CHI 2021

Hybrid Line-Based and Region-Based Interactive Set Data Visualization

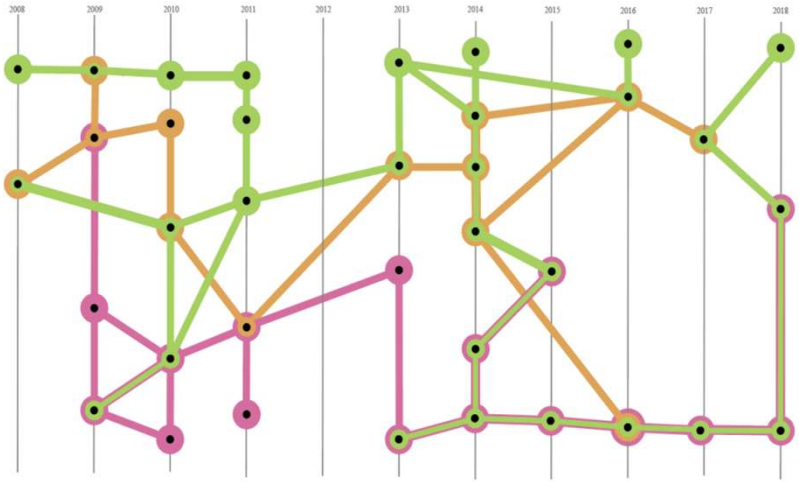
Xiaohan Wang, Chuyu Zhang , Yu Zhu, Xueyi Chen, Liming Shen, Richen Liu*, and Rongtao Qian

Nanjing Normal University

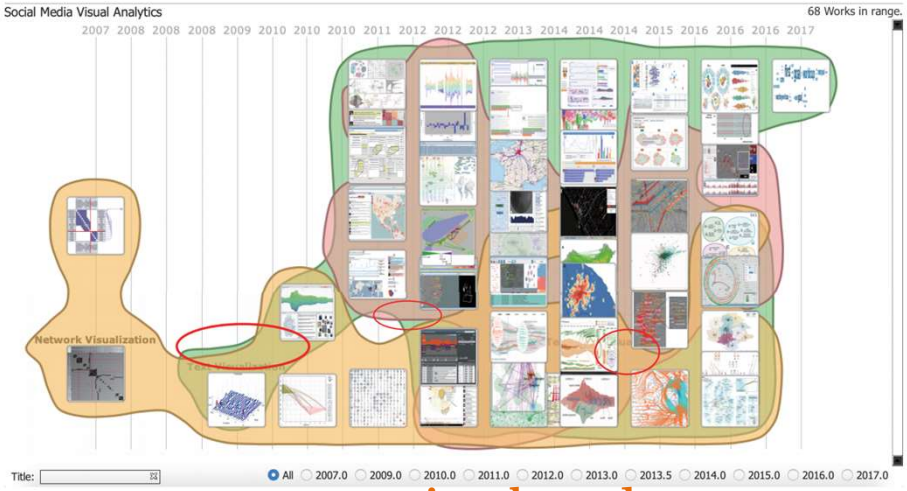
Motivation

- Data exploration is popular:
  - It is challenging to visualize all set data items
- Existing methods are imperfect:
  - The **tabular** form
  - The **line-based** scheme
  - The **region-based** scheme

tabular



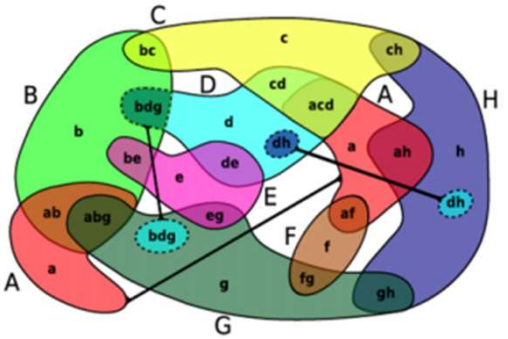
line-based



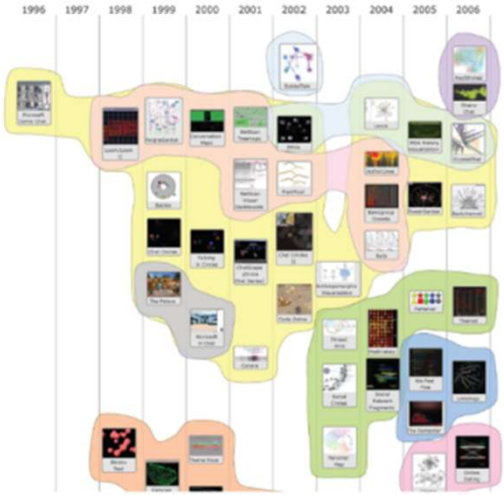
region-based

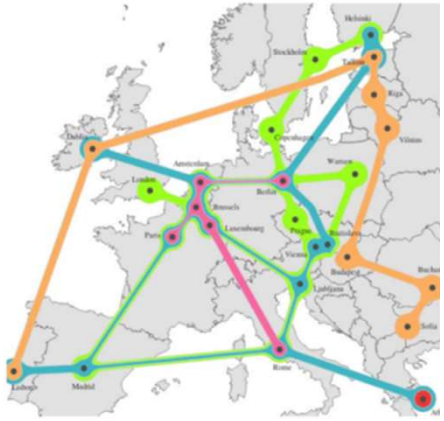
Background and Related Work

- Related Work
  - **Euler and Venn** diagrams
  - **Region-based** methods to visualize the set membership
  - **Line-based** methods to visualize the set membership



Simonetto et al



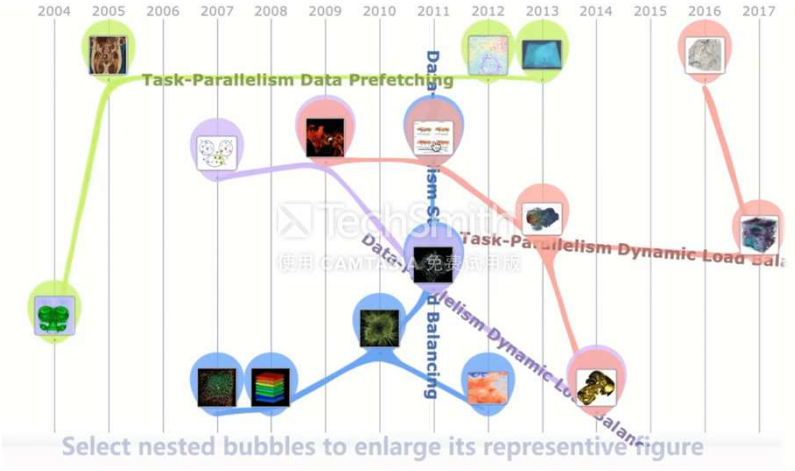


region-based overlays

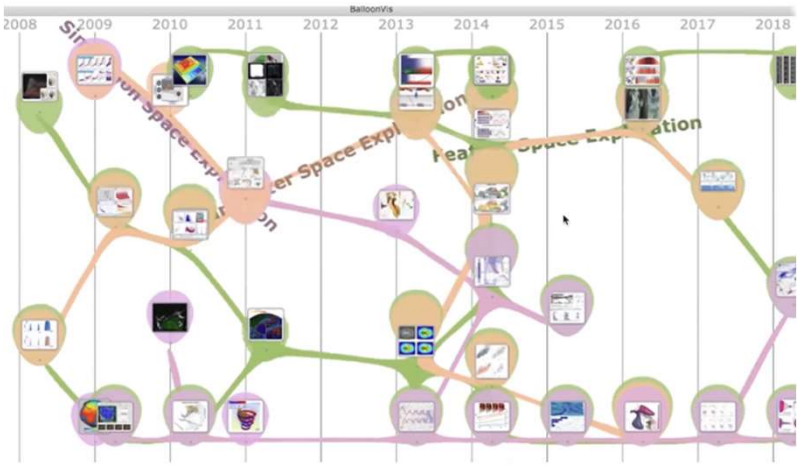
line-based overlays

Design Rationale

- Design Goals
  - Indicate the multiple categories which the set data item belongs to clearly.
  - Allow users to directly identify set data items belonging to an identical category.
  - Alleviates artefacts caused by empty overlapping regions in region-based methods without disconnected regions.
  - Avoid too much visual clutter while preserving the original layout of the timeline.



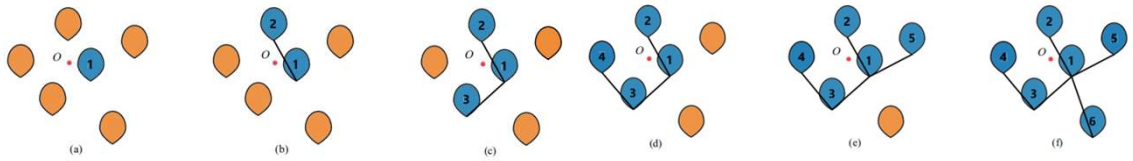
Show representative image



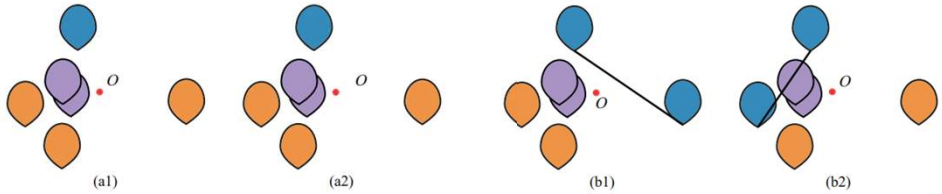
Drag items to re-layout

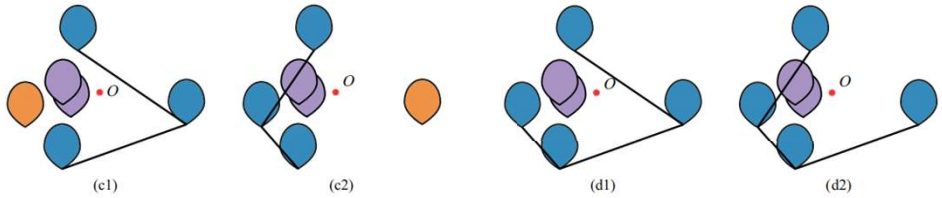
Algorithm 1: Balloon Connection

- The Connections Between Balloons



The connection process





The comparisons between the method using cost value

Algorithm 1 `Balloon_Blob_Line_Connection()` function.

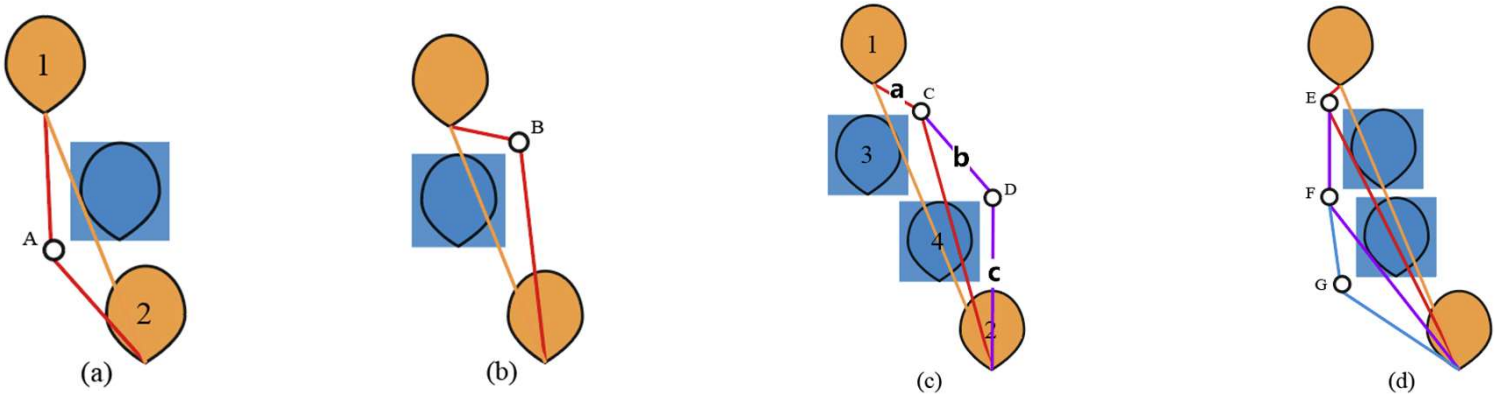
```
function BA_BLOB_LINE_CONN(start_list, end_list, sub_cato_pts[pts_size] )
    find central point C based on sub_cato_location[item_size]
    for point_iterator = 0 to pts_size do
        start_list.add(sub_cato_pts[point_iterator])
    end for
    for point_iterator = 0 to pts_size do
        cost_pts[point_iterator]=
        α*distance(start_list[point_iterator], C)
        + β*obstacles(sub_cato_pts[point_iterator], C)
    end for
    start_list.remove(item with lowest cost)
    end_list.add(item with lowest cost)
    while start_list is not empty do
        for start_iterator = 0 to start_list.size do
            for end_iterator = 0 to end_list.size do
                cost_pts[start_iterator][end_iterator]=
                α*distance(start_list[start_iterator],
                end_list[end_iterator]) + β*obstacles(start_list[start_iterator],
                end_list[end_iterator])
            end for
        end for
        find minimum cost_pts[M][N]
        Connect start_list[M] and end_list[N]
        start_list.remove(M)
        end_list.add(N)
    end while
end function
```

Algorithm 2: Line Connection

- Line Connection Algorithms
  - Consider each balloon as a rectangle
  - Add control points

Algorithm 2 `Balloon_Avoidance_Line_Conn()` function.

```
function BA_BLOB_LINE_CONN(start_pt, end_pt, control_pts_list )
    while The line from start_pt to end_pt intersects the rectangle do
        Find the smaller area A
        if The top left or right point is in A then
            control_pts_list.add(the top left or right point )
            start_pt=the top left or right point
        else
            control_pts_list.add(the bottom left or right point )
            start_pt=the bottom left or right point
        end if
    end while
end function
```



Algorithm 3: Hierarchical Merge

- Hierarchical Clustering and Merging
  - Different depths of the tree indicate the different degrees
  - The number of nodes is the number of balloons automatically merged into this level.

Algorithm 3 `Calculate_Distance()` function.

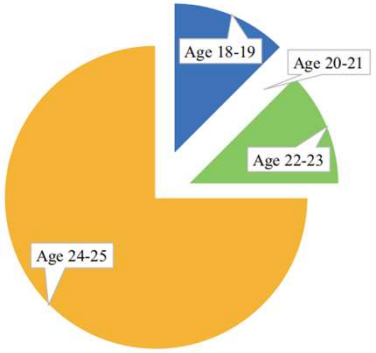
```
function CAL_DIST(possibility_cluster_a, possibility_cluster_b)
    dist←0
    for each grid point do
        possibility_a=possibility_cluster_a[grid point]
        possibility_b=possibility_cluster_b[grid point]
        dist←dist+|possibility_a - possibility_b|
    end for
    return dist
end function
```

Algorithm 4 `Balloon_Merge()` function.

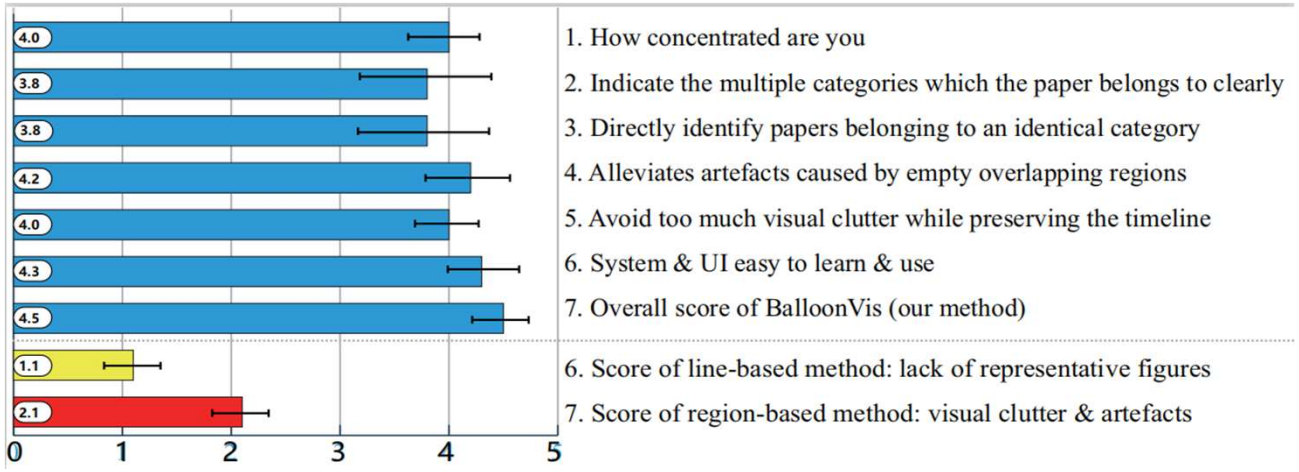
```
function BA_MERGE(ba_list, new_run_list, glyph_list )
    for ba_iterator = 0 to ba_list.size do
        cur_run_list = ba_list[ba_iterator].getRunList()
        for run_iterator = 0 to cur_run_list.size do
            if cur_is_not_new(cur_run_list[run_iterator],new_run_list) then
                new_run_list.add(cur_run_list[run_iterator])
            end if
        end for
    end for
    for ba_iterator = 0 to ba_list.size do
        glyph_list.add(ba_list[ba_iterator].getGlyph()
    end for
    new_glyph=merge_Glyph(glyph_list)
    new_balloon=construct_balloon(new_run_list, new_glyph)
    for ba_iterator = 0 to ba_list.size do
        BalloonVis.add(new_balloon)
    end for
end function
```

User Study

- We recruited 12 participants
- The participants identify literatures cited in a survey paper by:
  - Line-based method
  - Region-based method
  - Our method



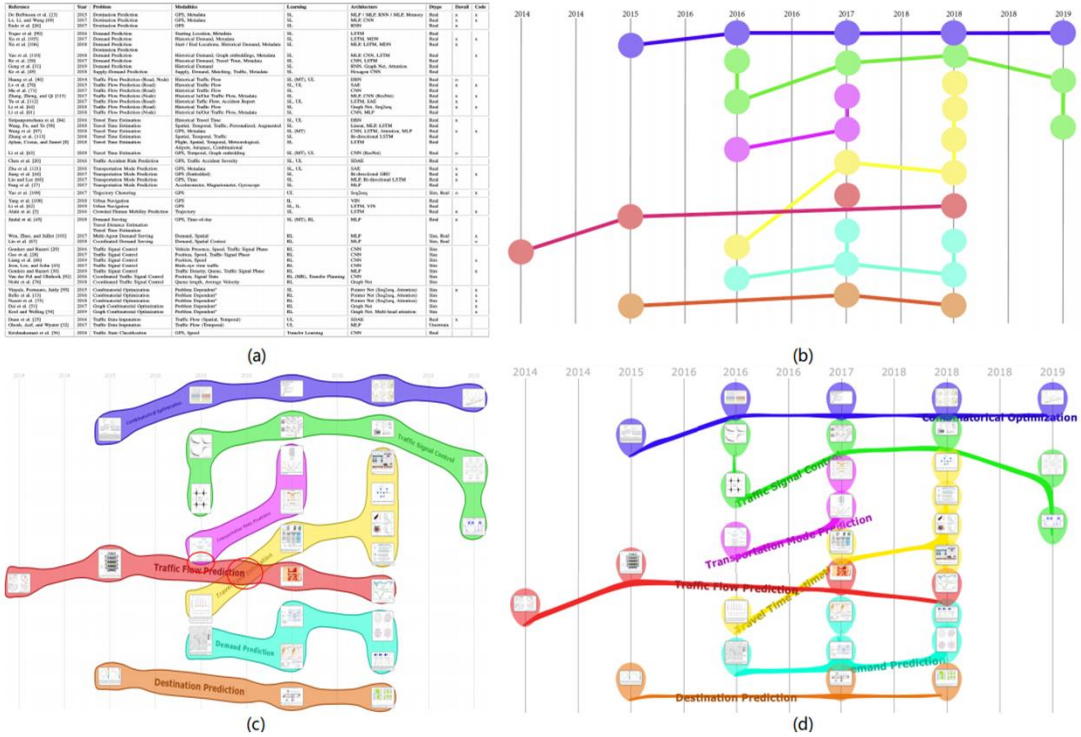
Age distribution



Post-study results

Case Study

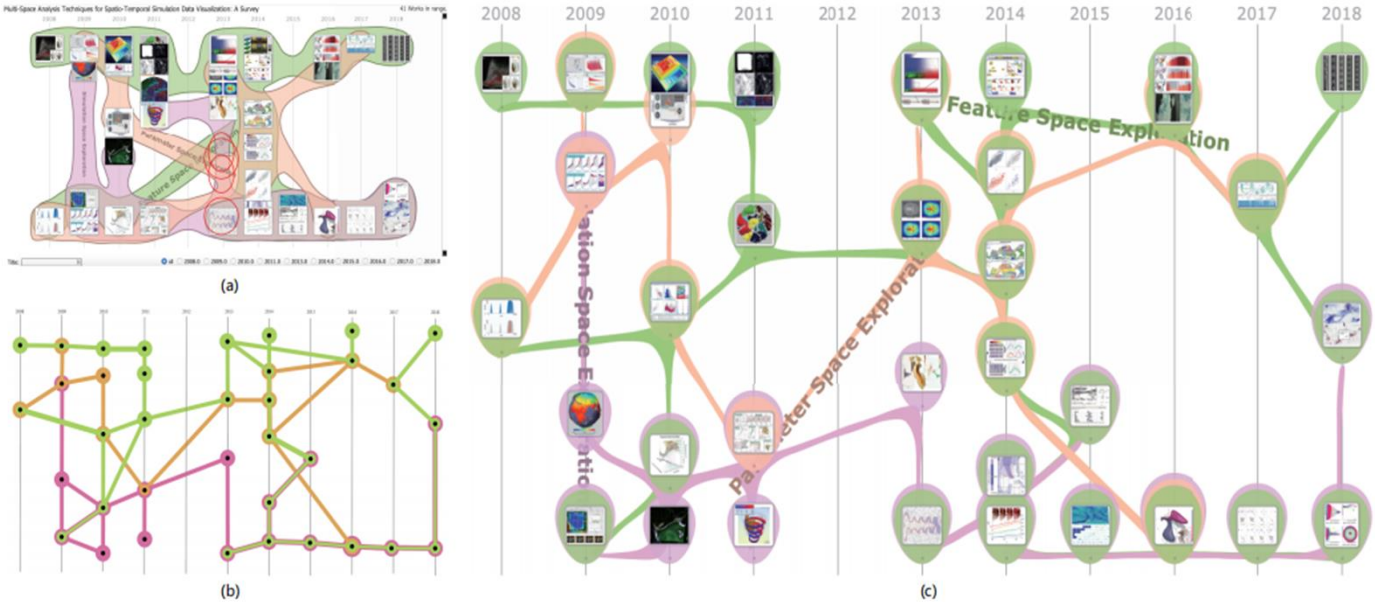
- Conduct three evaluation tests on:
  - The survey of emerging trends of deep learning methods
  - The survey on simulation data visualization
  - The survey on social media data visualization



A survey paper on "emerging trends of deep learning methods"

Case Study

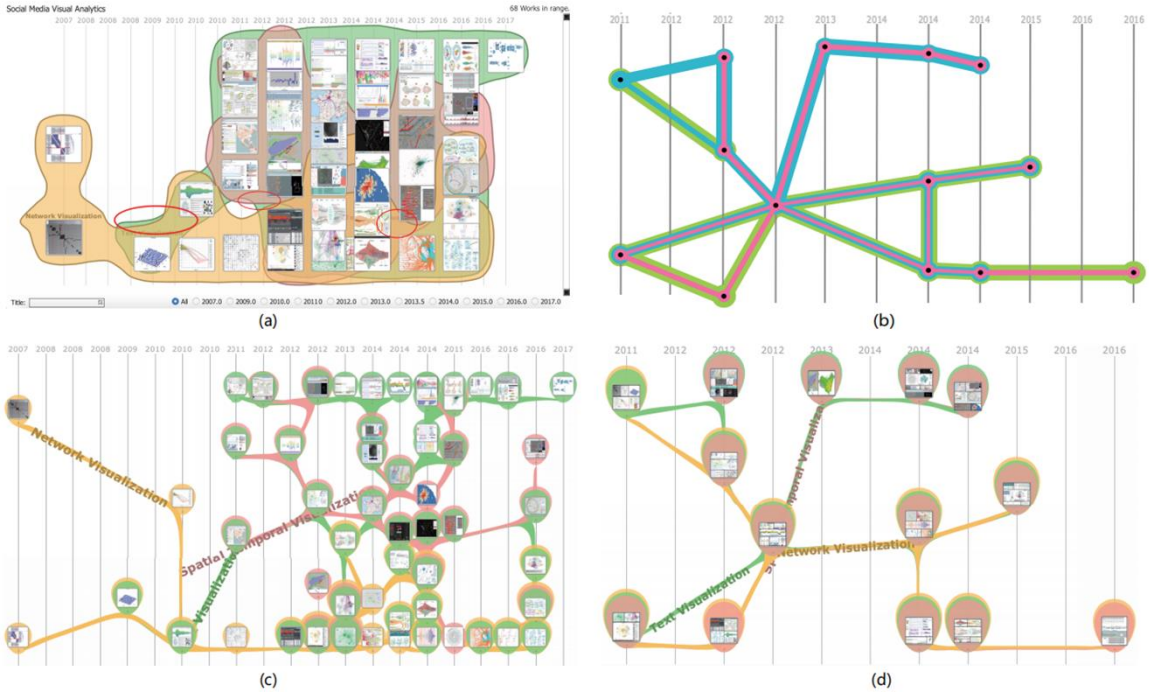
- Conduct three evaluation tests on:
  - The survey of emerging trends of deep learning methods
  - The survey on simulation data visualization
  - The survey on social media data visualization



A survey paper on "Simulation data visualization"

Case Study

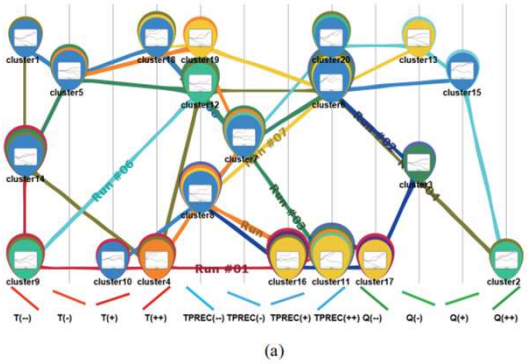
- Conduct three evaluation tests on:
  - The survey of emerging trends of deep learning methods
  - The survey on simulation data visualization
  - The survey on social media data visualization



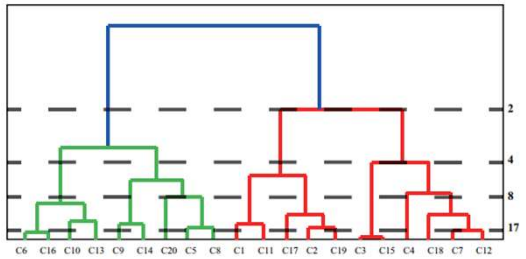
A survey on "Social media data visualization"

Case Study

GEOS-5

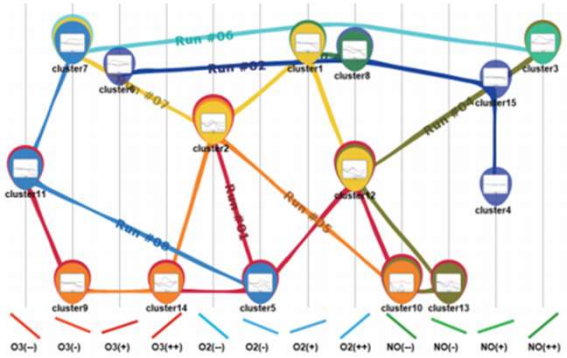


(a)

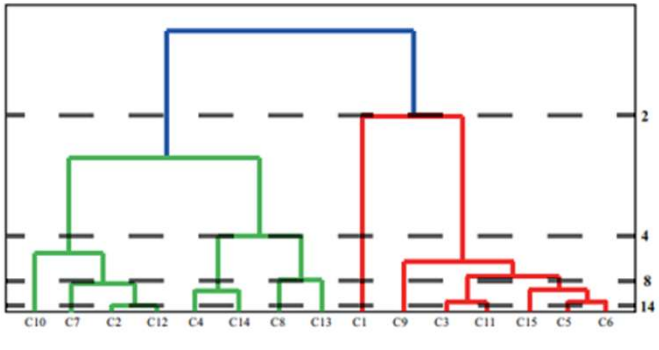


(c)

- Case study on two cases:
  - Goddard Earth Observing System Model, Version 5 (GEOS-5)
  - Model of Ozone and Related Tracers, Version 4 (MOZART-4)
- Visualize **set information** in the overlapping clusters and overlapping simulation runs



(b)



(d)

MOZART-4

Discussion and Future Work

- **Limitations**
  - Introduce visual clutter when the number of data items is too large
  - It is hard to perceive the colors when there are too many set categories
- **Future Work**
  - Enable users better control the merging processes interactively when hierarchical merging

Conclusions

- A novel method to explore set relations interactively in set data based on hybrid strategy: line-based and region-based
- Support placing the representative illustration for each set data item
- Reduce visual clutters
- The scalability is guaranteed

Acknowledgement

- The reviewers for their valuable comments
- All the user study participants
- Funding
  - National Nature Science Foundation of China (61702271)
  - Postgraduate Research & Practice Innovation Program of Jiangsu Province (SJCX20_0445)

Thank You!