# Multi-Agent Frequency Allocation Project
# USER MANUAL

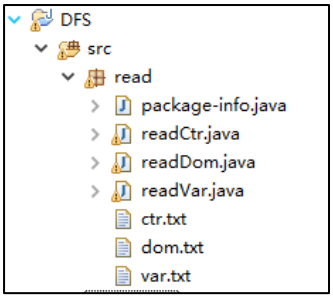
*ZunzunWANG&YishuoLYU*

Our software that solves the FAPP problem is divided into two parts. The first part of the project is achieved by Java and the second is achieved by Jacamo.
**DFS_ZunzunWANG_YishuoLYU.zip** is the source code java. (Data treatment)
**projet_jacamo_FAPP_ZunzunWANG_YishuoLYU.zip** is the source code jacamo**.** (Multi-agent)

## *Part1 preprocessing the data.*

First, we can read the file .txt and quickly collect the data needed (antennas in the same domain and the restrictions), and then we will input the data that we get in the first step into the other Java project for getting the DFS tree (the relationship of parent, pseudo-parent, children and pseudo-children).
Second, we will input the DFS tree and the restrictions of every two antennas into the Jacamo project. After the end of the project execution, we can get the final result that satisfier the problem FAPP.

Detailed steps are as follows:
Note: We will use the data of domain 5 as the example, and there are 5 antennas in this domain (141, 142, 251, 252, 293 and 294).
(1) We should put the original data file in the folder of java project (DFS/src/read).




Note: We can modify the path of every data file in the java project (DFS/src/read/readCtr.java, DFS/src/read/readDom.java, DFS/src/read/readVar.java). For example:

```
21          Scanner in = new Scanner(new File("H:/mobiledisk/eclipse/projet/DFS/DFS/src/read/var.txt"));
```

(2) After executing the java files mentioned above, you can get the data in the file.txt.
For example:

```
1   1
2   1
3   2
4   2
5   1
6   1
7   1
8   1
9   1
10  1
11  1
12  1
13  1
14  1
15  1
16  1
17  1
18  1
19  2
20  2
21  3
22  3
23  3
24  3
25  1
26  1
```

(3) And you can get antennas in certain domain or the restrictions of certain two antennas via input a specific value.
For example:

```
Please input the domain:
5
141
142
251
252
293
294
```

(4) Now, we have gotten the values we need, and we will input the values into the other Java project (DFS/src/relation/TestSearch).
For example:

- We will initialize the list ‘frequence’via the frequencies distributed in certain domain.

```
11      static int[] frequence = {6,142,170,240,380,408,478};
```

- We initialize the different antennas, and you can add the number of agents according to your needs.

```
49          Agent agent1=new Agent("141",0);
50          Agent agent2=new Agent("142",1);
51          Agent agent3=new Agent("251",2);
52          Agent agent4=new Agent("252",3);
53          Agent agent5=new Agent("293",4);
54          Agent agent6=new Agent("294",5);
```

- We will define the neighbors of every agent by it’s restrictions.

```
56          agent1.set_voisinlist("142");
57          agent1.set_voisinlist("251");
58          agent1.set_voisinlist("252");
59
60          agent2.set_voisinlist("141");
61          agent2.set_voisinlist("251");
62          agent2.set_voisinlist("252");
63
64          agent3.set_voisinlist("141");
65          agent3.set_voisinlist("142");
66          agent3.set_voisinlist("252");
67
68          agent4.set_voisinlist("141");
69          agent4.set_voisinlist("142");
70          agent4.set_voisinlist("251");
71
72          agent5.set_voisinlist("294");
73
74          agent6.set_voisinlist("293");
```

- Here, n and e represent the number of agents and their relations and label represent the name of antennas.

```
86          int n=6,e=7;
87          String labels[]={"141","142","251","252","293","294"};
```

- We initialize the restrictions of every two antennas, the last character ‘1’ represent the weight.

```
92          graph.insertEdge(0, 1, 1);
93          graph.insertEdge(0, 2, 1);
94          graph.insertEdge(0, 3, 1);
95          graph.insertEdge(1, 2, 1);
96          graph.insertEdge(1, 3, 1);
97          graph.insertEdge(2, 3, 1);
98          graph.insertEdge(4, 5, 1);
```

(5) After executing this java project, you can get the complete DFS tree.
For example:

```
agent: 141
parent: null
pseudoparent: []
enfant: [142]
pseudoenfant: [251, 252]
..........
agent: 142
parent: 141
pseudoparent: []
enfant: [251]
pseudoenfant: [252]
..........
agent: 251
parent: 142
pseudoparent: [141]
enfant: [252]
pseudoenfant: []
..........
agent: 252
parent: 251
pseudoparent: [142, 141]
enfant: []
pseudoenfant: []
```

(6) You can get the parent, pseudo-parent, children and pseudo-children of certain agent via input the agent you want to know.
For example:

```
Please input the label:
142
The frequence inputed is: 142
agent: 142
parent: 141
pseudoparent: []
enfant: [251]
pseudoenfant: [252]
```

## *Part2 create the multi-agent systems.*

Then we will use jacamo to generate our multi-agent.

(7) Firstly we create the jcm file to declare our all agents and workspace.

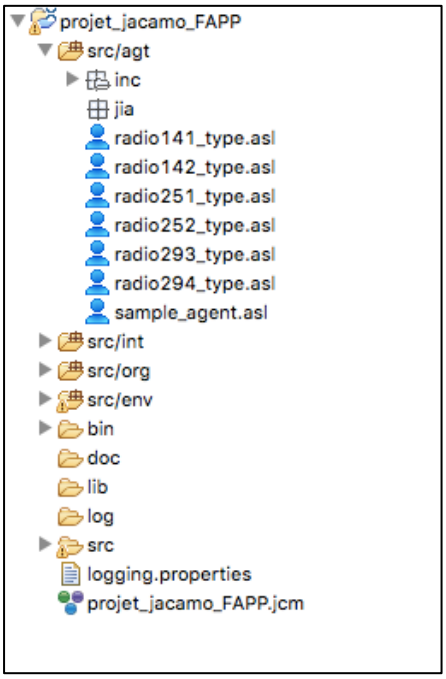
*Ref:projet_jacamo_FAPPA.jcm*

projet_jacamo_FAPP.jcm

```
*/

mas projet_jacamo_FAPP {

    agent  radio141 : radio141_type.asl {
    beliefs: message("this is 141")
}
    agent  radio142 : radio142_type.asl {
        beliefs: message("this is 142")
    }
    agent  radio251 : radio251_type.asl {
        beliefs: message("this is 251")
    }
    agent  radio252 : radio252_type.asl {
        beliefs: message("this is 252")
    }
    agent  radio293 : radio293_type.asl {
        beliefs: message("this is 293")
    }

    agent  radio294 : radio294_type.asl {
        beliefs: message("this is 294")
    }

     workspace radio_env {
        artifact fapp : projet_jacamo_FAPP.FappArtifact {
            focused-by: radio141, radio142, radio251, radio252, radio293,radio294
            }
            }
```

(8) Secondly we create 5 agents files and each file means a radio station.

*Ref: radio141_tyoe.asl; radio142_tyoe.asl; radio251_tyoe.asl; radio252_tyoe.asl; radio293_tyoe.asl; radio294_tyoe.asl*



(9) In each file we will definite the relationship between the agents.

For example:

*Ref: radio142_type.asl*

```
// Agent radio142_type in project projet_jacamo_FAPP

/* Initial beliefs and rules */
parent(radio141).
children(radio251).
pseudoparent(0).
pseudochildren(radio252).

/* Initial goals */
!start.
//!my_cost_function.

/* Plans */
+!start : true <- .print("radio142 is started.").


{ include("$jacamoJar/templates/common-cartago.asl") }
{ include("$jacamoJar/templates/common-moise.asl") }

// uncomment the include below to have a agent that always complies with its organization
{ include("$jacamoJar/templates/org-obedient.asl") }
{ include("./inc/dpop.asl") }

```

(10) Create the dpop.asl file, and here we will declare the **algorithm Dpop** in this file.

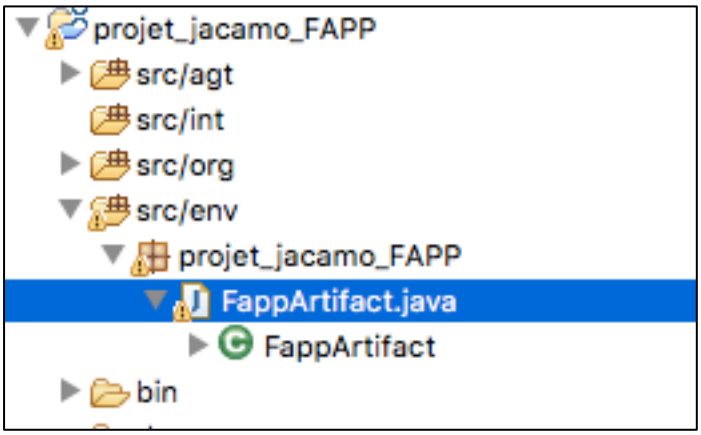
*Ref: dpop.asl*

projet_jacamo_FAPP.jcm | radio141_type.asl | radio142_type.asl | dpop.asl

```
// Agent dpop in project projet_jacamo_FAPP

/* Initial beliefs and rules */

/* Initial goals */
!findparent.
!findpseudoparent.
!findchildren.
!findpseudochildren.
!do_calcule_costfunction.
!send_my_costfunction.
!send_message_value.


/* Plans */
// Initialization parent pseudoparent children pseudochildren.
+!findparent:true <- ?parent(Parent);.print("my parent is ",Parent).
+!findpseudoparent:true <-?pseudoparent(Pseudoparent);.print("my Pseudoparent is ",Pseudoparent).
+!findchildren:true<-?children(Children);.print("my children is ",Children).
+!findpseudochildren:true<-?pseudochildren(Pseudochildren);.print("my pseudochildren is ",Pseudochildren).

@send_my_costfunction1
+!send_my_costfunction:children(Children)&Children==0&pseudochildren(Pseudochildren)&Pseudochildren==0 <-?par

@send_my_costfunction2
+!send_my_costfunction:parent(Parent)&Parent\==0 <-?parent(Parent);.send(Parent,tell,cost_function).

@do_calcule_costfunction1
+!do_calcule_costfunction:children(Children)&Children==0&pseudochildren(Pseudochildren)&Pseudochildren==0 <-

@do_calcule_costfunction2
+!do_calcule_costfunction:children(Children)&cost_function[source(Children)] <- .print("here is calcule cost

@do_calcule_my_value1
+!do_calcule_my_value:parent_value(Parent_value) <- .print("here is calcule my value fonction");?parent_value

@do_calcule_my_value2
+!do_calcule_my_value:pseudoparent_value(Pseudoparent_value) <- .print("here is calcule my value fonction").
```

(11) Create the Artifact file, and here we will declare the **all operation action** in this file.

*Ref: FappArtifact.java*




(12) And then run our project, we can observe the processus of **Dpop** in the console of each agent.