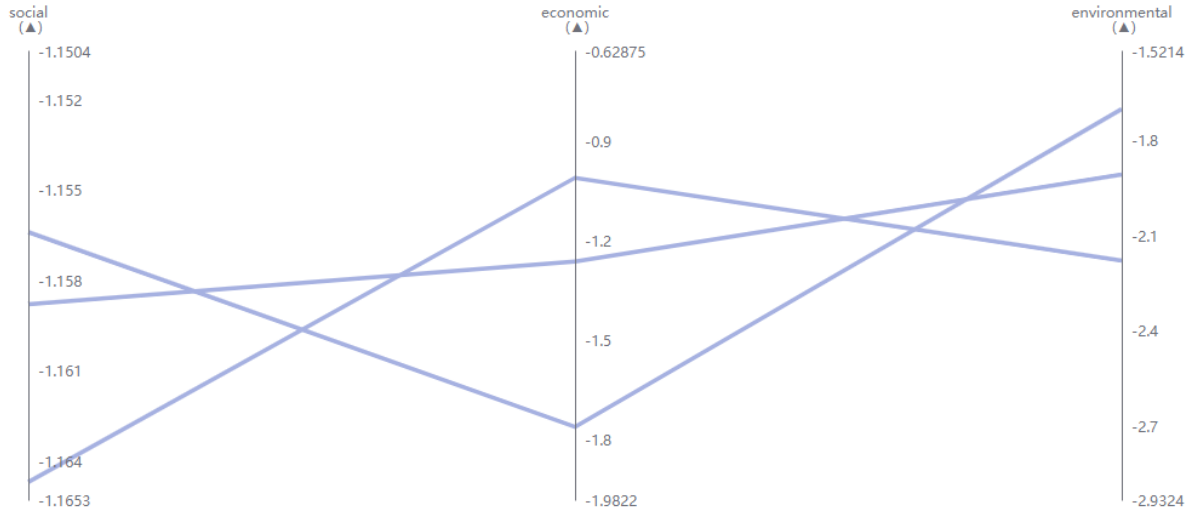

Solution 1
environmental
economic
-1.2622
social
-2.1
-1.8
-1.5
-1.2
-0.9
-0.6
-0.3
0
Solution 2
environmental
economic
social
-2.5
-2
-1.5
-1
-0.5
0
Solution 3
environmental
economic
social
-1.8
-1.5
-1.2
-0.9
-0.6
-0.3
0

Petal Chart

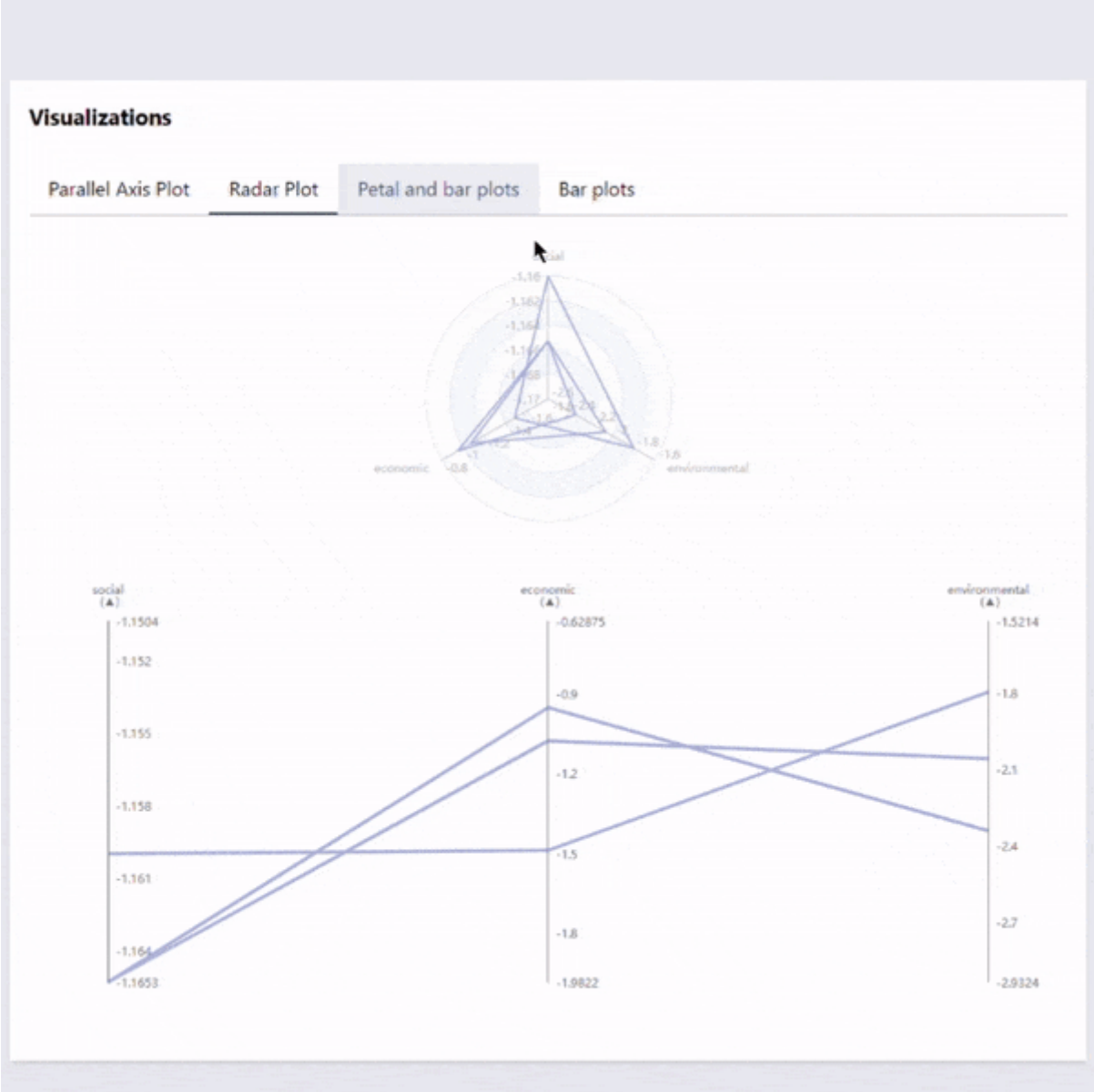




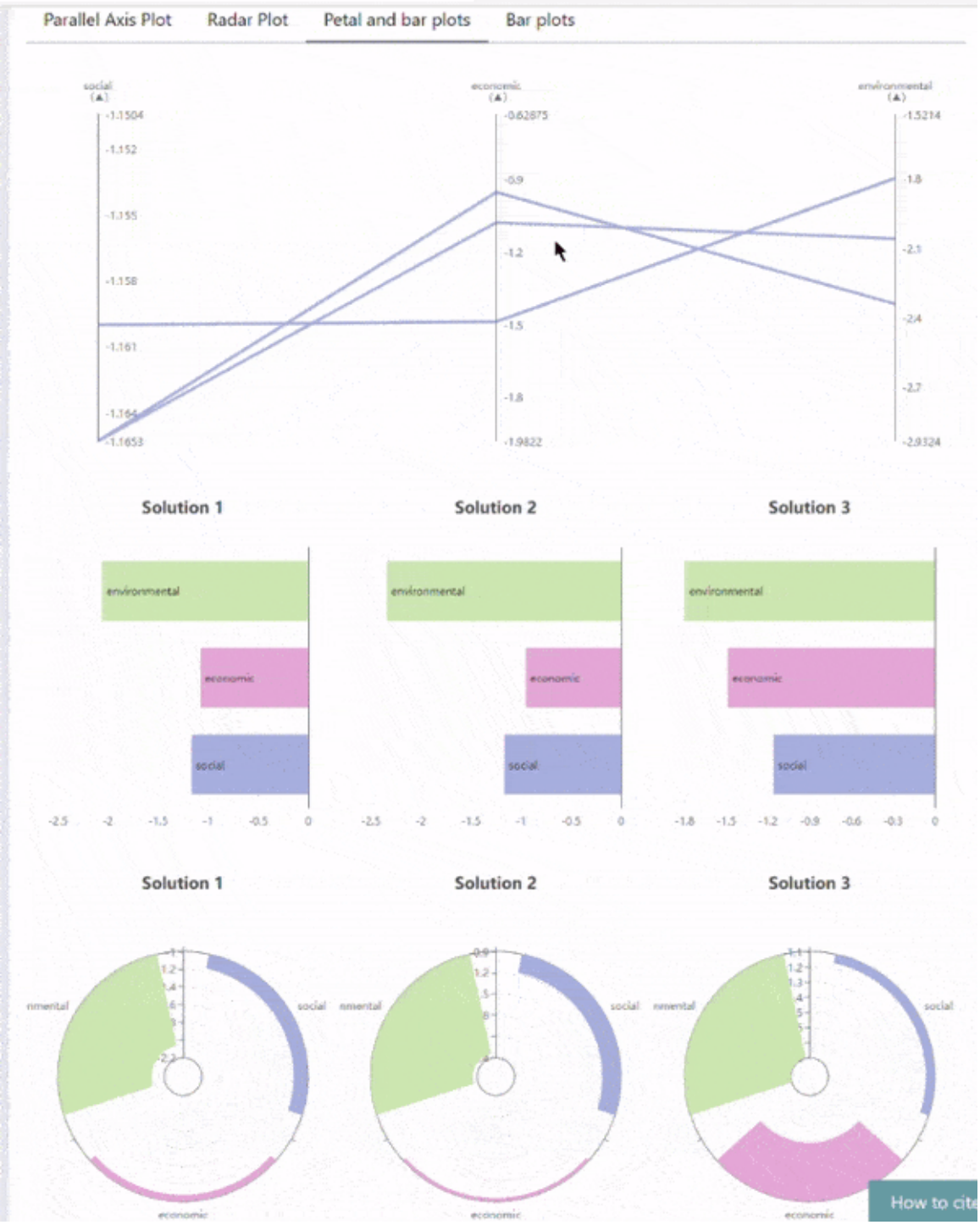
Parallel Coordinate Plot



Linking functionality:



- **Other example**



Folder structure:

visual -folder

- Contains files that the different components use, such as constants.ts and helperFunctions.ts

general -folder

- Contains the EchartsComponent.svelte, which is used to create a chart and render it. This component is used in all visualization components.

preference-interaction -folder

- This folder contains components that are used in helping the user to select the preferred value.

visualization -folder

- This folder contains the visualization components. The components are divided into two subfolders.

- **props-linking -folder**
    - Contains components that use props to link selections and highlighting with the rest of the cahrts.
    - These components don't use ECharts to link the charts, but instead use props for it.

- **multi-charts -folder**
    - Contains components that uses ECharts to render multiple charts in one component.
    - This is achieved by using the series option in ECharts.
        - Each solution is a different series object in the series array.

How to use the components

To add a component to your HTML, use it as a normal svelte component with props. The props are defined in the JSDOC format for each component. For example, if you wish to add a parallel coordinate plot, first import it:

```
import ParallelCoordinatePlotBase from
"$lib/components/visual/visualization/props-
linking/ParallelCoordinatePlot.svelte";
```

Then add the component as a HTML tag and pass the data as props:

```
<ParallelCoordinatePlot {values} names={["Objective1", "Objective2",
"Objective3", "Objective4"]} ranges={[ { min: 0, max: 10 }, { min: -2.324,
max:
10 }, { min: 0, max: 10 }, { min: undefined, max: undefined }, ]} />
```

How to make your own component

Contributing

If you wish to contribute to the visual components, please read the general instructions from the master branch README.md

Step by step

0. Create a new .svelte file in the visual folder

- Import EchartsComponent.svelte and types

```
<script lang="ts">
  import EchartsComponent from
"$lib/components/visual/general/EchartsComponent.svelte";
  import type * as echarts from "echarts";

  const option: echarts.EChartOption = {
    title: title,
  };
  // Rest of the code here
</script>

<EchartsComponent {option} />
```

1. Creating the chart

- Configure ECharts options, for reference use echarts own documentation: ECharts Docs

```
const option: echarts.EChartOption = {
  title: title,
  //Other options...
};
```

- Then use the EchartsComponent.svelte to create the chart and pass the option as a prop.

```
<EchartsComponent
  {option}
  // Other (optional) props...
/>
```

- If you want to use the created echart instance, you should bind it to a variable

```
<EchartsComponent
  {option}
  bind:chart
  // Other (optional) props...
/>
```

  - This will allow you to use the instance in the new component:

```
let chart: echarts.EChartsType;
```

2. Configuring the svelte component

- Add the props that you want to use in the component

```
export let colors: string[];
export let values: number[][];
export let selectedIndices: number[] = [];
export let highlightedIndex: number | undefined = undefined;
export let maxSelections: number | undefined = undefined;
export let disableAnimation = false;
```

- Remember to add the component in your main svelte file:

```
<MyComponent colors="{colors}" {values} disableAnimation="{true}"
/>
```