Romain **ANDRÉ** - Joseph **THIBAULT** - Clément **CHAMAYOU**

ENHSP

SJR Logs Visualizer

Advanced Artificial Intelligence

TABLE OF CONTENT

1. **Context**
   a. The Project
   b. PDDL+
   c. ENHSP Solver
   d. Specifications
   e. D3.js
2. **The Visualizer**
   a. Overview
   b. Interface’s code
3. **Backend**
   a. Tree Creation
   b. Tree properties reading
   c. Json logs file structure
   d. State parsing
   e. Filling of the assignment panel
4. **Detailed Examples**
   a. HVAC
   b. Solar Rover
   c. Linear-Car-2
5. **Conclusion**

CONTEXT - The Project

Creating a tool in order to **visualize** the **decision tree** of a PDDL+ problem solved by the **ENHSP solver**

CONTEXT - PDDL+

Planning **D**omain **D**efinition **L**anguage

Language to solve automatically a problem in the planning area

Domain file

Information about the domain and the application (predicates, types, operators)

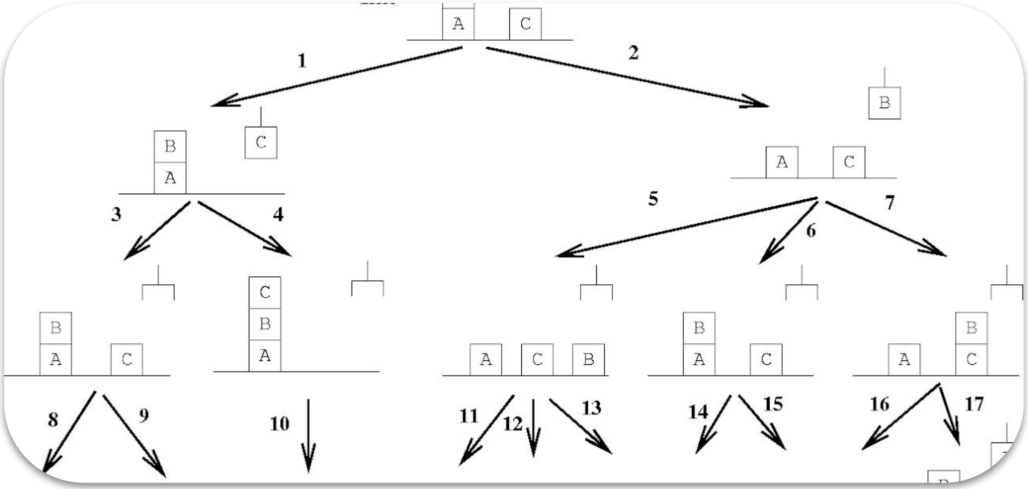
Problem file

The actual problem to solve

(initial state, goal, objects)

CONTEXT - ENHSP solver

ENHSP



CONTEXT - Specifications

1. Developed as a webpage using D3.js
2. Able to load a .sp_log file from the file explorer in the user PC
3. Shows the action chosen on each node
4. Shows the *visit_step* integer and whether a node was visited or not
5. When a node is clicked, an assignment panel should display the full assignments of that state and the distance given by the heuristic
6. Values that changed compared to the predecessor node should be highlighted and the previous value should be shown as well
7. Final path should be highlighted
8. Scrollable nodes properties panel
9. Indications on the measure of the tree
10. At first, it should display the root node and its branch
11. Possibility to expand the tree node after node
12. Possibility to expand (and retract) the entire tree with one button
13. Focus on the root node when a new file is opened
14. Available online hosted on GitHub pages

CONTEXT - D3.js

D3.js *(or D3 for* ***Data-Driven Documents****)* is a JavaScript graphics library that allows the display of numerical data in a graphical and dynamic form. It is an important tool that uses current SVG, JavaScript and CSS technologies for data visualization.

D3 uses pre-built JavaScript functions to select elements and create SVG objects. In addition, large databases with associated values can be fed to JavaScript functions to generate conditional and/or rich graphical documents.

In our case we will load and parse our **SJR JSON Logs** *(.sp_log files)*, and show the solution tree for the given problem.

THE VISUALIZER - Demo

Before looking at the code in details let’s first do a quick demonstration of the visualizer running in order to get a better understanding of the tool.

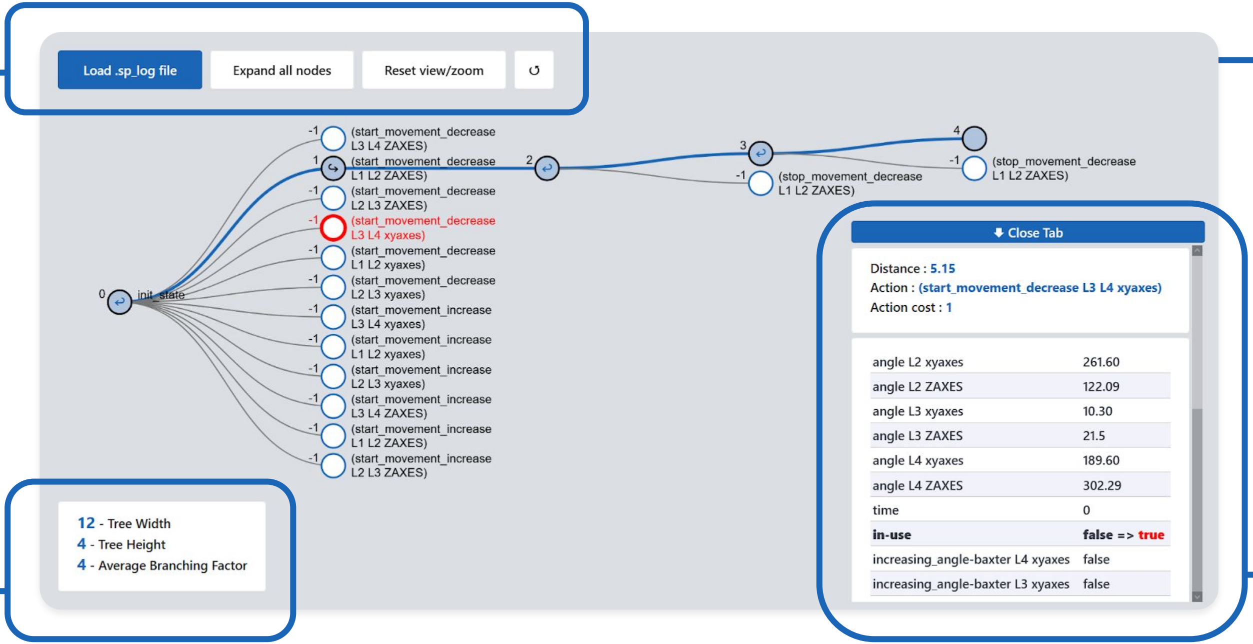
Online Tool

There will be more detailed demonstrations with clear explanations of the PDDL problems later.

THE VISUALIZER - Overview

Toolbar to control the visualizer.

The tree properties panel.


Load .sp_log file
Expand all nodes
Reset view/zoom
↺
0
init_state
-1 (start_movement_decrease L3 L4 ZAXES)
1 (start_movement_decrease L1 L2 ZAXES)
-1 (start_movement_decrease L2 L3 ZAXES)
-1 (start_movement_decrease L3 L4 xyaxes)
-1 (start_movement_decrease L1 L2 xyaxes)
-1 (start_movement_decrease L2 L3 xyaxes)
-1 (start_movement_increase L3 L4 xyaxes)
-1 (start_movement_increase L1 L2 xyaxes)
-1 (start_movement_increase L2 L3 xyaxes)
-1 (start_movement_increase L3 L4 ZAXES)
-1 (start_movement_increase L1 L2 ZAXES)
-1 (start_movement_increase L2 L3 ZAXES)
2
3
-1 (stop_movement_decrease L1 L2 ZAXES)
4
-1 (stop_movement_decrease L1 L2 ZAXES)
Close Tab

12 - Tree Width
4 - Tree Height
4 - Average Branching Factor

Distance : 5.15
Action : (start_movement_decrease L3 L4 xyaxes)
Action cost : 1

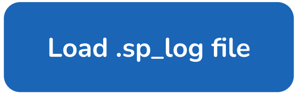
angle L2 xyaxes	261.60
angle L2 ZAXES	122.09
angle L3 xyaxes	10.30
angle L3 ZAXES	21.5
angle L4 xyaxes	189.60
angle L4 ZAXES	302.29
time	0
in-use	**false => true**
increasing_angle-baxter L4 xyaxes	false
increasing_angle-baxter L3 xyaxes	false

The main canvas, displaying the tree.

The node properties panel.

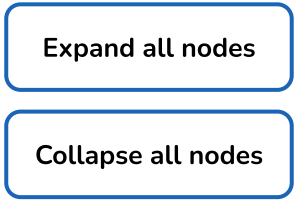
THE VISUALIZER - Toolbar

LOAD LOG FILE



The first button allows you to **load a JSON log file** for visualization. When you click on this button, you will be prompted to select the .sp_log file that you generated using ENHSP.

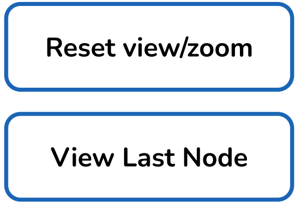
EXPAND NODES



These two buttons allow you to **expand and collapse all nodes in the tree**, respectively. This can be useful for quickly navigating the tree and exploring its structure.

The collapse all nodes button reveals itself when hovering the former one.

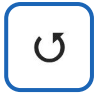
RESET ZOOM



This button **resets the zoom level to its default state, and refocuses the view back to the root of the tree**, which can be helpful if you have made changes to the visualization that you would like to undo.

View Last Node **centers the view on the node with the highest visit_step**.

SWAP CONTROLS



By default, **the left mouse button** is used to **open the node properties**, while the **right mouse button** or left mouse button while holding the shift key is used to **expand a node**. However, some users may find it more natural to reverse these controls, and this button allows you to do so.

THE VISUALIZER - Tree Properties

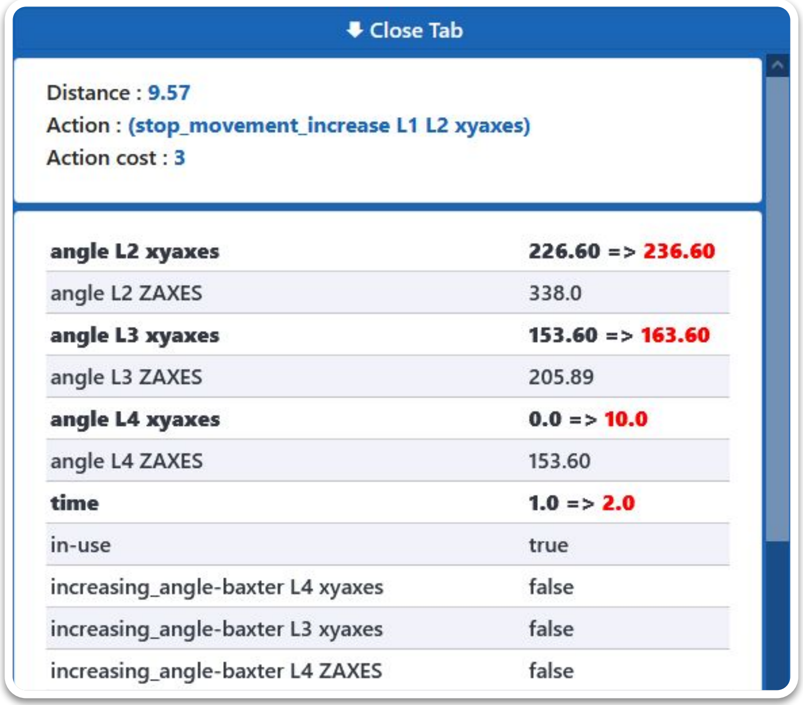
16 - Tree Width

8 - Tree Height

3.2 - Average Branching Factor

- **The tree width** corresponds to the maximum number of nodes in any level of the solution tree.
- **The tree height** corresponds to the number of levels in the solution tree.
- **The average branching factor** corresponds to the average number of children that each node in the tree has.

THE VISUALIZER - Node Properties



Distance : 9.57
Action : (stop_movement_increase L1 L2 xyaxes)
Action cost : 3

angle L2 xyaxes	**226.60 => 236.60**
angle L2 ZAXES	338.0
angle L3 xyaxes	**153.60 => 163.60**
angle L3 ZAXES	205.89
angle L4 xyaxes	**0.0 => 10.0**
angle L4 ZAXES	153.60
time	**1.0 => 2.0**
in-use	true
increasing_angle-baxter L4 xyaxes	false
increasing_angle-baxter L3 xyaxes	false
increasing_angle-baxter L4 ZAXES	false

THE VISUALIZER - Interface’s code

HTML File

Main architecture of the website

CSS File

Style of the website

JS File

Functionality of the website

THE VISUALIZER - JS Interface Functions

Update (main graphical function)

- Add the hover action to each nodes (change of style)
- Add the on click *(and right click)* actions to each nodes
    - action(focus, d, action2), picks the correct behavior depending which action (left or right click) and whether the controls are inverted or not.
        - propertiesPanel(d), open the node properties panel for the node **d** + change of style.
        - fold(d)/unfold(d), shows/hides a node's children + change of style.
- Updates the style of each nodes (if node is visited or not)
- Highlights the main plan path
- Adds text indicator for each visit step and actions (actions text is split if too long)

To highlight the main plan path

- getLastVisitedNode(), sorts all nodes by visit_step order and returns the last one.
- getPlanPath(lastNode), recursively finds the path all the way to the root from the given node.
- on update(), if path belongs to the main plan path, it is colored.

THE VISUALIZER - JS Interface Functions

Zoom

- zoomed() callback called on D3 zoom event
  - Locks zoom to a min and a max value.
- resetZoom() function allows zoom to reset and recenters view on root
  - Called when button pressed or on new tree load.

View Last Node

- findNodePosition(node) returns the node position in the canvas (accounting for default zoom).
- focusLastNode() called by the button, resets zoom and centers view on last visit_step node

Unfold nodes

- unfold(d), fold(d) called to hide or show a node's children.
- collapse(d), uncollapse(d) called to hide or show ALL node's children recursively.
- displayAllNodes(), foldAllNodes() calls collapse/uncollapse on the root node, called by pressing the corresponding buttons.

THE VISUALIZER - JS Interface Functions

Other button calls

- toggleReversedActions(), swap the main and second action, called when the correspond button is pressed
- closePanel(), closes the node properties panel, called when the corresponding button is pressed.

BACKEND - Tree creation

First listener

- Sets some parameters for the canvas (width, height, disable default d3.js right click and double click behaviors)
- Sets the size of each node

Second listener inside the first one

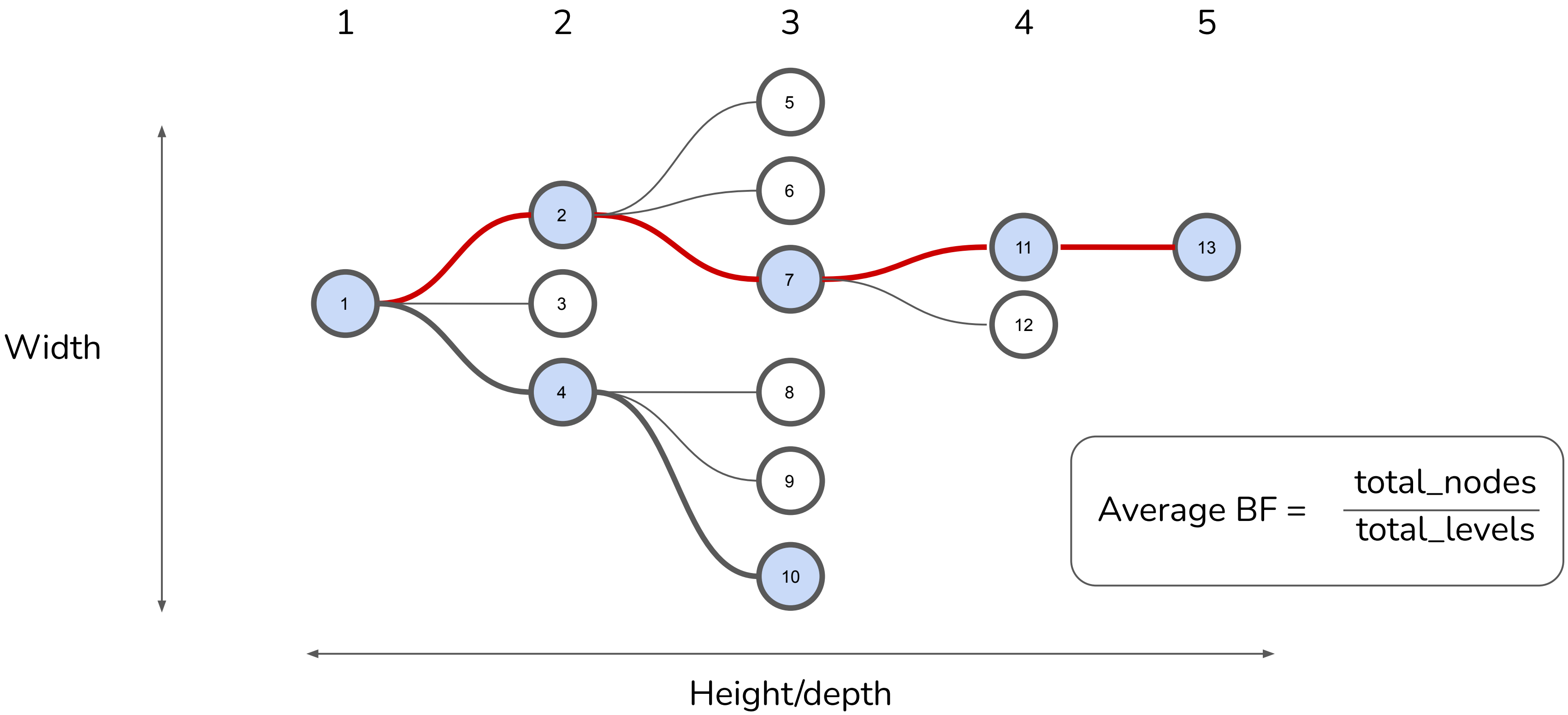
- Wait for the user to load a file
- Calls the loadTree() function
- Make new buttons appear (Expand all nodes, Reset zoom ...) and others disappear (ENHSP link) using the corresponding CSS attributes

BACKEND - Tree creation

Loading the tree

- Empties the node properties panel
- Creates the tree hierarchy using d3.js function d3.hierarchy()
- Calls the function treeInfo() in order to retrieve the dimensions of the tree and to display them
- Gets the final path using getPlanPath(lastNode)
- Resets zooms and centers view on the root of the tree
- Closes the node properties panel
- Fold all the nodes

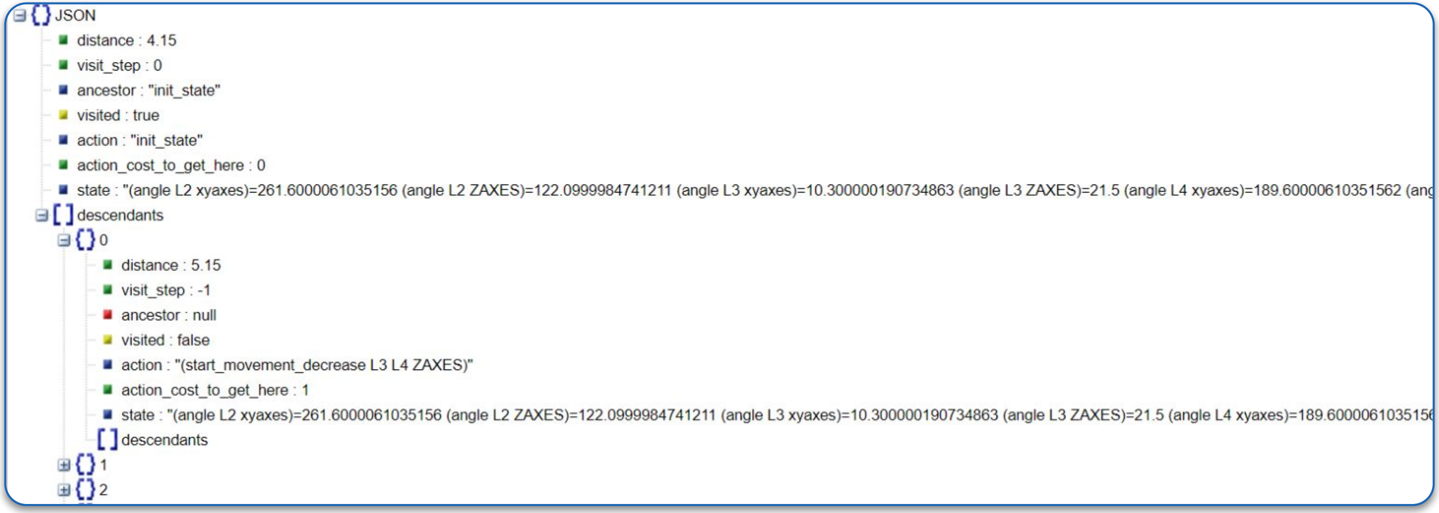
BACKEND - Tree properties reading



$$\text{Average BF} = \frac{\text{total\_nodes}}{\text{total\_levels}}$$

BACKEND - .sp_log Files

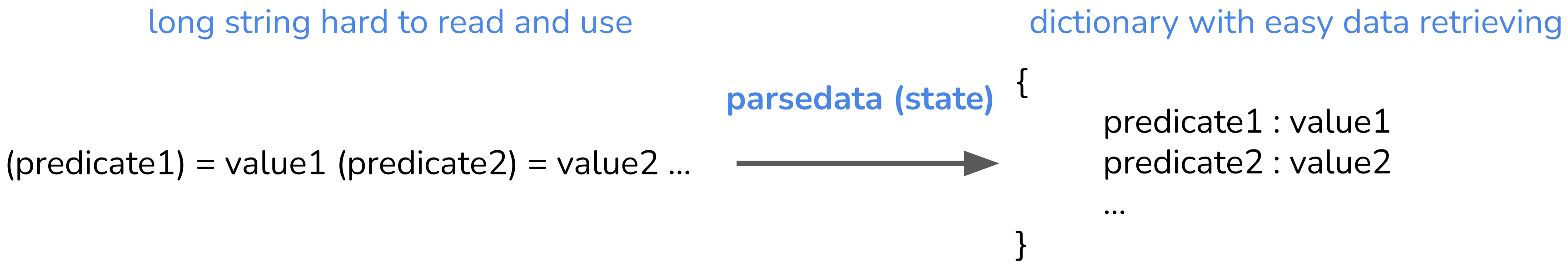
- Output of ENHSP solving (when run with the `-sjr` argument)
- Very large JSON file containing the search trace



```
⊟ {} JSON
    distance : 4.15
    visit_step : 0
    ancestor : "init_state"
    visited : true
    action : "init_state"
    action_cost_to_get_here : 0
    state : "(angle L2 xyaxes)=261.6000061035156 (angle L2 ZAXES)=122.0999984741211 (angle L3 xyaxes)=10.300000190734863 (angle L3 ZAXES)=21.5 (angle L4 xyaxes)=189.6000061035156 (ang
  ⊟ [] descendants
    ⊟ {} 0
        distance : 5.15
        visit_step : -1
        ancestor : null
        visited : false
        action : "(start_movement_decrease L3 L4 ZAXES)"
        action_cost_to_get_here : 1
        state : "(angle L2 xyaxes)=261.6000061035156 (angle L2 ZAXES)=122.0999984741211 (angle L3 xyaxes)=10.300000190734863 (angle L3 ZAXES)=21.5 (angle L4 xyaxes)=189.6000061035156
        [] descendants
    ⊞ {} 1
    ⊞ {} 2
```

Example of a .sp_log file

BACKEND - State Parsing



long string hard to read and use

(predicate1) = value1 (predicate2) = value2 ...

parsedata (state)

dictionary with easy data retrieving

```
{
        predicate1 : value1
        predicate2 : value2
        ...
}
```

BACKEND - Filling of the assignment panel

- Assignment panel: Interface panel that displays information about the node, such as predicates.

1. Gather the predicate table by its id
2. Empties it
3. For each key of the dictionary
    a. Insert predicate name
    b. if parent data = current data then insert “**current data**”
    c. if parent data != current data then insert “**parent data =>** **current data**”

EXAMPLE 1 - HVAC

The domain - HVAC

- **Types** : room, request
- **Predicates** : alwaysfalse, satisfied ?r
- **Functions** : air-flow ?l, temp ?l, temp_sa ?l, time, time_requested ?l ?r, temp_requested ?l ?r, comfort
- **Constraints:** temperature_domain ?l
- **Actions** : satisfier ?l ?r, increase_air_flow ?l, decrease_air_flow ?l, increase_temp ?l, decrease_temp ?l
- **Processes**: time_passing, thermal_change ?l

with:
?l - room
?r - request

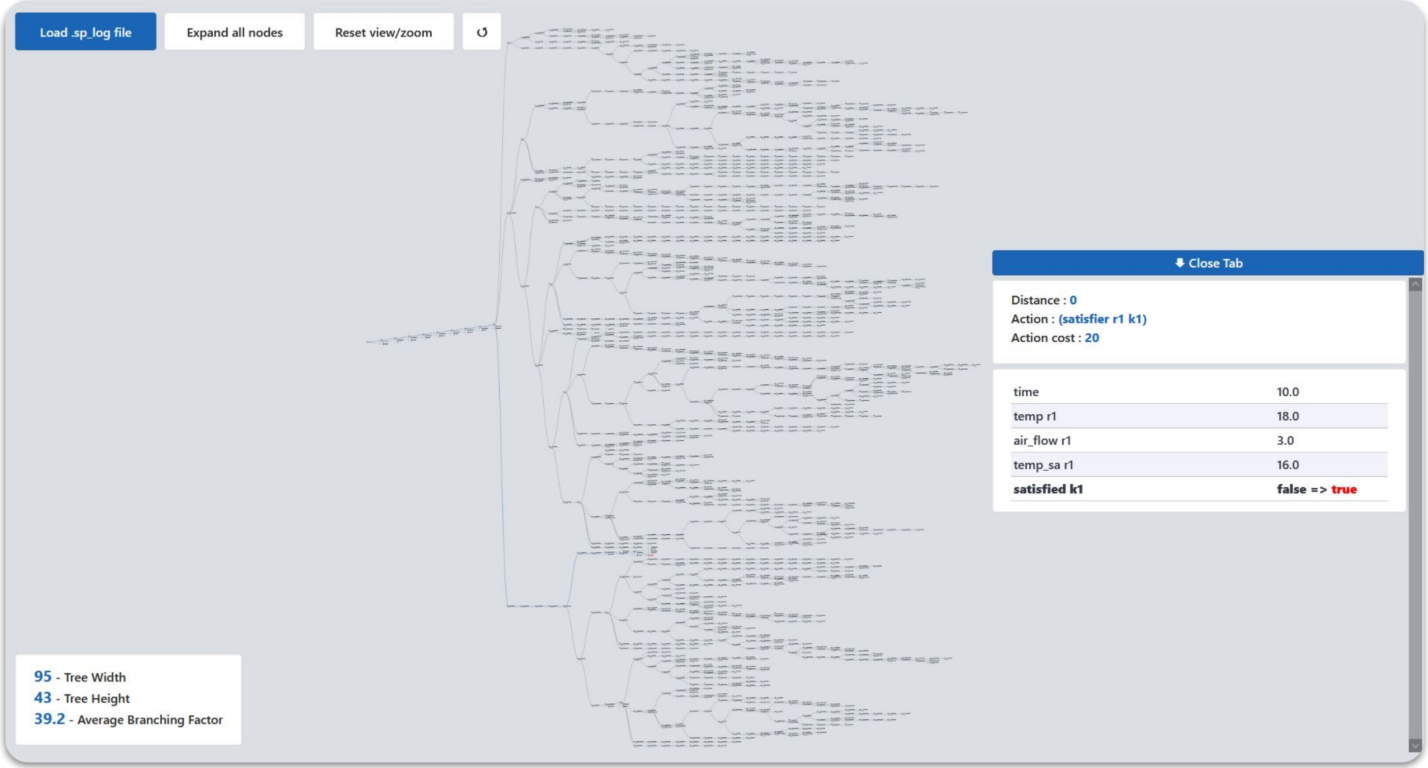
EXAMPLE 1 - HVAC

The problem

- **Objects**: room r1, request k1
- **Init state**:
  - ○ (const) time_requested r1 k1 = 10
  - ○ (const) temp_requested r1 k1 = 20
  - ○ (const) comfort = 2
  - ○ time = 0
  - ○ temp r1 = 15
  - ○ air_flow r1 = 0
  - ○ temp_sa r1 = 10
  - ○ satisfied k1 = False
- **Goal:** satisfied k1
- **Metric:** No metric specified

with:
r1 - room
k1 - request

EXAMPLE 1 - HVAC



Distance : 0
Action : (satisfier r1 k1)
Action cost : 20

time	10.0
temp r1	18.0
air_flow r1	3.0
temp_sa r1	16.0
satisfied k1	**false => true**

95 - Tree Width
43 - Tree Height
39.2 - Average Branching Factor

EXAMPLE 2 - Solar rover

The domain

- **Types** : generalBattery, battery
- **Predicates** : alwaysfalse, gboff, gbon, off, on, night, sunexposure, solarsupport, datatosend, datasent and roversafe
- **Functions** : roverenergy, SoC, time, sunexposure_time
- **Actions** : switchGenBatteryOn, start_useBattery, sendData
- **Processes**: useBattery, passingTime
- **Events**: end_useBattery, sunshine, sunexposure_event

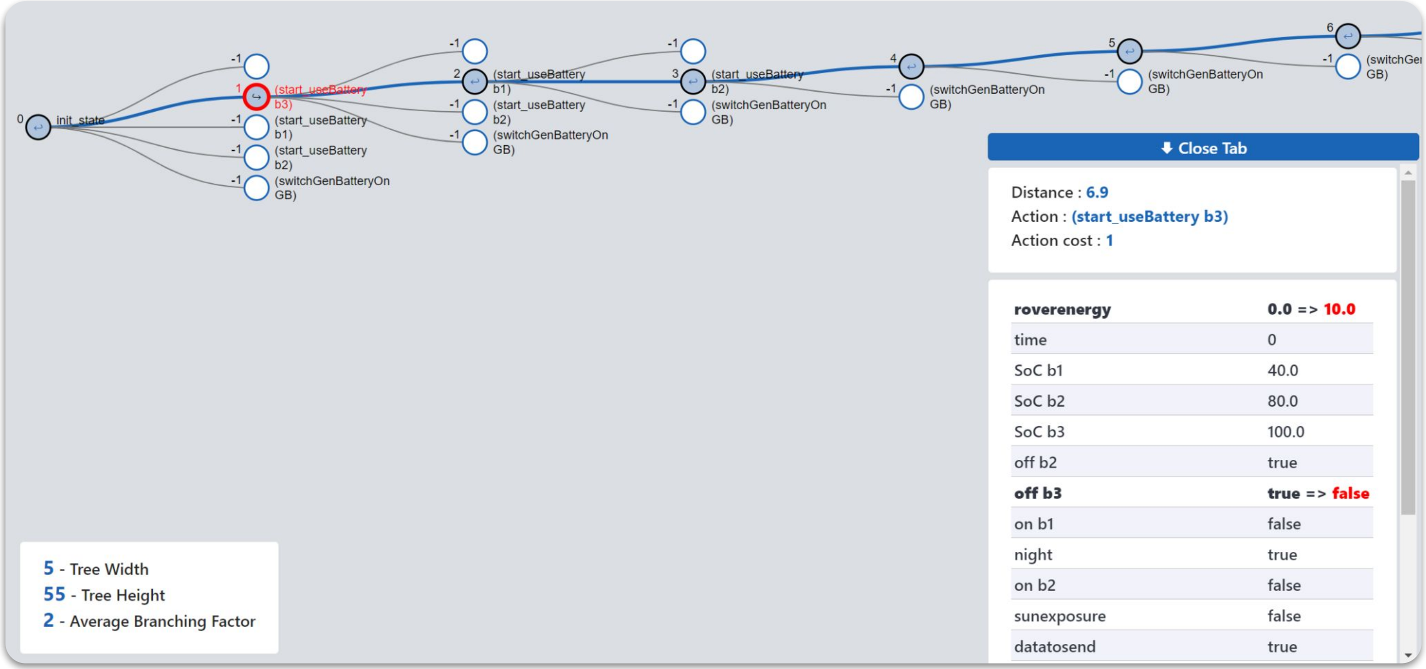
EXAMPLE 2 - Solar rover

The problem

- **Objects** : 1 general battery, 3 batteries
- **Init state**:
  - The rover's energy level is initially 0.
  - It is night time.
  - Data needs to be sent.
  - The general battery is off.
  - Battery b1 has a state of charge of 40 and is off.
  - Battery b2 has a state of charge of 80 and is off.
  - Battery b3 has a state of charge of 100 and is off.
  - The time since the start of the mission is 0.
  - The sunexposure_time is set to 50.
- **Goal** : send data
- **Optimization**: minimize the total time

EXAMPLE 2 - Solar rover

The visualization



Distance : **6.9**
Action : **(start_useBattery b3)**
Action cost : **1**

roverenergy	**0.0 => 10.0**
time	0
SoC b1	40.0
SoC b2	80.0
SoC b3	100.0
off b2	true
off b3	**true => false**
on b1	false
night	true
on b2	false
sunexposure	false
datatosend	true

5 - Tree Width
55 - Tree Height
2 - Average Branching Factor

EXAMPLE 3 - Linear-Car-2

The domain

- **Predicates** : engine_running, engine_stopped
- **Functions** : d, v, a, max_acceleration, min_acceleration, max_speed
- **Actions** : accelerate, stop_car, start_car, decelerate
- **Processes**: displacement, moving
- **Events**: idle

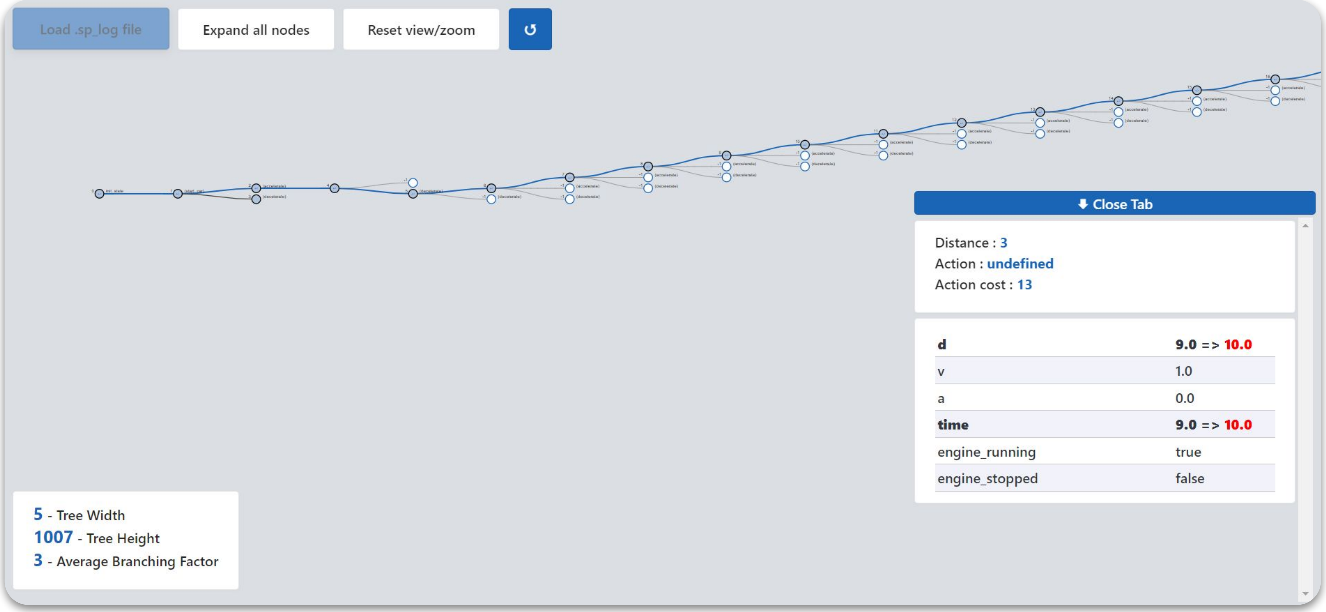
EXAMPLE 3 - Linear-Car-2

The problem

- **Init state**:
  - The car is not moving and its engine is stopped
  - Car’s displacement is 0
  - Car’s velocity is 0
  - Car’s acceleration is 0
  - Car’s maximum acceleration is 1
  - Car’s minimum acceleration is -1
  - Car’s maximum speed is 100
- **Goal** : the car needs to stop at a specific distance in the range [1000.5; 1001.5] and have its engine stopped
- **Metric :** no metric specified

EXAMPLE 3 - Linear-car-2

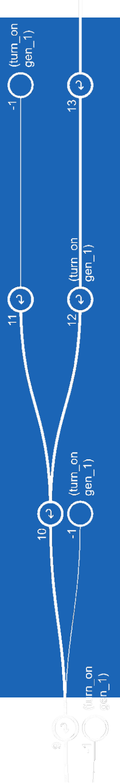
The visualization



Distance : 3
Action : undefined
Action cost : 13

d	**9.0 => 10.0**
v	1.0
a	0.0
time	**9.0 => 10.0**
engine_running	true
engine_stopped	false

5 - Tree Width
1007 - Tree Height
3 - Average Branching Factor



CONCLUSION