Planning for a train scheduling

Mattia Pannone 1803328

1. Problem description

In this problem I want to schedule the trips of different train traveling on common rails to reach stations or railways, without them being at the same time on the same piece of track thus causing a blockage of the line, in fact if two train are on the same piece of track no one can proceed, given that once left, each train can proceed only forward until reaching the goal, it can never go back. For "piece of track" I mean a rail which is located between two nodes and a node is a point of the track where two or more tracks are joined. The problem can be initialized with a number N of station, M trains and K lines between the stations, the goal is to find a plan where each train is moved between the railways one at time.
The process I use to produce a plan is illustrated in [Fig. 1].

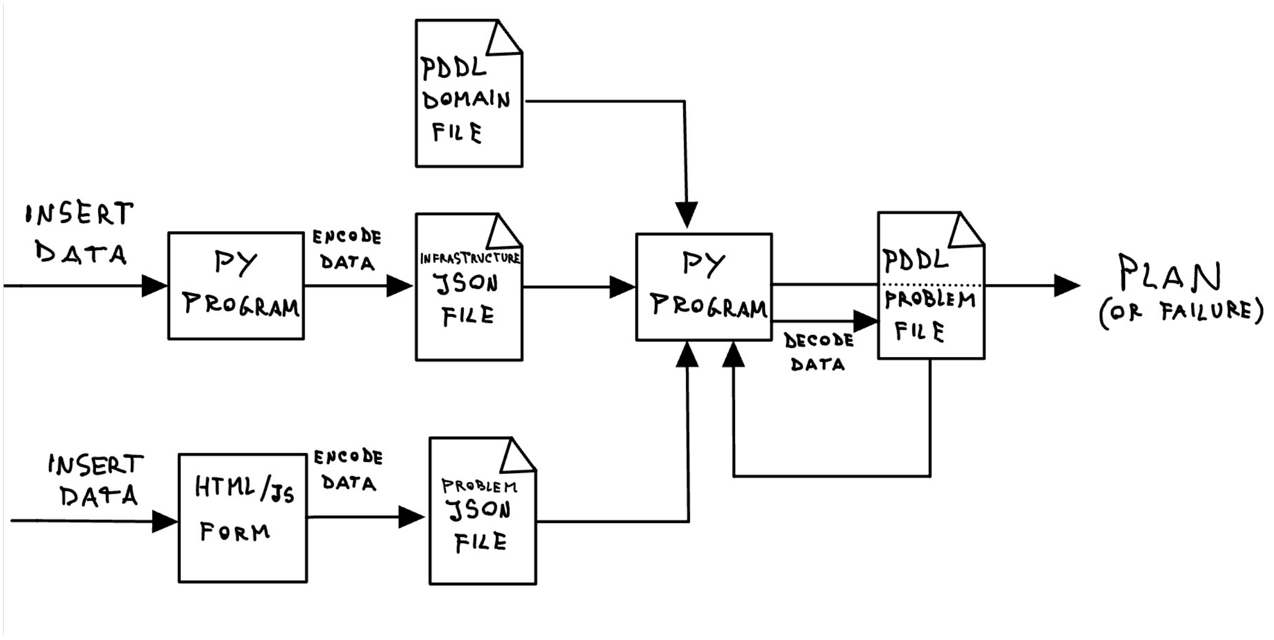


Fig. 1

2. Data representation

First important thing to do is to represent data in an appropriate way and decode it through a program to let the right representation in a PDDL syntax. I chose to represent it trough a JSON file, where for each key correspond a list of relative data; the insertion of data is divided in two parts, one for the

infrastructure of station and railways and one for the problem where we can set the trains with their itinerary.
To generate the infrastructure part we have to lunch a python program from shell, this program will take the input and will encode data into a JSON file, in particular data inserted are:

- **STATIONS:** *NameStation1,NumOfRailways-…*
  —> List of available stations and its railways: Name of station, a comma, the number of railways of this station, the data are separated from "-"

- **PTH_BTW_STATIONS:** *NameStation_NumOfRailway-Node1-Node2…-NodeN-NameStation_NumOfRailway; …*
  —> Represent the different lines between stations: the station is represented with its name and the number of the railway of those line (separated form "_"), next the list of the nodes on those line separated form "-" until the final station, each line is separated from ";"

This operation is not needed to execute the program, in fact there are 3 default files representing 3 different infrastructures (related to [Fig. a-b-c]) that can be used.
Oss.: If you define a new infrastructure, note that you are inserting only the names of the stations and of the nodes, the program will detect a rail for each track included between two nodes and assign it a name in an automatic and progressive way (R_1, R_2, …. R_n). In this way it is simplified the definition and the planning of a problem without the long procedure to assign a name to each rail by the user.
To generate a problem with trains and their itinerary, there is an HTML form to fill to generate a JSON file with following data:

- **TRAINS:** *NameTrain1-NameTrain2-…*
  —> List of available trains: Name of trains separated from a "-"

- **TRAINS_IN_STATION_INIT:** *Train1,Station_Railway - …*
  —> List of Initial position of trains starting from a station: Name of train separated from "," with the name of station and the railway of the station from which it start, data separated form "-"

- **TRAINS_OUT_STATION_INIT:** *Train1,Node1,Node2-…*
  —> List of Initial position of trains starting from a railway: Name of train separated from "," with the name of nodes in which the piece of railway is included, data separated form "-"

- **TRAINS_IN_STATION_GOAL:** *Train1,Station - …*
  —> List of final position of trains arriving in a station: Name of train separated from “,” with the name of station, data separated form “-”

- **TRAINS_OUT_STATION_GOAL:** *Train1,Node1,Node2-…*
  —> List of final position of trains arriving a railway: Name of train separated from “,” with the name of nodes in which the piece of railway is included, data separated form “-”

It is important inserting data referring to the infrastructure you chose or you created; after the insertion, the web page will download the JSON file (which you have to move in the project directory). In the original directory it is present a default file containing the problem instance for the infrastructure_1, which is presented as example n.1 in this report.

3. Domain and Problem Representation

To find a plan, the problem is represented in PDDL (Planning Domain Definition Language), that is a standard encoding language for “classical” planning tasks. It allow to represent *objects*: things in the world that interest us; *predicates*: properties of objects that we are interested in; can be true or false; the *initial state*: the state of the world that we start in; the g*oal specification*: things that we want to be true; *actions/operators*: ways of changing the state of the world. All these features are represented in two files: a domain and a problem file, which are passes to a plan solver which will find a plan (a set of action) if it exists. In the domain file will be defined the predicates and the actions needed for the problem, while in the problem file will be defined the objects, the initial state and the goal of an instance of the problem.
For this specific problem, I pass to the program a “train-domain.pddl” file with the follow proprieties:
- The requirements “strips”, which mean to allows the usage of basic add and delete effects (set the predicates true and false) as specified in STRIPS, a more complicated planning language.

- The follow predicates:
  - *train*: to define a train
  - *station*: to define a station
  - *railway*: to define a railway
  - *isin*: to define that a railway is part of a station
  - *conn*: to define that two railways are interconnected between them
  - *free*: to define that a railway is not busy and a train can cross it
  - *at*: to define where a train is
  - *instation*: to define if a train is stopped into a station. If it is true, the train is not ready to start and can’t move

- *passed*: to define if a train already crossed a railway (this to let trains to go only in the froward sense, *demonstration at pag.9)

- The follow actions:
  - *move*: to let a train to move from a railway to another one, it is the main action trough which let the train to travel
  - s*top*: to indicate that a train, after it is moved on an intern railway of a station, it is stopped and no more available to leave
  - *ready to start*: to indicate that a train that holds on an intern railway of a station, is authorized to leave and can be moved

The "problem.pddl" file is generated from a program that take as argument the infrastructure and the problem JSON files, which are decoded, generating a file containing: the object used (trains, stations and railways), the initial state and the goal state.

4. The planner

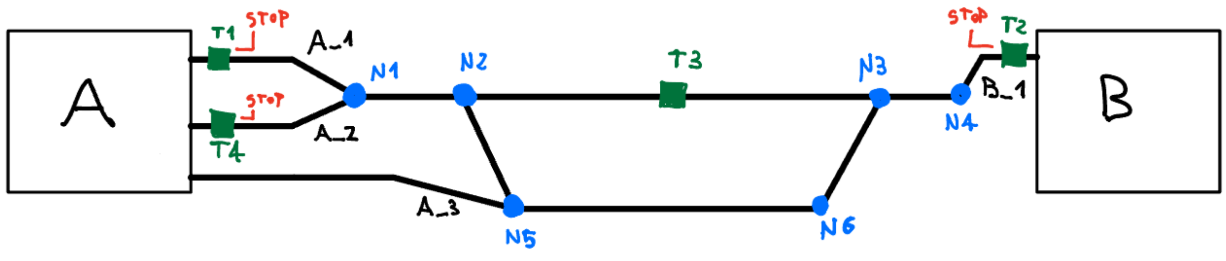
After inserting data and creating the problem file, the program take as the domain and the problem files and return a plan if it exist. This is done with a http request to 'http://solver.planning.domains/solve', an automated planner and validator in the cloud, where the file are send with a JSON form and the response is also in JSON form. From the latter, if a solution exist and no errors occurs, I output the details of the search and the plan, in particular: the kind of search, the cost and the time to find the plan, the nodes generated and expanded during the search; then there is the plan with all action and finally I decode this plan in a readable way, in particular substituting the name of the railways (that we do not know because assigned randomly from the program) with the pieced of the lines crossed by the trains, that is with the names of the nodes to cross to reach the goal.

5. Experiment

I done 3 experiments with 3 different infrastructures (illustrated at the appendix) to verify the functionality of the programs: I produced first an infrastructure inserting data and generating a JSON file, next I produced a problem inserting data of trains from the HTML form, finally I lunched the program to solve the problem and a plan was produced.

5.1 Experiment 1

In this experiment I created an instance of the problem with 2 stations (A with 3 internal railways and B with one internal railways), 4 trains (T1, T2, T3, T4) and some lines with some nodes between the station, all initialized as in the follow figure:



And the final goal with train T1 at station B, train T2 and T3 at station A and train T4 between nodes N5 and N6.
The produced JSON files are:

-for the infrastructure:

```
{
        "STATIONS": "A,3-B,1",
        "PTH_BTW_STATIONS": “A_1-N1-N2-N3-N4-B_1;A_2-N1-N2-N3-N4-B_1;
                              A_3-N5-N6-N3-N4-B_1;A_2-N1-N2-N5-N6-N3-N4-B_1”,
}
```

-for the problem

```
{
        "TRAINS": "T1-T2-T3-T4",
        "TRAINS_IN_STATION_INIT": "T1,A_1-T2,B_1-T4,A_2",
        "TRAINS_OUT_STATION_INIT": "T3,N2,N3",
        "TRAINS_IN_STATION_GOAL": "T1,B-T2,A-T3,A",
        "TRAINS_OUT_STATION_GOAL": "T4,N5,N6"
}
```

After lunching the program to find a plan, this is the result:
- the report on the search

```
Details on search:
BFS search completed
Plan found with cost: 19
Nodes expanded during search: 104
Nodes generated during search: 434
Total time: 0.14
```

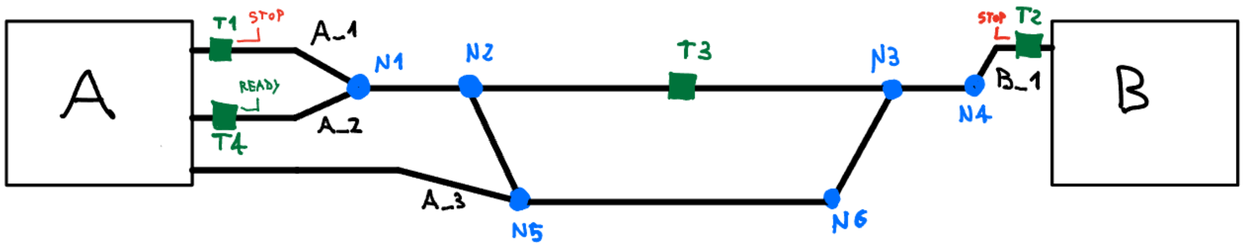
- the plan found:

```
(ready_to_start t4 a a_2)
(move t4 a_2 r_1)
(move t4 r_1 r_6)
(move t3 r_2 r_1)
```

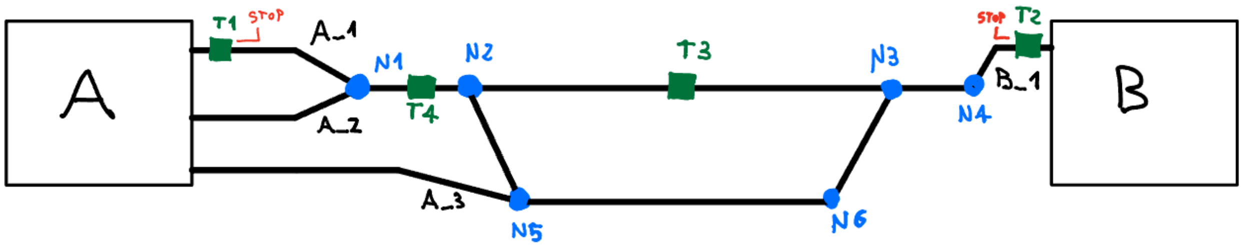
```
(ready_to_start t2 b b_1)
(move t3 r_1 a_2)
(stop t3 a_2 a)
(ready_to_start t1 a a_1)
(move t2 b_1 r_3)
(move t2 r_3 r_5)
(move t1 a_1 r_1)
(move t1 r_1 r_2)
(move t1 r_2 r_3)
(move t2 r_5 r_4)
(move t2 r_4 a_3)
(stop t2 a_3 a)
(move t4 r_6 r_4)
(move t1 r_3 b_1)
(stop t1 b_1 b)
```

In detail the plan, explained and illustrated from the following images is:

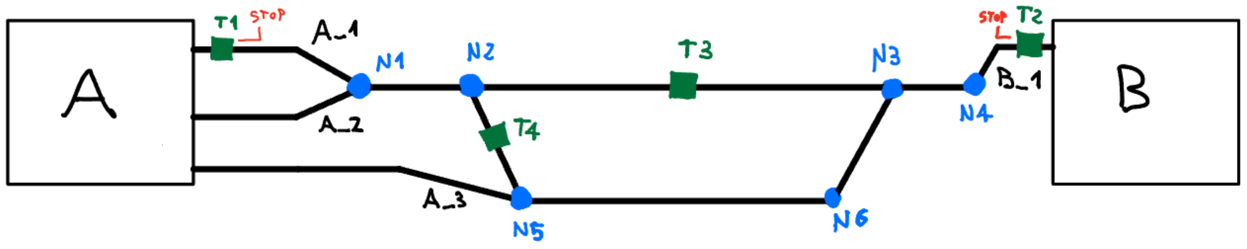
```
ACTION N.1:
Train T4 available on railway A_2 of station A
```



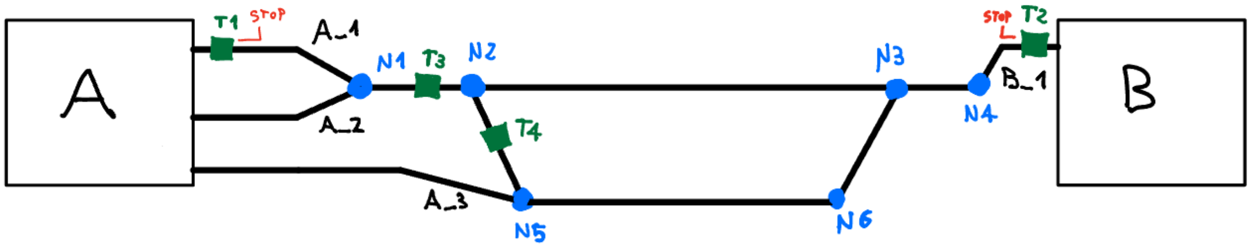
```
ACTION N.2:
Move train T4 on rail A_2-N1-N2
```



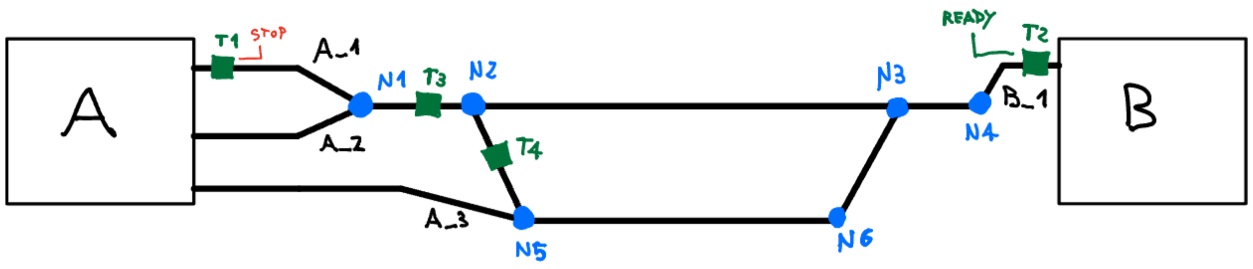
ACTION N.3:
Move train T4 on rail N1-N2-N5



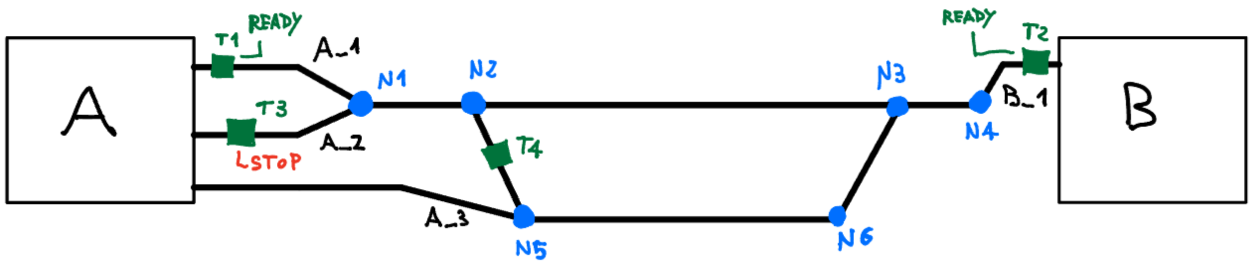
ACTION N.4:
Move train T3 on rail N3-N2-N1



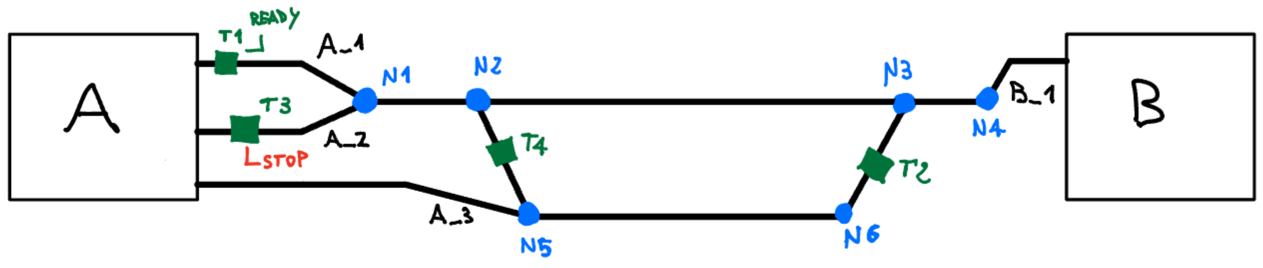
ACTION N.5:
Train T2 available on railway B_1 of station B



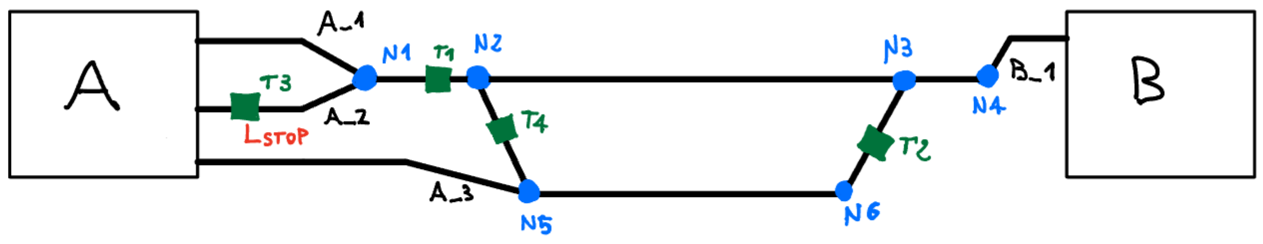
ACTION N.6:
Move train T2 on rail N2-N1-A_2
ACTION N.7:
Train T3 stopped at railway A of station A_2
ACTION N.8:
Train T1 available on railway A_1 of station A



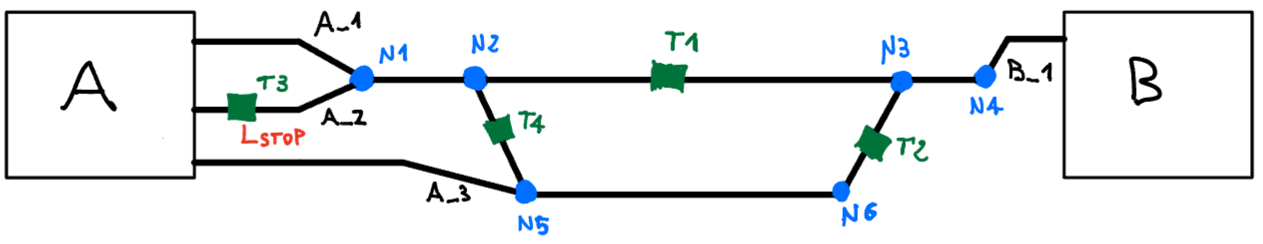
ACTION N.9:
Move train T2 on rail B_1-N4-N3



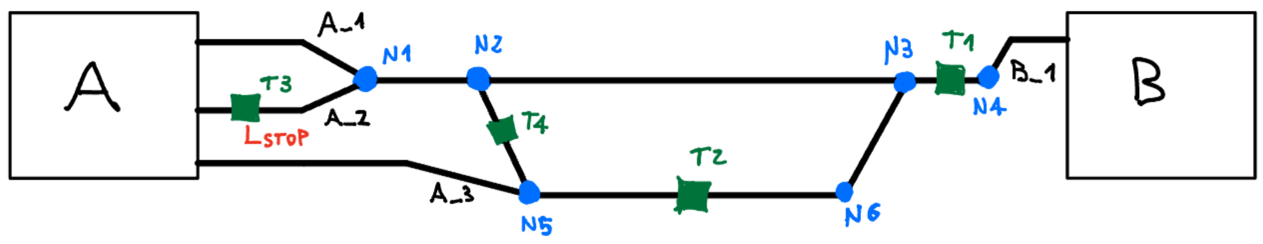
ACTION N.10:
Move train T2 on rail N4-N3-N6



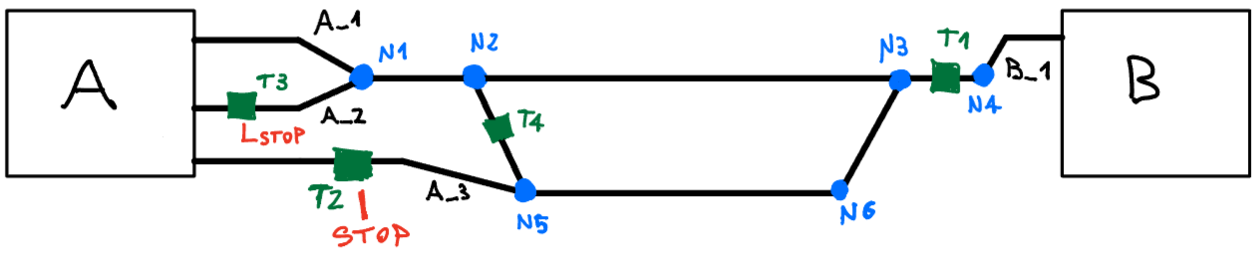
ACTION N.11:
Move train T1 on rail A_1-N1-N2



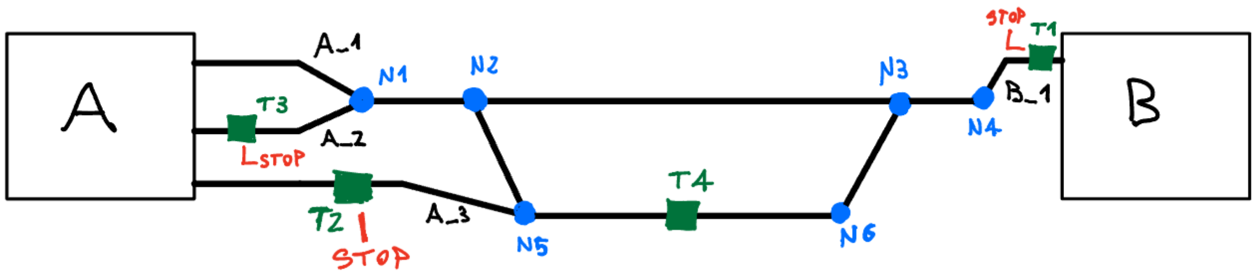
ACTION N.12:
Move train T1 on rail N1-N2-N3
ACTION N.13:



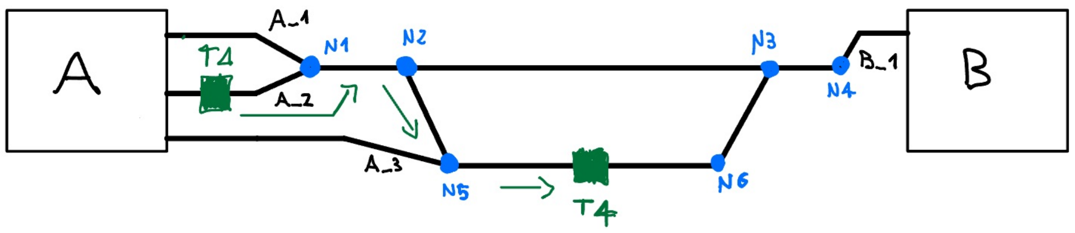
Move train T1 on rail N2-N3-N4
ACTION N.14:
Move train T2 on rail N3-N6-N5
ACTION N.15:
Move train T2 on rail N6-N5-A_3
ACTION N.16:
Train T2 stopped at railway A of station A_3



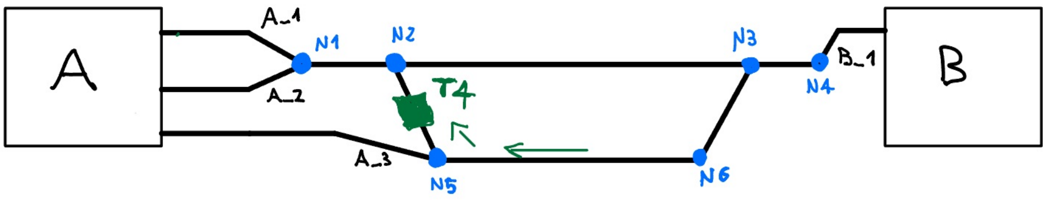
ACTION N.17:
Move train T4 on rail N2-N5-N6
ACTION N.18:
Move train T1 on rail N3-N4-B_1
ACTION N.19: (GOAL)
Train T1 stopped at railway B of station B_1



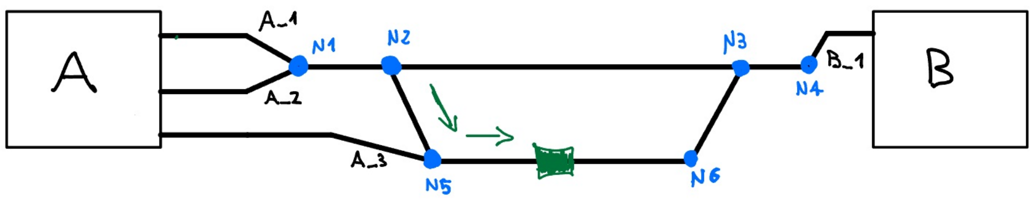
***Demonstration of importance of the predicate "passed"**
Executing the plan with a domain where the predicate "passed" is eliminated, using the previous one example, we obtain a plan long 23 actions (longer than the previous one) where the T4 train turn back with respect to its natural forward direction.
The following images shows the artifact:
- Starting in the A_2 rail, with the ACTION N.9 the T4 train arrive between nodes N5 and N6:



- In ACTION N.19 the plan move it back between nodes N5 and N2:

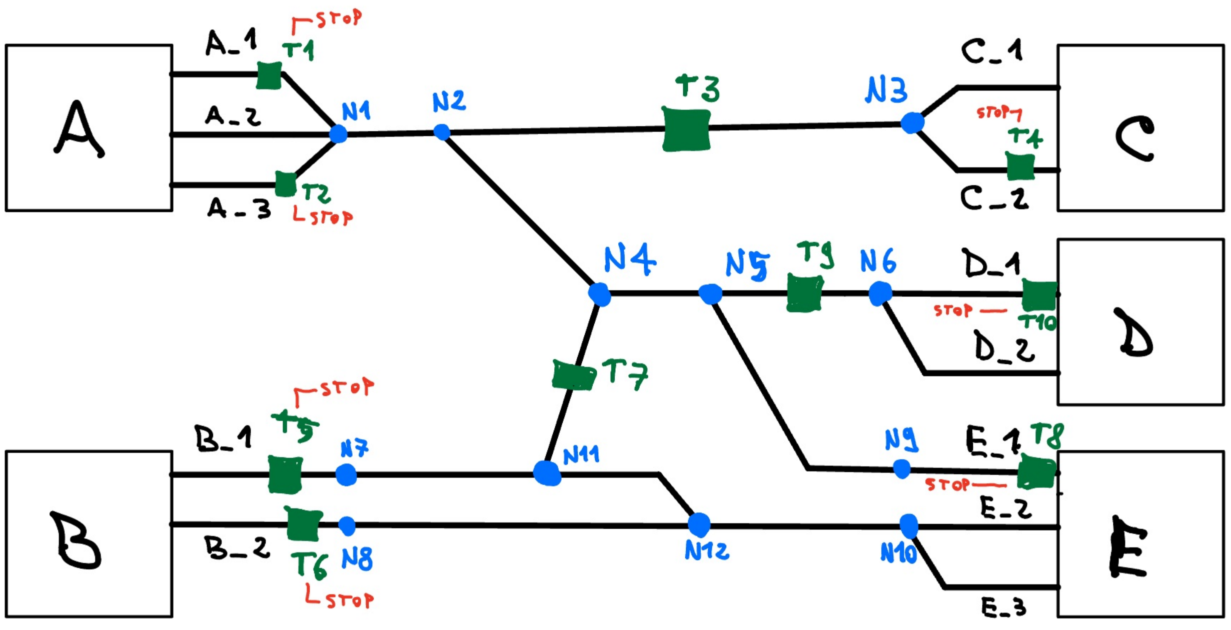


- In ACTION N.22 the plan move it again between nodes N5 and N6:



5.2 Experiment 2

In this other example I created an instance of the problem with 5 stations (A with 3 internal railways and B with 2 internal railways, C with 2 internal railways, D with 2 internal railways, E with 3 internal railways), 10 trains (T1, T2, T3, T4, T5, T6, T7, T8, T9, T10) and some lines with some nodes between the station, with the following initial state:



The goal for this problem is to have:
T1 in D, T2 in E, T3 in B, T4 in A, T5 between N2 and N3, T6 in C, T7 in A, T8 between N7 and N11, T9 in E, T10 in C.

The produced JSON files are:

-for the infrastructure:

```
{
        "STATIONS": “A,3-B,2-C,2-D,2-E,3”,
        "PTH_BTW_STATIONS": “ A_1-N1-N2-N3-C_1;
                              A_1-N1-N2-N3-C_2;
                              A_1-N1-N2-N4-N5-N6-D_1;
                              A_1-N1-N2-N4-N5-N6-D_2;
                              A_1-N1-N2-N4-N5-N9-E_1;
                              A_1-N1-N2-N4-N11-N7-B_1;
                              A_1-N1-N2-N4-N11-N12-N8-B_2;
                              A_1-N1-N2-N4-N11-N12-N10-E_2;
                              A_1-N1-N2-N4-N11-N12-N10-E_3;
                              A_2-N1-N2-N3-C_1;
                              A_2-N1-N2-N3-C_2;
                              A_2-N1-N2-N4-N5-N6-D_1;
                              A_2-N1-N2-N4-N5-N6-D_2;
                              A_2-N1-N2-N4-N5-N9-E_1;
                              A_2-N1-N2-N4-N11-N7-B_1;
                              A_2-N1-N2-N4-N11-N12-N8-B_2;
                              A_2-N1-N2-N4-N11-N12-N10-E_2;
                              A_2-N1-N2-N4-N11-N12-N10-E_3;
                              A_3-N1-N2-N3-C_1;
                              A_3-N1-N2-N3-C_2;
                              A_3-N1-N2-N4-N5-N6-D_1;
                              A_3-N1-N2-N4-N5-N6-D_2;
                              A_3-N1-N2-N4-N5-N9-E_1;
                              A_3-N1-N2-N4-N11-N7-B_1;
                              A_3-N1-N2-N4-N11-N12-N8-B_2;
                              A_3-N1-N2-N4-N11-N12-N10-E_2;
                              A_3-N1-N2-N4-N11-N12-N10-E_3;
                              B_1-N7-N11-N4-N2-N3-C_1;
                              B_1-N7-N11-N4-N2-N3-C_2;
                              B_1-N7-N11-N4-N5-N6-D_1;
                              B_1-N7-N11-N4-N5-N6-D_2;
                              B_1-N7-N11-N4-N5-N9-E_1;
                              B_1-N7-N11-N12-N10-E_2;
                              B_1-N7-N11-N12-N10-E_3;
                              B_2-N8-N12-N10-E_2;
                              B_2-N8-N12-N10-E_3;
                              B_2-N8-N12-N11-N4-N5-N9-E_1;
                              B_2-N8-N12-N11-N4-N5-N6-D_1;
                              B_2-N8-N12-N11-N4-N5-N6-D_2;
                              B_2-N8-N12-N11-N4-N2-N3-C_1;
                              B_2-N8-N12-N11-N4-N2-N3-C_2;
```

```
                                        C_1-N3-N2-N4-N5-N6-D_1;
                                        C_1-N3-N2-N4-N5-N6-D_2;
                                        C_1-N3-N2-N4-N5-N9-E_1;
                                        C_1-N3-N2-N4-N11-N12-N10-E_2;
                                        C_1-N3-N2-N4-N11-N12-N10-E_3;
                                        C_2-N3-N2-N4-N5-N6-D_1;
                                        C_2-N3-N2-N4-N5-N6-D_2;
                                        C_2-N3-N2-N4-N5-N9-E_1;
                                        C_2-N3-N2-N4-N11-N12-N10-E_2;
                                        C_2-N3-N2-N4-N11-N12-N10-E_3;
                                        D_1-N6-N5-N9-E_1;
                                        D_1-N6-N5-N4-N11-N12-N10-E_2;
                                        D_1-N6-N5-N4-N11-N12-N10-E_3;
                                        D_2-N6-N5-N9-E_1;
                                        D_2-N6-N5-N4-N11-N12-N10-E_2;
                                        D_2-N6-N5-N4-N11-N12-N10-E_3"
}
```

-for the problem

```
{

“TRAINS":"T1-T2-T3-T4-T5-T6-T7-T8-T9-T10",
“TRAINS_IN_STATION_INIT":"T1,A_1-T2,A_3-T4,C_2-T5,B_1-T6,B_2-T8,E_1-T10,D_1",
“TRAINS_IN_STATION_GOAL":"T1,D-T2,E-T3,B-T4,A-T6,C-T7,A-T9,E-T10,C",
“TRAINS_OUT_STATION_INIT":"T3,N2,N3-T7,N11,N4-T9,N5,N6",
“TRAINS_OUT_STATION_GOAL”:"T5,N2,N3-T8,N7,N11"

}
```

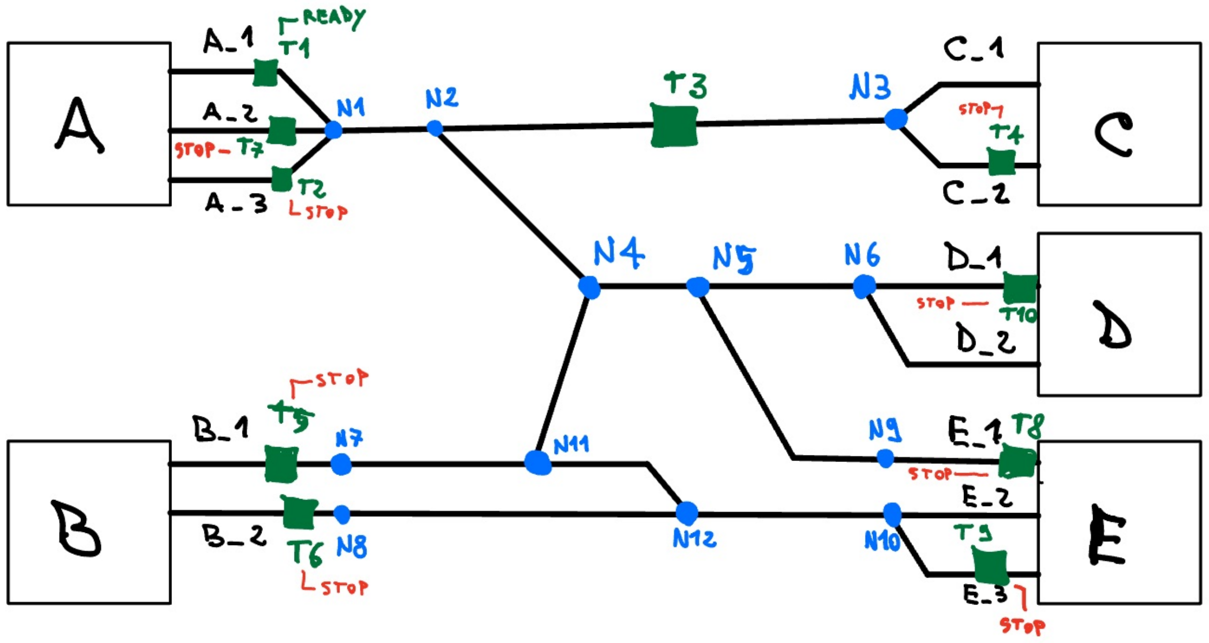
```
Details on search:
IW search completed
Nodes expanded during search: 4985
Nodes generated during search: 7766
Total time: 2.436
#RP_fluents 0Plan found with cost: 62

Plan:
(move t7 r_7 r_3)
(move t7 r_3 r_1)
(move t7 r_1 a_2)
(stop t7 a_2 a)
(move t9 r_5 r_4)
(move t9 r_4 r_7)
(move t9 r_7 r_9)
(move t9 r_9 r_11)
(move t9 r_11 e_3)
```

```
(stop t9 e_3 e)
(ready_to_start t1 a a_1)
(move t1 a_1 r_1)
(move t1 r_1 r_3)
(move t1 r_3 r_4)
(move t1 r_4 r_5)
(move t1 r_5 d_2)
(stop t1 d_2 d)
(move t3 r_2 r_1)
(move t3 r_1 r_3)
(ready_to_start t4 c c_2)
(move t4 c_2 r_2)
(move t4 r_2 r_1)
(move t4 r_1 a_1)
(stop t4 a_1 a)
(move t3 r_3 r_7)
(ready_to_start t5 b b_1)
(move t5 b_1 r_8)
(move t5 r_8 r_9)
(move t3 r_7 r_8)
(move t3 r_8 b_1)
(stop t3 b_1 b)
(ready_to_start t8 e e_1)
(move t8 e_1 r_6)
(move t8 r_6 r_4)
(move t8 r_4 r_7)
(move t8 r_7 r_8)
(ready_to_start t10 d d_1)
(move t10 d_1 r_5)
(move t10 r_5 r_4)
(move t10 r_4 r_3)
(move t10 r_3 r_2)
(move t10 r_2 c_2)
(stop t10 c_2 c)
(ready_to_start t2 a a_3)
(move t2 a_3 r_1)
(move t2 r_1 r_3)
(move t2 r_3 r_4)
```

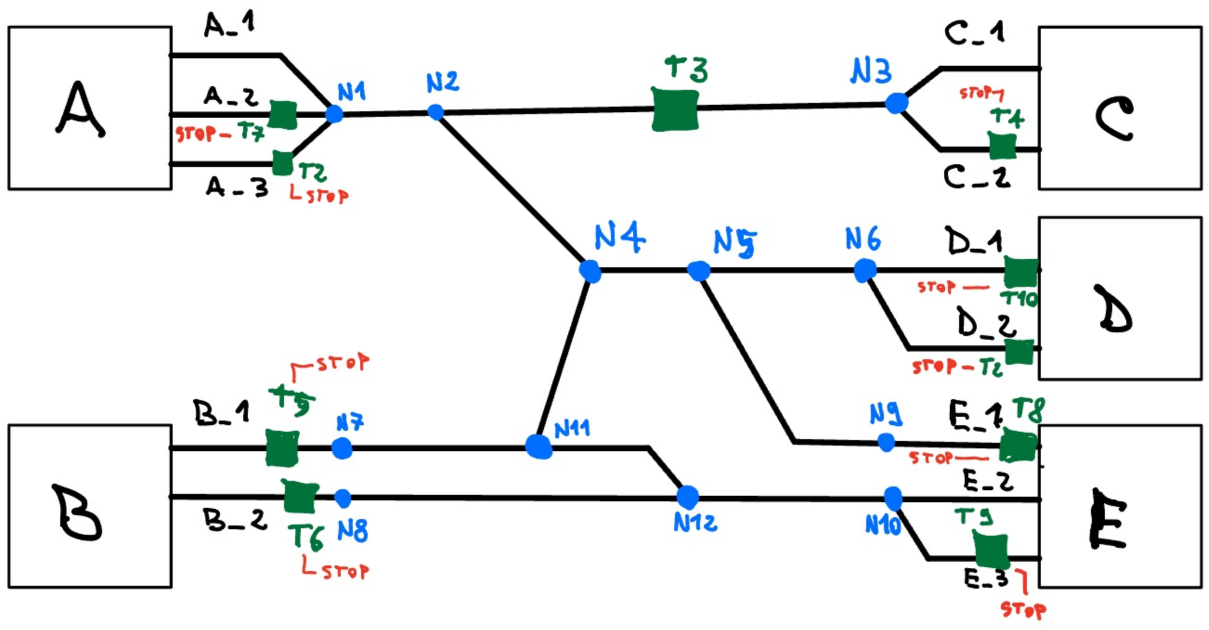
```
(move t2 r_4 r_6)
(move t2 r_6 e_1)
(stop t2 e_1 e)
(move t5 r_9 r_7)
(move t5 r_7 r_4)
(ready_to_start t6 b b_2)
(move t6 b_2 r_10)
(move t6 r_10 r_9)
(move t6 r_9 r_7)
(move t6 r_7 r_3)
(move t6 r_3 r_2)
(move t6 r_2 c_1)
(stop t6 c_1 c)
(move t5 r_4 r_3)
(move t5 r_3 r_2)
```

In detail the plan, explained and illustrated from the following images is:

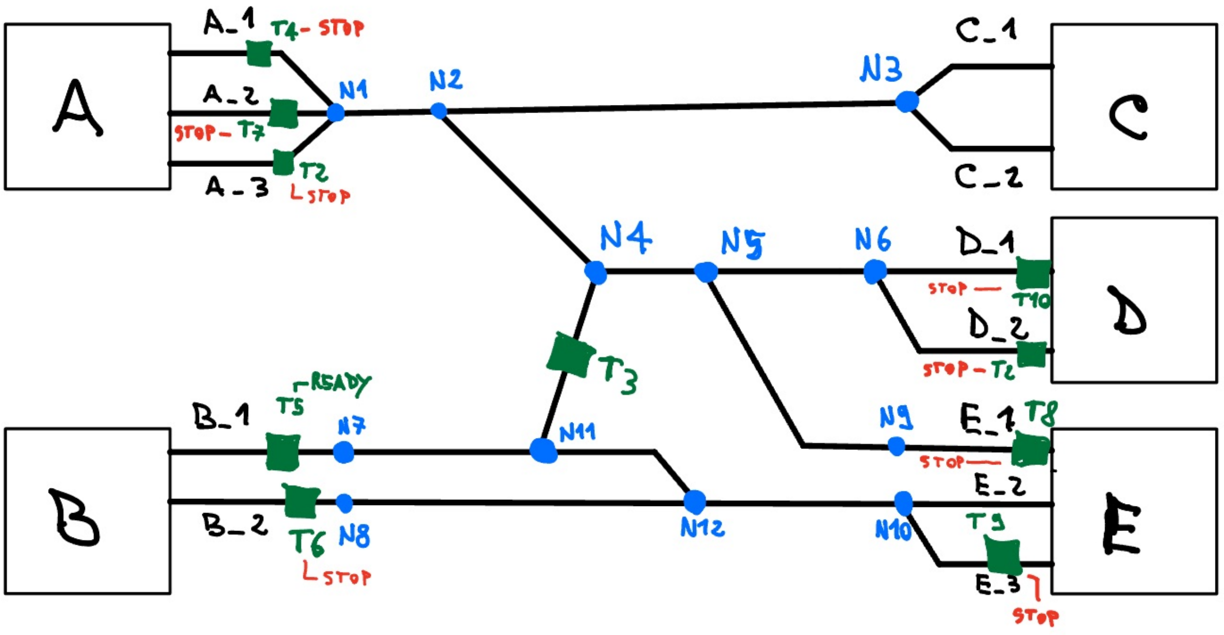
ACTION N.1:
Move train T7 on rail N11-N4-N2
ACTION N.2:
Move train T7 on rail N4-N2-N1
ACTION N.3:
Move train T7 on rail N2-N1-A_2
ACTION N.4:
Train T7 stopped at railway A_2 of station A
ACTION N.5:
Move train T9 on rail N6-N5-N4
ACTION N.6:
Move train T9 on rail N5-N4-N11
ACTION N.7:
Move train T9 on rail N4-N11-N12
ACTION N.8:
Move train T9 on rail N11-N12-N10
ACTION N.9:
Move train T9 on rail N12-N10-E_3
ACTION N.10:
Train T9 stopped at railway E_3 of station E
ACTION N.11:
Train T1 avaiable on railway A_1 of station A



ACTION N.12:
Move train T1 on rail A_1-N1-N2
ACTION N.13:
Move train T1 on rail N1-N2-N4
ACTION N.14:
Move train T1 on rail N2-N4-N5
ACTION N.15:
Move train T1 on rail N4-N5-N6
ACTION N.16:
Move train T1 on rail N5-N6-D_2
ACTION N.17:
Train T1 stopped at railway D_2 of station D

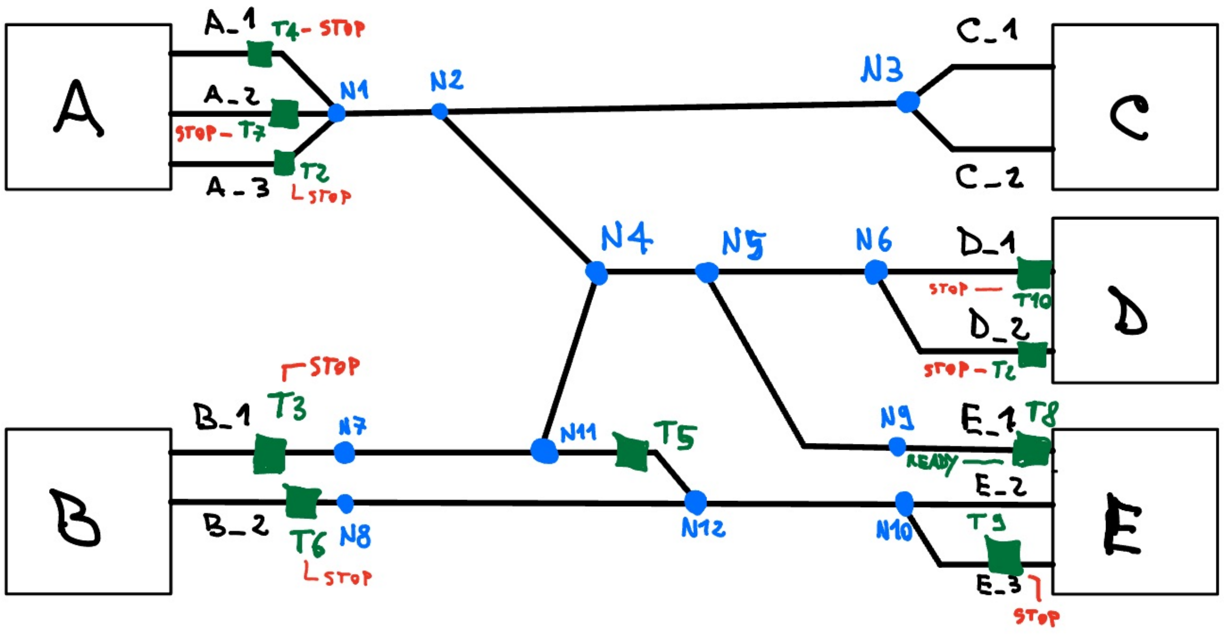


ACTION N.18:
Move train T3 on rail N3-N2-N1
ACTION N.19:
Move train T3 on rail N1-N2-N4
ACTION N.20:
Train T4 available on railway C_2 of station C
ACTION N.21:
Move train T4 on rail C_2-N3-N2
ACTION N.22:
Move train T4 on rail N3-N2-N1
ACTION N.23:
Move train T4 on rail N2-N1-A_1
ACTION N.24:
Train T4 stopped at railway A_1 of station A
ACTION N.25:
Move train T3 on rail N2-N4-N11
ACTION N.26:
Train T5 available on railway B_1 of station B

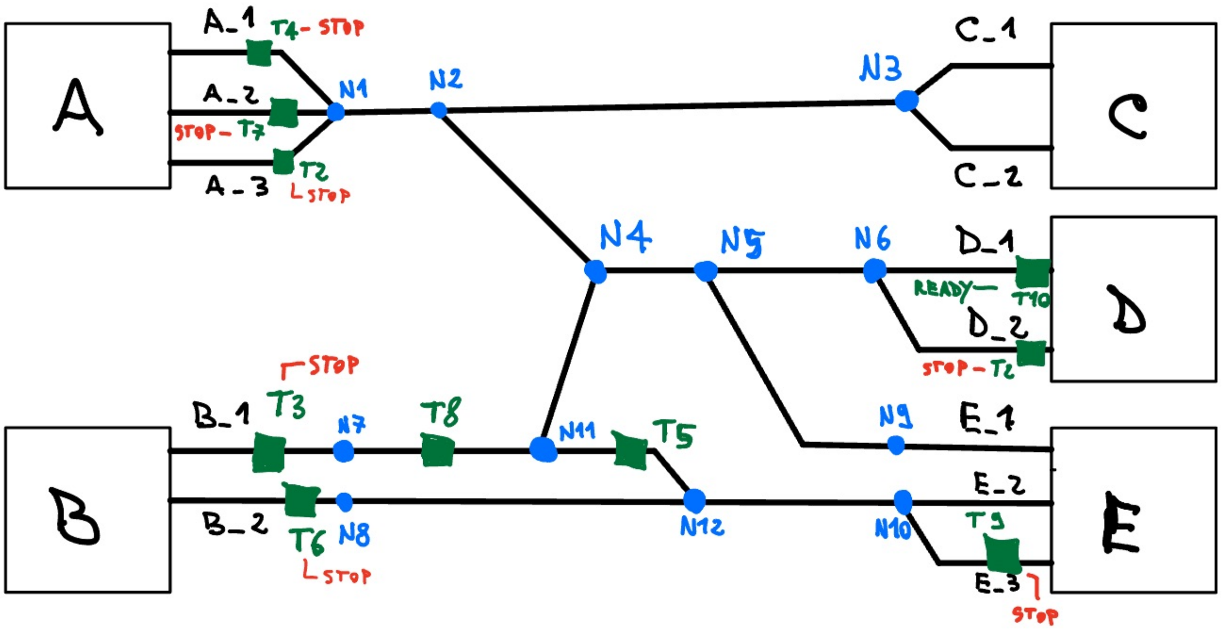


ACTION N.27:
Move train T5 on rail B_1-N7-N11
ACTION N.28:
Move train T5 on rail N7-N11-N12
ACTION N.29:
Move train T3 on rail N4-N11-N7
ACTION N.30:

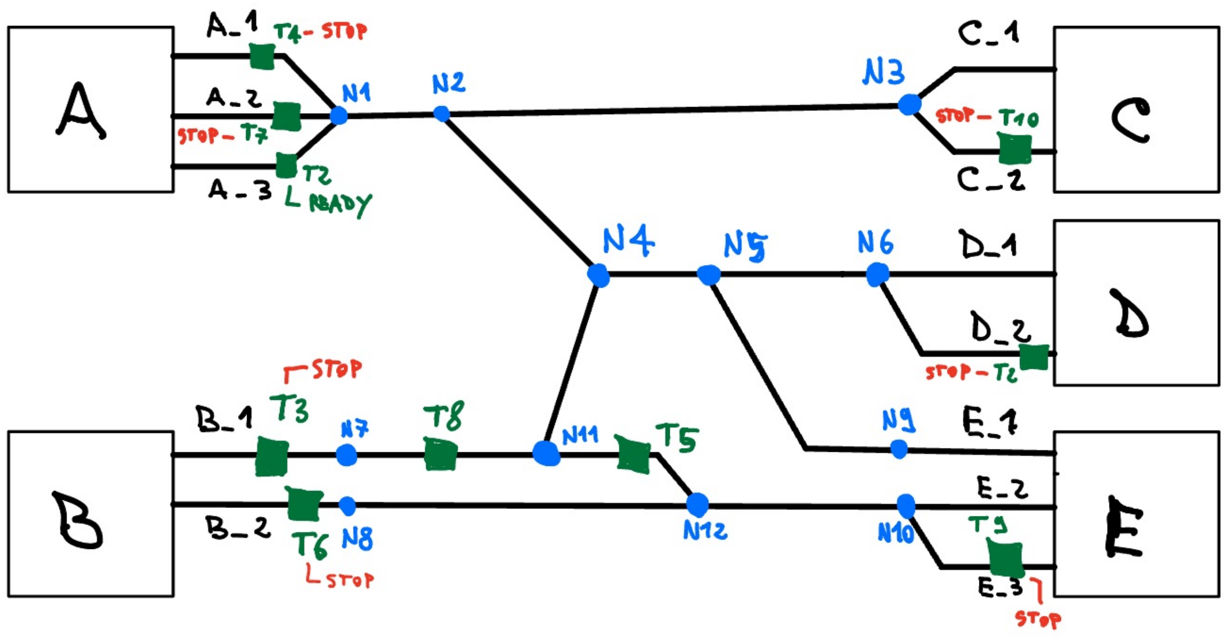
Move train T3 on rail N11-N7-B_1
ACTION N.31:
Train T3 stopped at railway B_1 of station B
ACTION N.32:
Train T8 available on railway E_1 of station E



ACTION N.33:
Move train T8 on rail E_1-N9-N5
ACTION N.34:
Move train T8 on rail N9-N5-N4
ACTION N.35:
Move train T8 on rail N5-N4-N11
ACTION N.36:
Move train T8 on rail N4-N11-N7
ACTION N.37:
Train T10 available on railway D_1 of station D

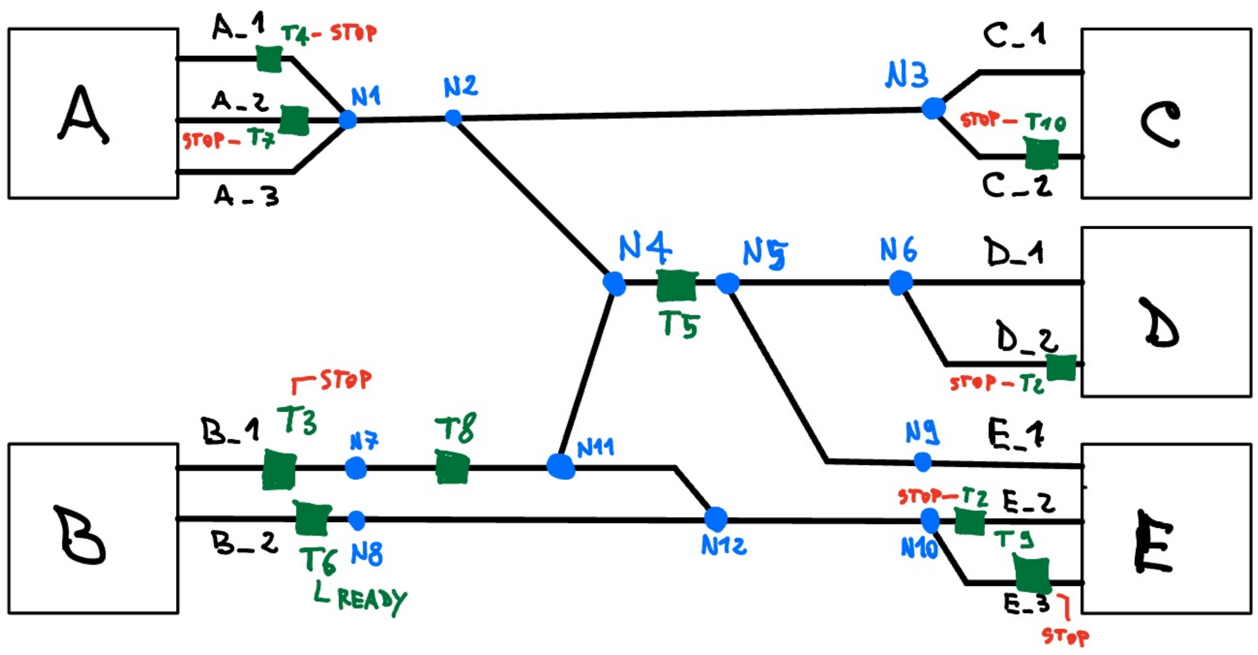


ACTION N.38:
Move train T10 on rail D_1-N6-N5
ACTION N.39:
Move train T10 on rail N6-N5-N4
ACTION N.40:
Move train T10 on rail N5-N4-N2
ACTION N.41:
Move train T10 on rail N4-N2-N3
ACTION N.42:
Move train T10 on rail N2-N3-C_2
ACTION N.43:
Train T10 stopped at railway C_2 of station C
ACTION N.44:
Train T2 available on railway A_3 of station A

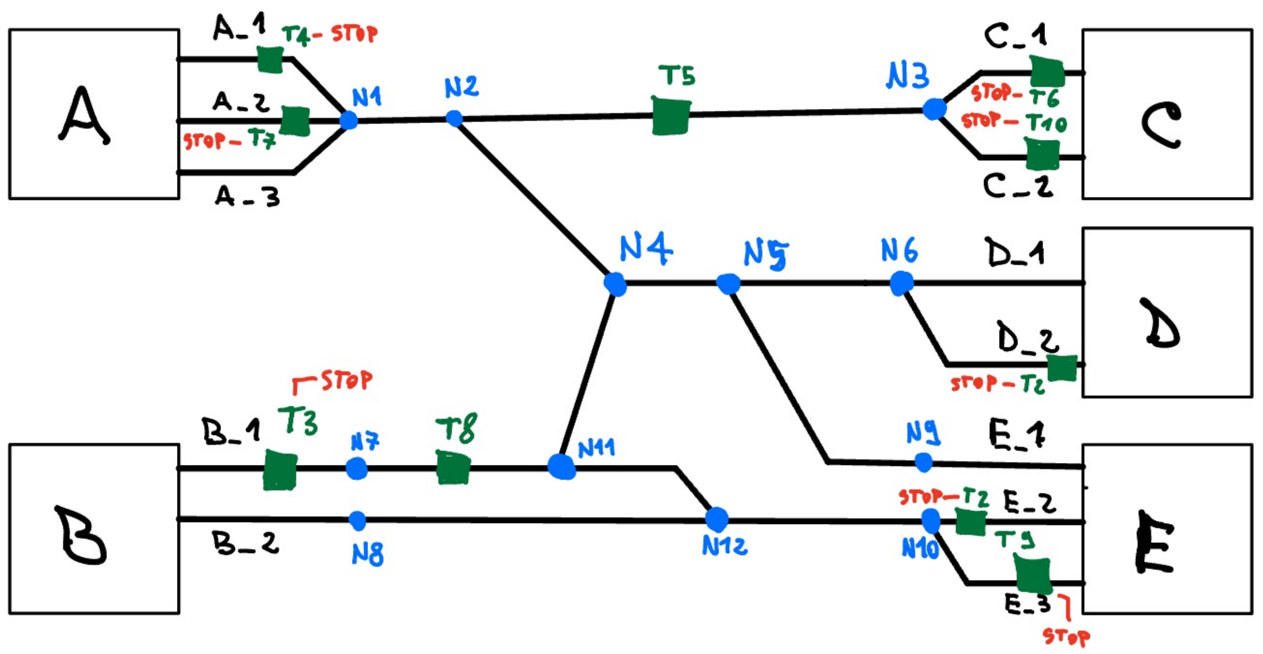


ACTION N.45:
Move train T2 on rail A_3-N1-N2
ACTION N.46:
Move train T2 on rail N1-N2-N4
ACTION N.47:
Move train T2 on rail N2-N4-N5
ACTION N.48:
Move train T2 on rail N4-N5-N9
ACTION N.49:
Move train T2 on rail N5-N9-E_1

ACTION N.50:
Train T2 stopped at railway E_1 of station E
ACTION N.51:
Move train T5 on rail N12-N11-N4
ACTION N.52:
Move train T5 on rail N11-N4-N5
ACTION N.53:
Train T6 available on railway B_2 of station B

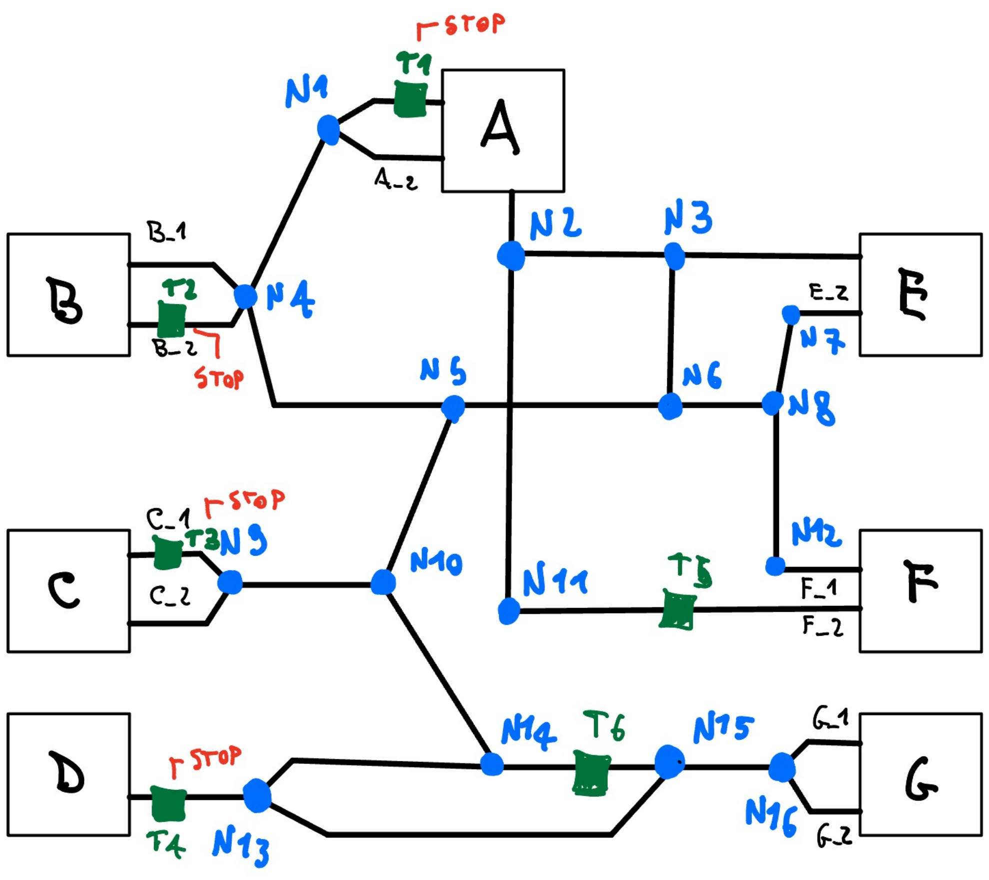


ACTION N.54:
Move train T6 on rail B_2-N8-N12
ACTION N.55:
Move train T6 on rail N8-N12-N11
ACTION N.56:
Move train T6 on rail N12-N11-N4
ACTION N.57:
Move train T6 on rail N11-N4-N2
ACTION N.58:
Move train T6 on rail N4-N2-N3
ACTION N.59:
Move train T6 on rail N2-N3-C_1
ACTION N.60:
Train T6 stopped at railway C_1 of station C
ACTION N.61:
Move train T5 on rail N5-N4-N2
ACTION N.62: (GOAL)
Move train T5 on rail N4-N2-N3



5.3 Experiment 3

In this other example I created an instance of the problem with 7 stations (A with 3 internal railways and B with 2 internal railways, C with 2 internal railways, D with 1 internal railway, E with 2 internal railways, F with 2 internal railways, G with 2 internal railways), 6 trains (T1, T2, T3, T4, T5, T6) and some lines with some nodes between the station. The init configuration is:



With the following goal state:
T1 in E, T2 in F, T3 between nodes N13 and N15, T4 in A, T5 in B, T6 in C.

The produced JSON files are:

-for the infrastructure:

```
{
        "STATIONS": "A,3-B,2-C,2-D,1-E,2-F,2-G,2",
        "PTH_BTW_STATIONS": “A_1-N1-N4-B_1;
                              A_1-N1-N4-B_2;
                              A_1-N1-N4-N5-N10-N9-C_1;
                              A_1-N1-N4-N5-N10-N9-C_2;
                              A_1-N1-N4-N5-N10-N14-N13-D_1;
                              A_1-N1-N4-N5-N10-N14-N15-N13-D_1;
                              A_1-N1-N4-N5-N10-N14-N15-N16-G_1;
                       A_1-N1-N4-N5-N10-N14-N15-N16-G_2;
                       A_1-N1-N4-N5-N6-N3-E_1;
                       A_1-N1-N4-N5-N6-N8-N7-E_2;
                       A_1-N1-N4-N5-N6-N8-N12-F_1;
                       A_1-N1-N4-N5-N6-N3-N2-N11-F_2;
                       A_2-N2;A_3-N2-N3-E_1;
                       A_3-N2-N3-N6-N8-N7-E_2;
                       A_3-N2-N3-N6-N8-N12-F_1;
                       A_3-N2-N3-N6-N5-N4-B_1;
                       A_3-N2-N3-N6-N5-N4-B_2;
                       A_3-N2-N3-N6-N5-N10-N9-C_1;
                       A_3-N2-N3-N6-N5-N10-N9-C_2;
                       A_3-N2-N3-N6-N5-N10-N14-N13-D_1;
                       A_3-N2-N3-N6-N5-N10-N14-N15-N13-D_1;
                       A_3-N2-N3-N6-N5-N10-N14-N15-N16-G_1;
                       A_3-N2-N3-N6-N5-N10-N14-N15-N16-G_2;
                       B_1-N4-N5-N6-N3-E_1;
                       B_1-N4-N5-N6-N3-N2-N11-F_2;
                       B_1-N4-N5-N6-N8-N7-E_2;
                       B_1-N4-N5-N6-N8-N12-F_1;
                       B_1-N4-N5-N10-N9-C_1,
                       B_1-N4-N5-N10-N9-C_2;
                       B_1-N4-N5-N10-N14-N13-D_1;
                       B_1-N4-N5-N10-N14-N15-N13-D_1;
                       B_1-N4-N5-N10-N14-N15-N16-G_1;
                       B_1-N4-N5-N10-N14-N15-N16-G_2;
                       B_2-N4;C_1-N9-N10-N5-N6-N8-N7-E_2;
                       C_1-N9-N10-N5-N6-N8-N12-F_1;
                       C_1-N9-N10-N5-N6-N3-N2-N11-F_2;
                       C_1-N9-N10-N14-N13-D_1;
                       C_1-N9-N10-N14-N15-N13-D_1;
                       C_1-N9-N10-N14-N15-N16-G_1;
                       C_1-N9-N10-N14-N15-N16-G_2;C_2-N9;
                       D_1-N13-N14-N10-N5-N6-N8-N7-E_2;
                       D_1-N13-N14-N10-N5-N6-N8-N12-F_1;
                       D_1-N13-N14-N10-N5-N6-N3-N2-N11-F_2;
```

D_1-N13-N15-N14-N10-N5-N6-N8-N7-E_2;
D_1-N13-N15-N14-N10-N5-N6-N8-N12-F_1;
D_1-N13-N15-N14-N10-N5-N6-N3-N2-N11-F_2;
D_1-N13-N14-N15-N16-G_1;
D_1-N13-N14-N15-N16-G_2;
D_1-N13-N15-N16-G_1;
D_1-N13-N15-N16-G_2;
E_1-N3-N2-N11-F_2;
E_1-N3-N6-N8-N12-F_1;
E_1-N3-N6-N5-N10-N14-N15-N16-G_1;
E_1-N3-N6-N5-N10-N14-N15-N16-G_2;
E_1-N3-N6-N5-N10-N14-N13-N15-N16-G_1;
E_1-N3-N6-N5-N10-N14-N13-N15-N16-G_2;
E_2-N7-N8-N12-F_1;
E_2-N7-N8-N6-N5-N10-N14-N15-N16-G_1;
E_2-N7-N8-N6-N5-N10-N14-N15-N16-G_2;
E_2-N7-N8-N6-N5-N10-N14-N13-N15-N16-G_1;
E_2-N7-N8-N6-N5-N10-N14-N13-N15-N16-G_2;
F_1-N12-N8-N6-N5-N10-N14-N15-N16-G_1;
F_1-N12-N8-N6-N5-N10-N14-N15-N16-G_2;
F_1-N12-N8-N6-N5-N10-N14-N13-N15-N16-G_1;
F_1-N12-N8-N6-N5-N10-N14-N13-N15-N16-G_2;
F_2-N11-N2-N3-N6-N5-N10-N14-N15-N16-G_1;
F_2-N11-N2-N3-N6-N5-N10-N14-N15-N16-G_2;
F_2-N11-N2-N3-N6-N5-N10-N14-N13-N15-N16-G_1;
F_2-N11-N2-N3-N6-N5-N10-N14-N13-N15-N16-G_2"

}

-for the problem

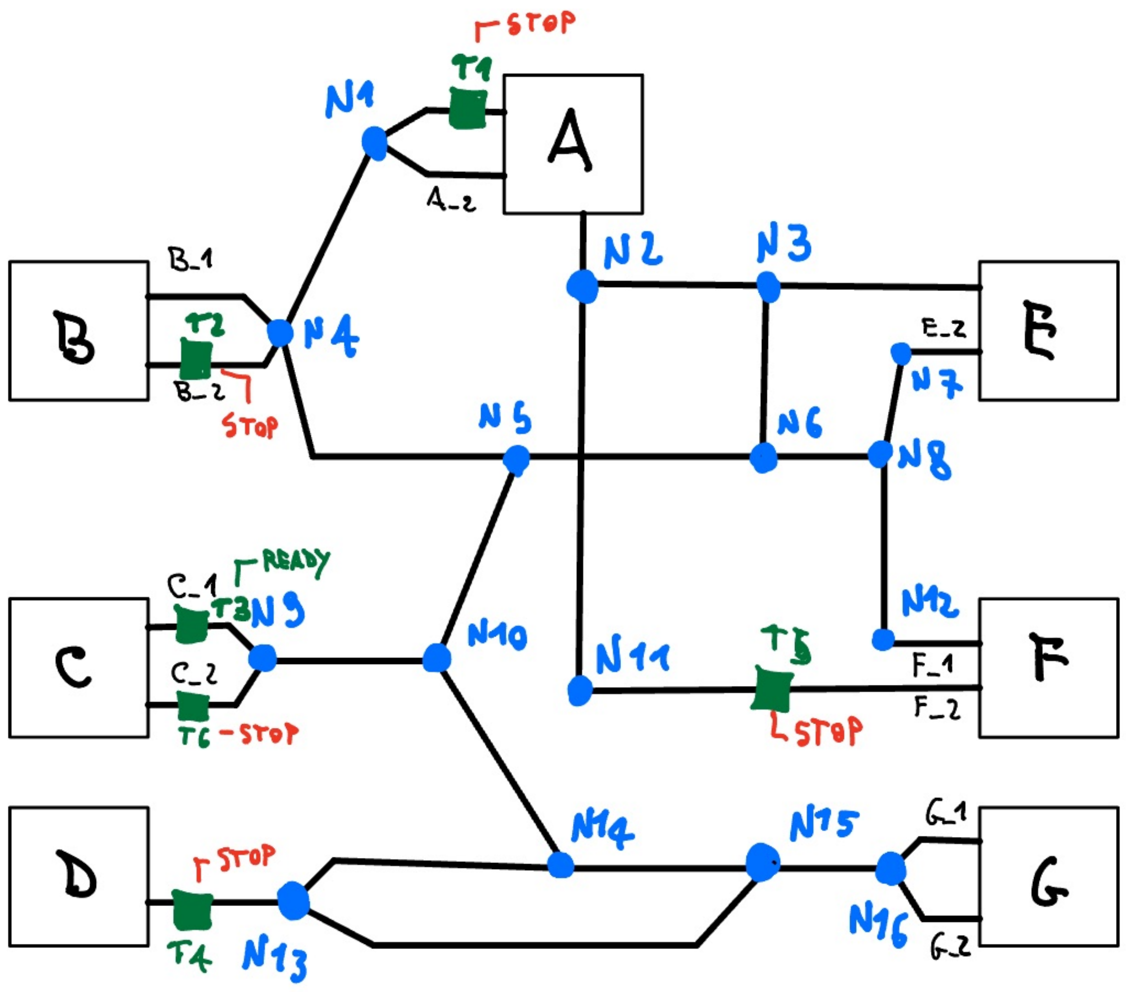
```
{
        "TRAINS":"T1-T2-T3-T4-T5-T6",
        "TRAINS_IN_STATION_INIT":"T1,A_1-T2,B_2-T3,C_1-T4,D_1-T5,F_2",
        "TRAINS_IN_STATION_GOAL":"T1,E-T2,F-T4,A-T5,B-T6,C",
        "TRAINS_OUT_STATION_INIT":"T6,N14,N15",
        "TRAINS_OUT_STATION_GOAL":"T3,N13,N15"
}
```

```
Details on search:
IW search completed
Nodes expanded during search: 2394
Nodes generated during search: 3858
Total time: 0.336
#RP_fluents 0Plan found with cost: 39

Plan:
(move t6 r_7 r_5)
(move t6 r_5 r_4)
```

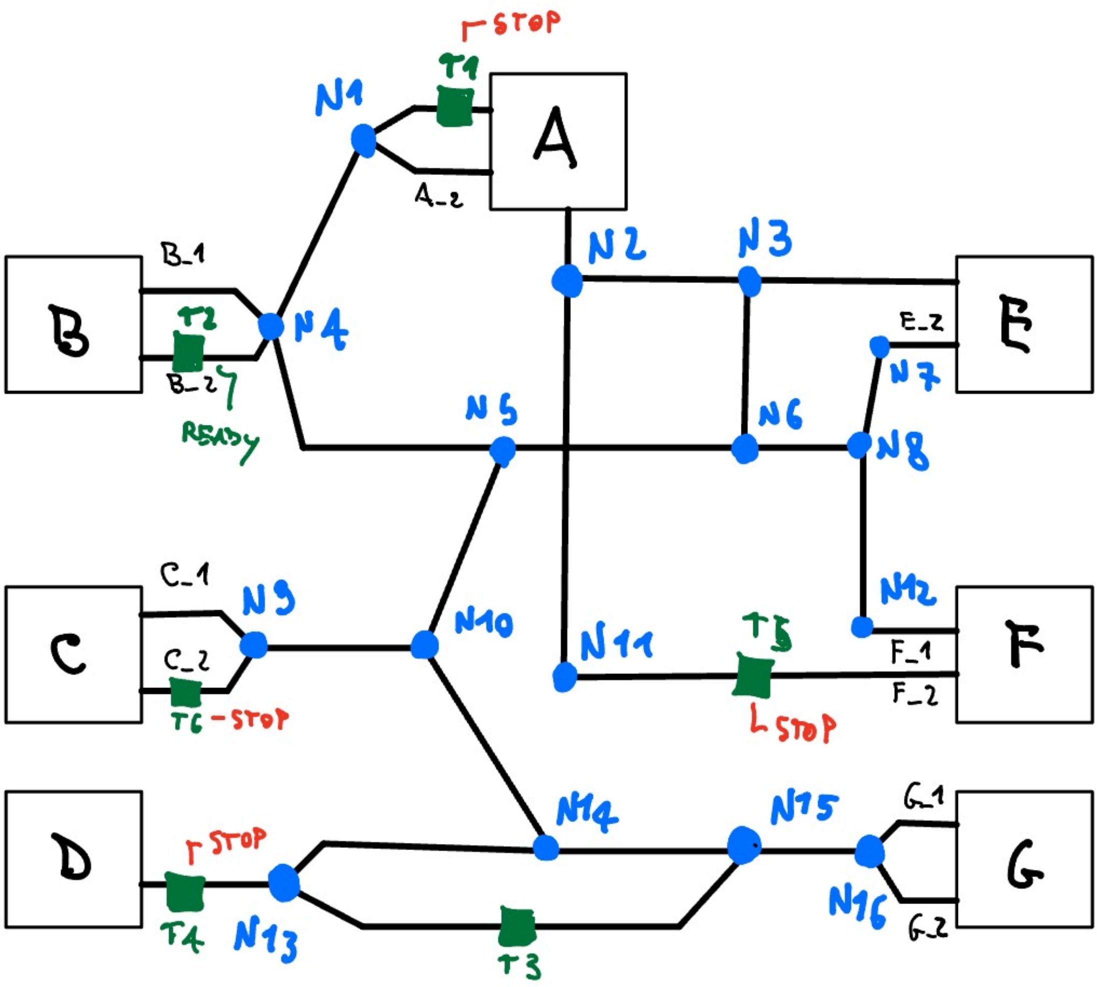
```
(move t6 r_4 c_2)
(stop t6 c_2 c)
(ready_to_start t3 c c_1)
(move t3 c_1 r_4)
(move t3 r_4 r_5)
(move t3 r_5 r_7)
(move t3 r_7 r_8)
(ready_to_start t2 b b_2)
(move t2 b_2 r_2)
(move t2 r_2 r_10)
(move t2 r_10 r_12)
(move t2 r_12 r_14)
(move t2 r_14 f_1)
(stop t2 f_1 f)
(ready_to_start t1 a a_1)
(move t1 a_1 r_1)
(move t1 r_1 r_2)
(move t1 r_2 r_10)
(move t1 r_10 r_11)
(move t1 r_11 e_1)
(stop t1 e_1 e)
(ready_to_start t5 f f_2)
(move t5 f_2 r_16)
(move t5 r_16 r_15)
(move t5 r_15 r_11)
(move t5 r_11 r_10)
(move t5 r_10 r_2)
(move t5 r_2 b_2)
(stop t5 b_2 b)
(ready_to_start t4 d d_1)
(move t4 d_1 r_6)
(move t4 r_6 r_5)
(move t4 r_5 r_3)
(move t4 r_3 r_2)
(move t4 r_2 r_1)
(move t4 r_1 a_1)
(stop t4 a_1 a)
```

In detail the plan, explained and illustrated from the following images is:

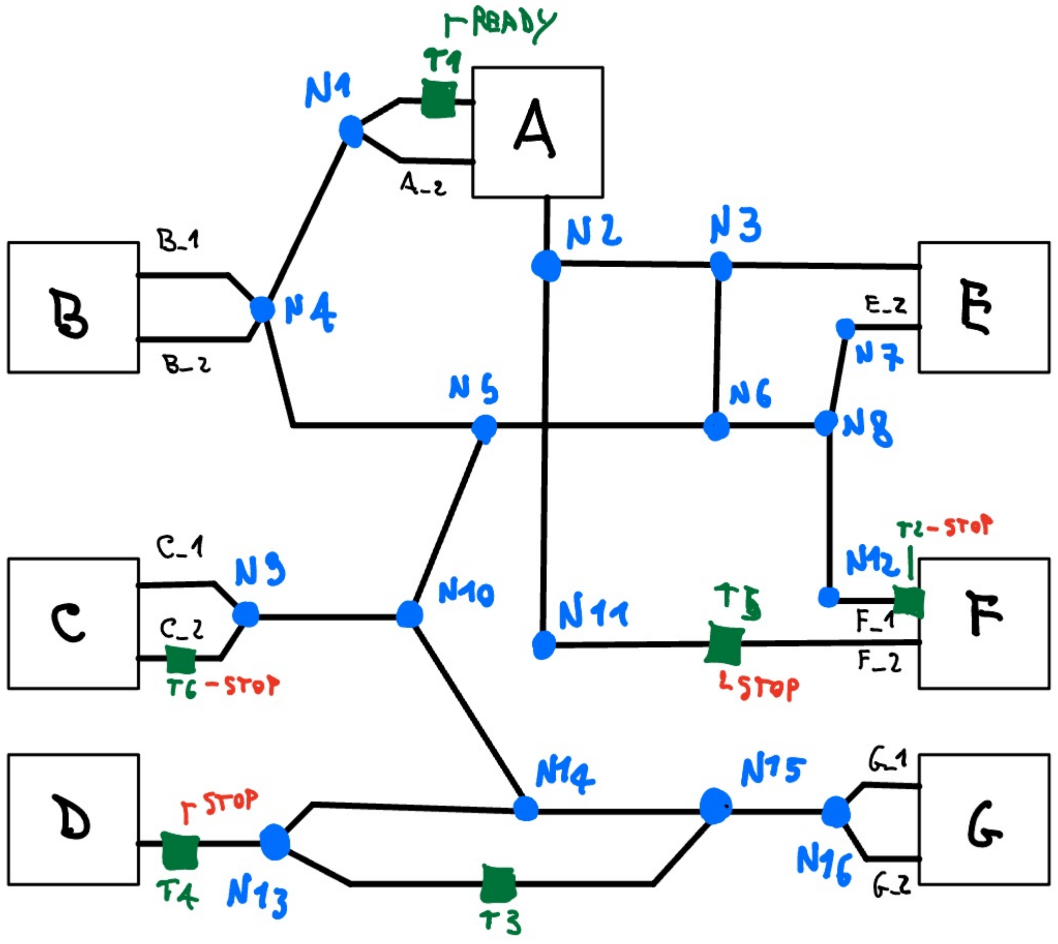
ACTION N.1:
Move train T6 on rail N15-N14-N10
ACTION N.2:
Move train T6 on rail N14-N10-N9
ACTION N.3:
Move train T6 on rail N10-N9-C_2
ACTION N.4:
Train T6 stopped at railway C_2 of station C
ACTION N.5:
Train T3 available on railway C_1 of station C



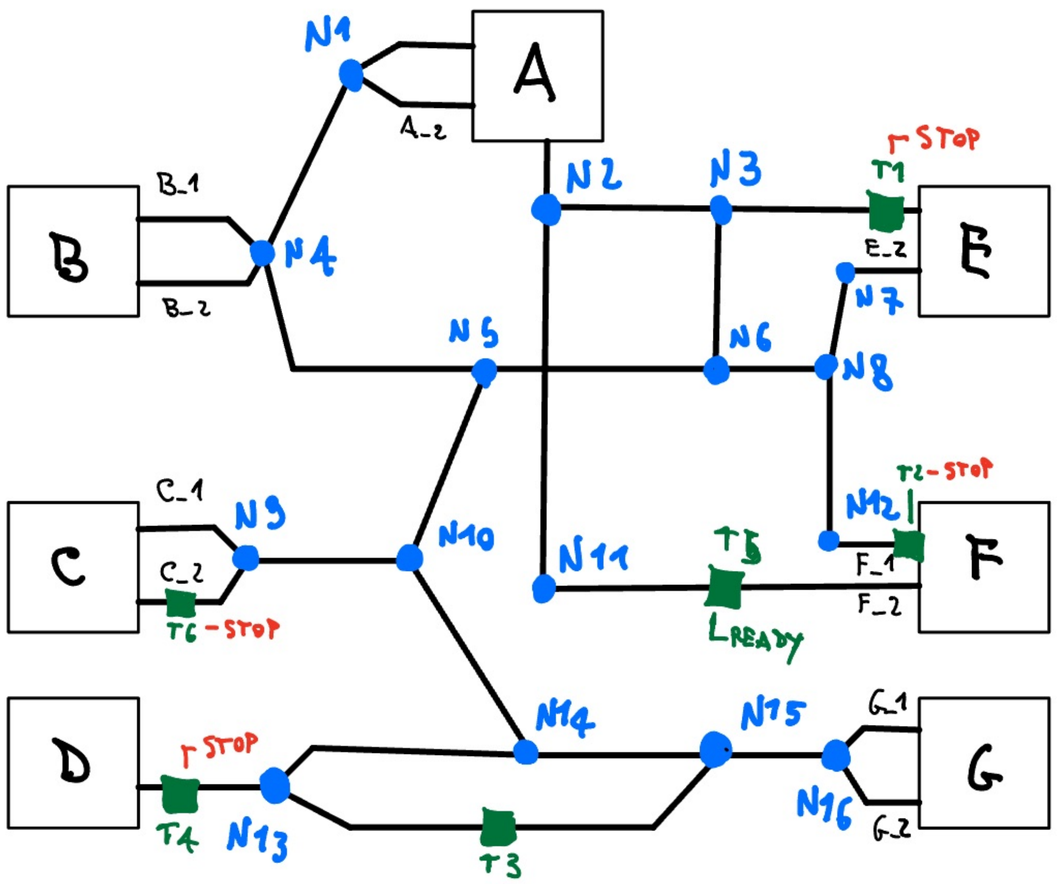
ACTION N.6:
Move train T3 on rail C_1-N9-N10
ACTION N.7:
Move train T3 on rail N9-N10-N14
ACTION N.8:
Move train T3 on rail N10-N14-N15
ACTION N.9:
Move train T3 on rail N14-N15-N13
ACTION N.10:
Train T2 available on railway B_2 of station B



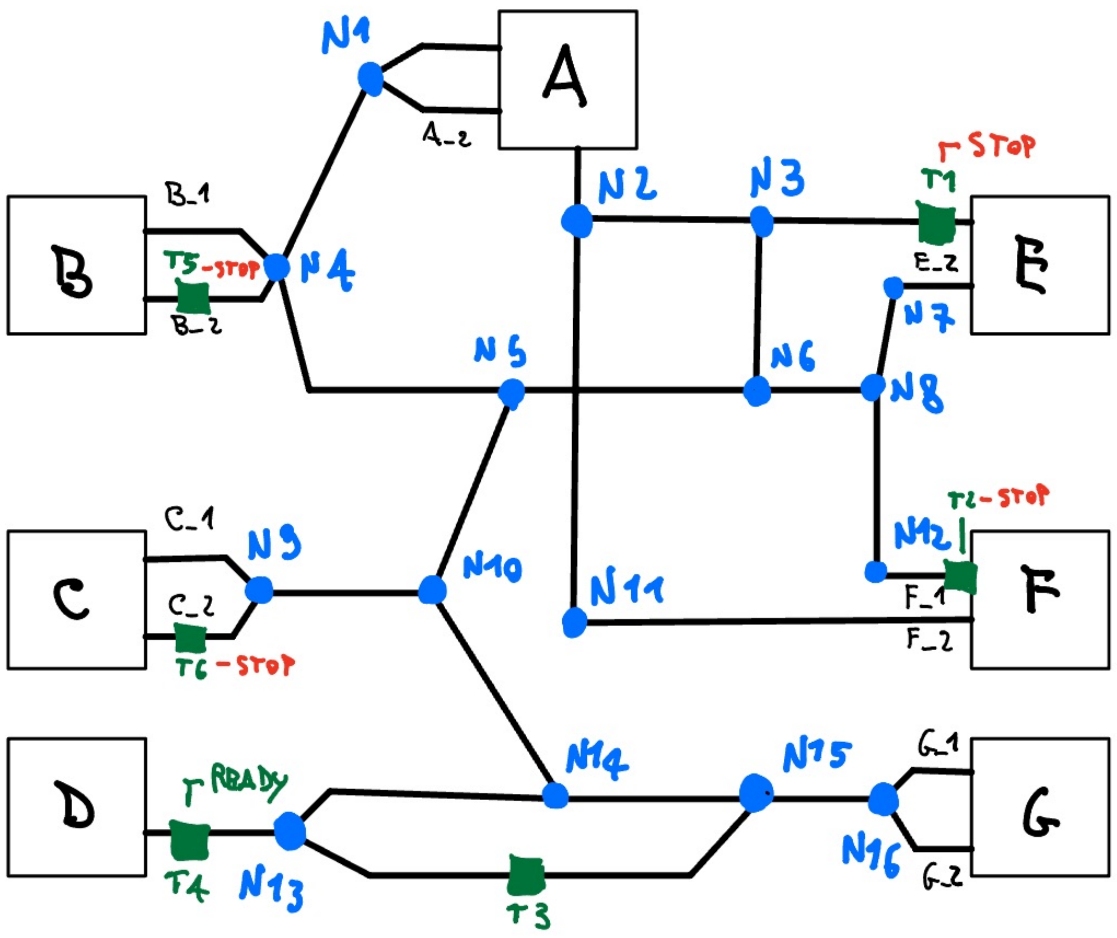
ACTION N.11:
Move train T2 on rail B_2-N4-N5
ACTION N.12:
Move train T2 on rail N4-N5-N6
ACTION N.13:
Move train T2 on rail N5-N6-N8
ACTION N.14:
Move train T2 on rail N6-N8-N12
ACTION N.15:
Move train T2 on rail N8-N12-F_1
ACTION N.16:
Train T2 stopped at railway F_1 of station F
ACTION N.17:
Train T1 available on railway A_1 of station A



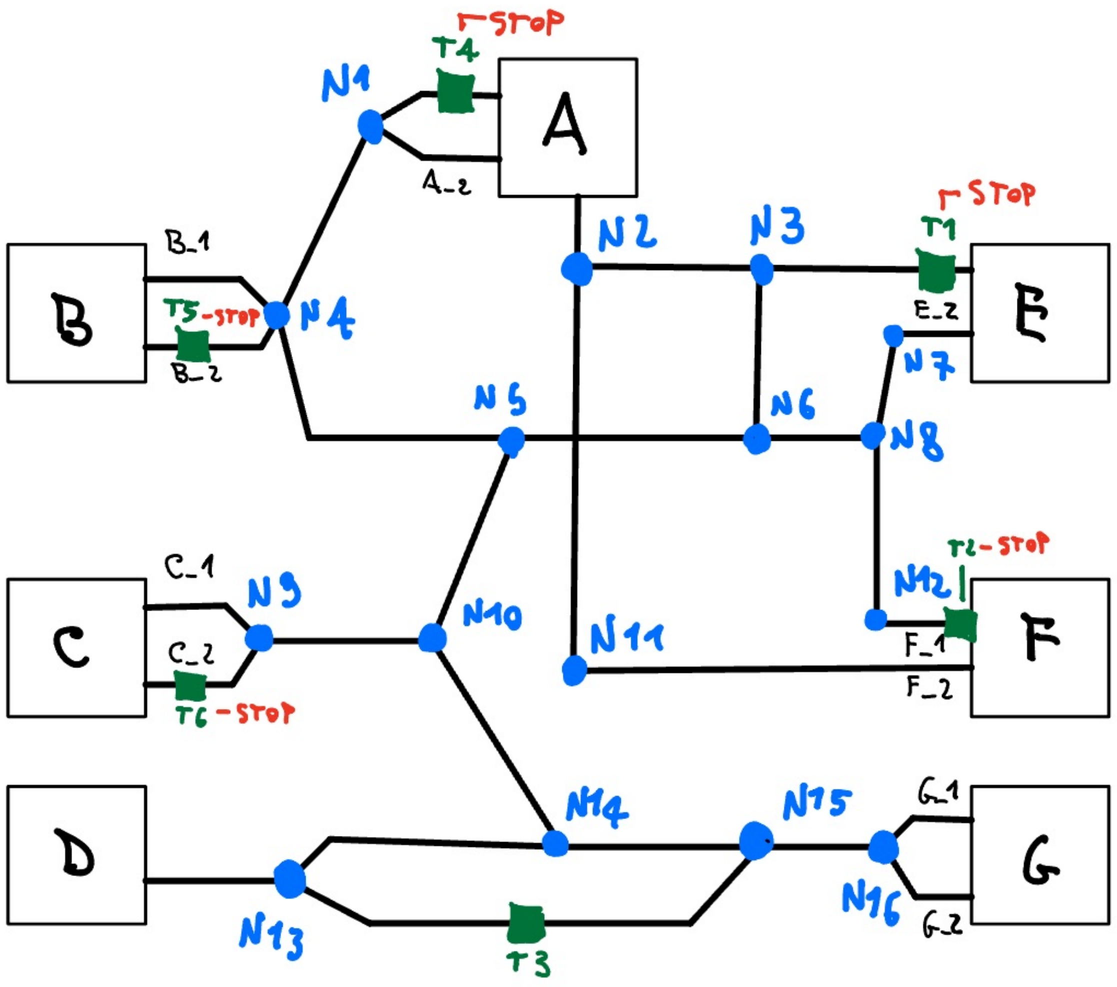
ACTION N.18:
Move train T1 on rail A_1-N1-N4
ACTION N.19:
Move train T1 on rail N1-N4-N5
ACTION N.20:
Move train T1 on rail N4-N5-N6
ACTION N.21:
Move train T1 on rail N5-N6-N3
ACTION N.22:
Move train T1 on rail N6-N3-E_1
ACTION N.23:
Train T1 stopped at railway E_1 of station E
ACTION N.24:
Train T5 available on railway F_2 of station F



ACTION N.25:
Move train T5 on rail F_2-N11-N2
ACTION N.26:
Move train T5 on rail N11-N2-N3
ACTION N.27:
Move train T5 on rail N2-N3-N6
ACTION N.28:
Move train T5 on rail N3-N6-N5
ACTION N.29:
Move train T5 on rail N6-N5-N4
ACTION N.30:
Move train T5 on rail N5-N4-B_2
ACTION N.31:
Train T5 stopped at railway B_2 of station B
ACTION N.32:
Train T4 available on railway D_1 of station D



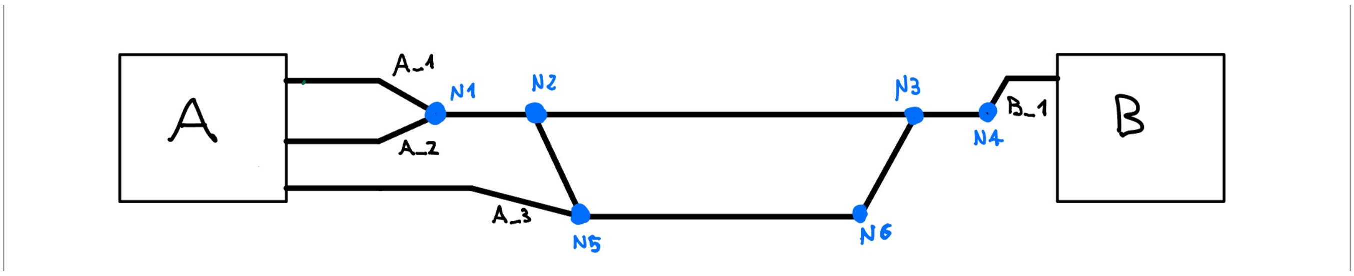
ACTION N.33:
Move train T4 on rail D_1-N13-N14
ACTION N.34:
Move train T4 on rail N13-N14-N10
ACTION N.35:
Move train T4 on rail N14-N10-N5
ACTION N.36:
Move train T4 on rail N10-N5-N4
ACTION N.37:
Move train T4 on rail N5-N4-N1
ACTION N.38:
Move train T4 on rail N4-N1-A_1
ACTION N.39: (GOAL)
Train T4 stopped at railway A_1 of station A



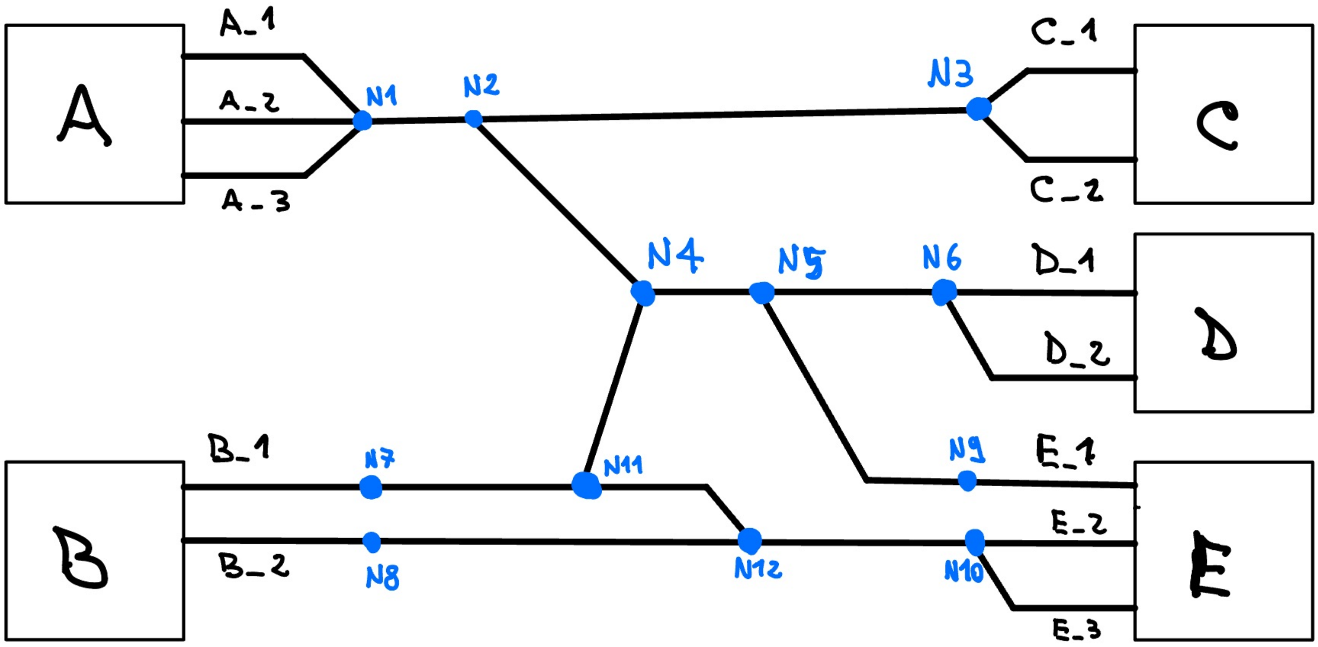
APPENDIX

In this appendix there are the figures of the three pre-configured infrastructures, needed if you want generate new problems.

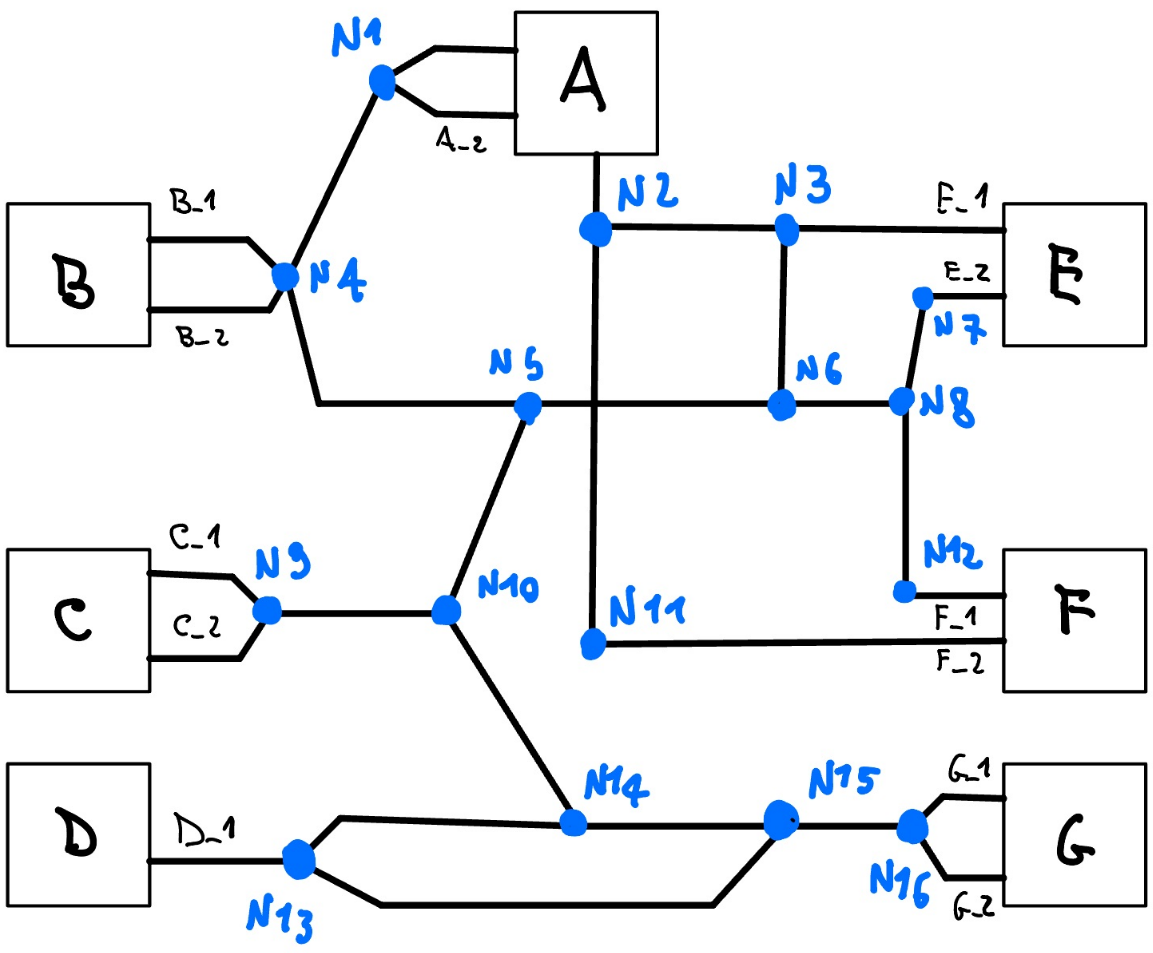
infrastructure 1



infrastructure 2



infrastructure_3


N1
A
A_2
N2
N3
E_1
B_1
B
N4
B_2
E_2
E
N7
N5
N6
N8
C_1
N9
N10
N12
C
C_2
N11
F_1
F
F_2
N14
N15
G_1
D
D_1
G
N13
N16
G_2