Spring School on Integrated Operational Problems

May 14-16, 2018, Troyes, France

PLAN

1rst session

Introduction (15-30 minutes)

C++ (1 hour)

- The Clarke and Wright algorithm
- Modifying the CAW and testing it
- Adding your own strategy to the CAW
- How to debug ?

2nd session

JSON (30 minutes)

- Understanding the role of the JSON files
- Adding two new algorithms to the CAW

PHP (30 minutes)

- Adding a new problem to the web-site

Conclusion will be yours !

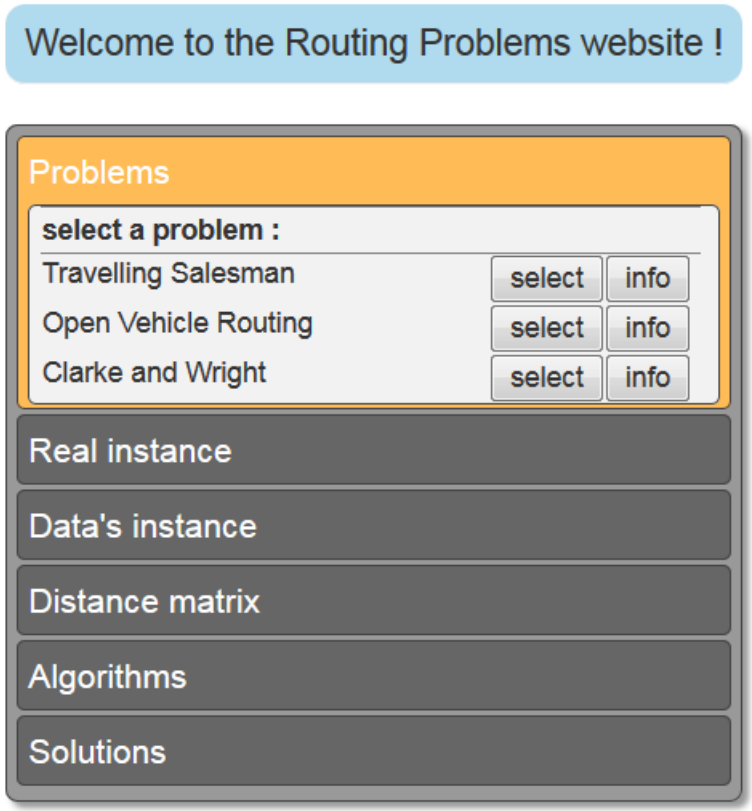
Pierre Bomel & Marc Sevaux
Université de Bretagne Sud,
Lab-STICC, Lorient, France, 2018

JSON or

"How to customize the web-services ?"

JSON customization

A JSON file contains the description
of the list of problems the web-site offers.



`<dir>\services\problems.json`

JSON customization

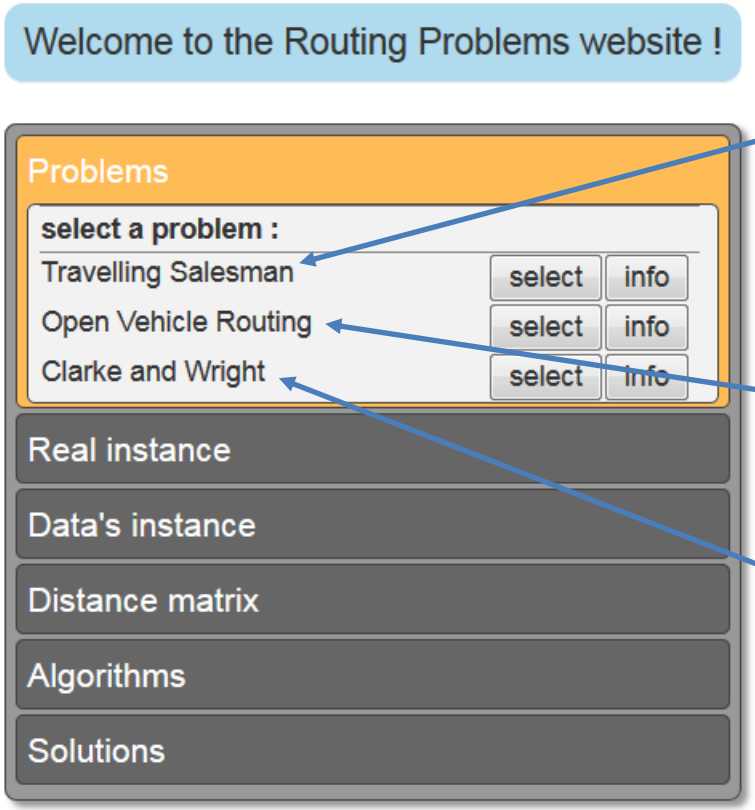
We cannot (and do not want to) teach you full JSON syntax in 20 minutes.
We'll give you here a simple procedure to follow « blindly ».

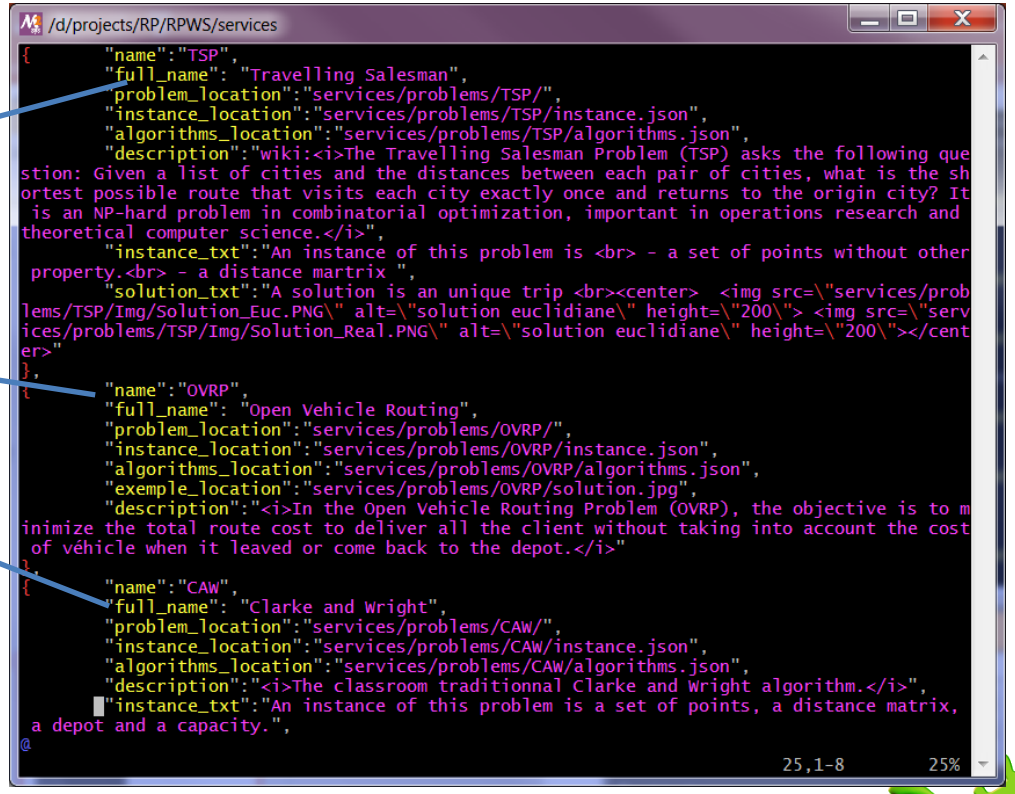
Hopefully, it's not so complex.

But, if you want to acquire more knowledge about JSON
have a look on Internet, there are thousands of on-line documentations about it.

JSON customization

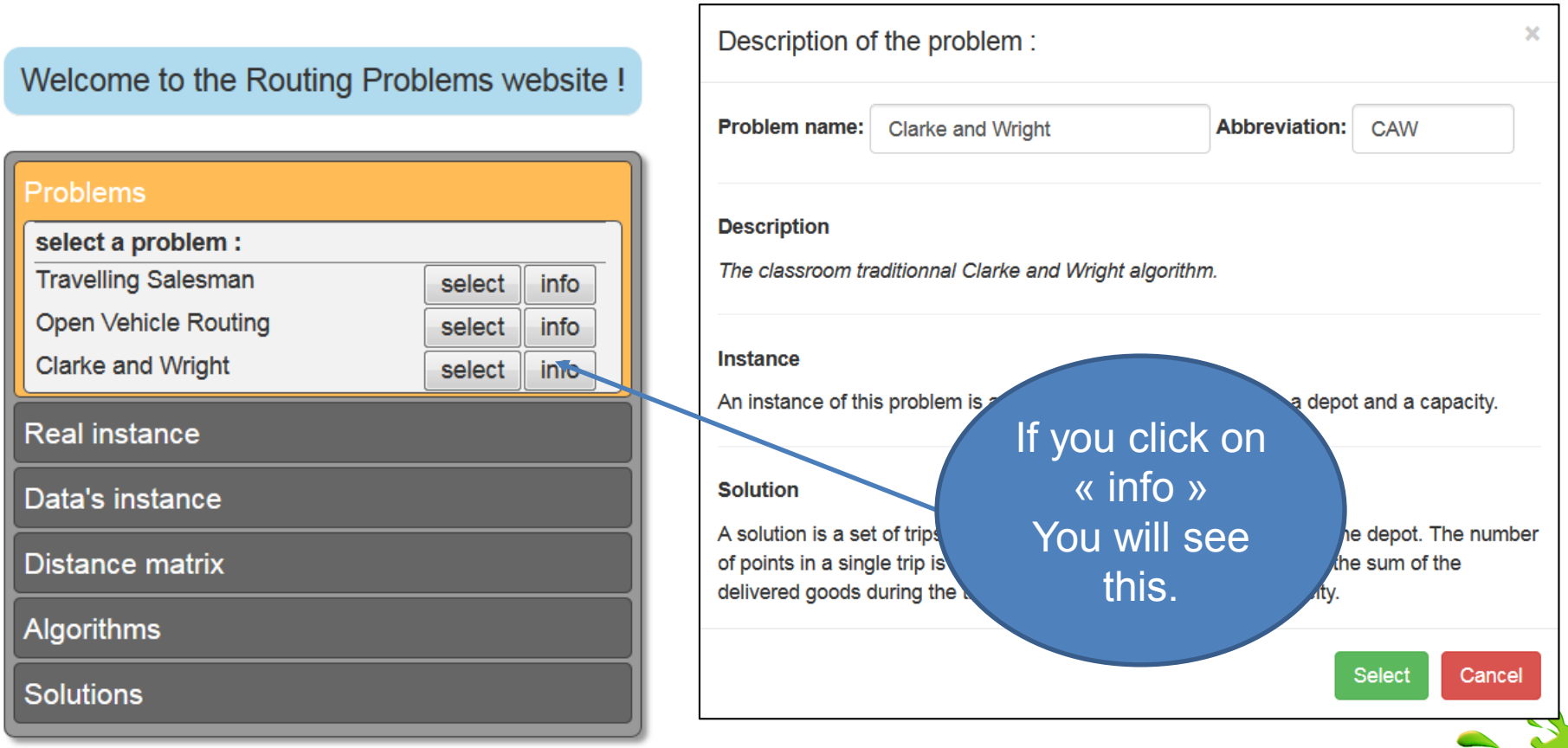
Each element of the JSON array in `problems.json` describes one web-service, with the help of several attributes.





JSON customization

Some will be displayed on the screen to help the user.



JSON customization

Some will be used « internally » to find the algorithms and data model definitions.

```
"problem_location":"services/problems/TSP/",
"instance_location":"services/problems/TSP/instance.json",
"algorithms_location":"services/problems/TSP/algorithms.json",
```

If you remember the structure of the files under `<dir>`
you will instantaneously understand that the values of

`problem_location`
`instance_location`
and
`algorithms_location`

are « pointers » (or path references) to the files stored bellow in the hierarchy.

JSON customization - problem

Let's create a new problem.
Go at `<dir>\services\problems`
and duplicate the `CAW` directory into `WAC`

/cygdrive/d/projects/RP/RPWS/services/problems

```
bomel@UBS-19760 /cygdrive/d/projects/RP/RPWS/services
$ ls
CVS  DMZ  problems  problems.json

bomel@UBS-19760 /cygdrive/d/projects/RP/RPWS/services
$ ls
CVS  DMZ  problems  problems.json

bomel@UBS-19760 /cygdrive/d/projects/RP/RPWS/services
$ cd problems

bomel@UBS-19760 /cygdrive/d/projects/RP/RPWS/services/problems
$ ls
CAW  CVS  DARP  hello  hellox  line  OVRP  rev  TSP  txt  VRP

bomel@UBS-19760 /cygdrive/d/projects/RP/RPWS/services/problems
$ cp -r CAW WAC

bomel@UBS-19760 /cygdrive/d/projects/RP/RPWS/services/problems
$ ls
CAW  CVS  DARP  hello  hellox  line  OVRP  rev  TSP  txt  VRP  WAC

bomel@UBS-19760 /cygdrive/d/projects/RP/RPWS/services/problems
$
```

JSON customization - problem

One step « upstairs »
add at the end of the `problems.json` file
a copy of the description of `CAW`
and modify it to become `WAC`

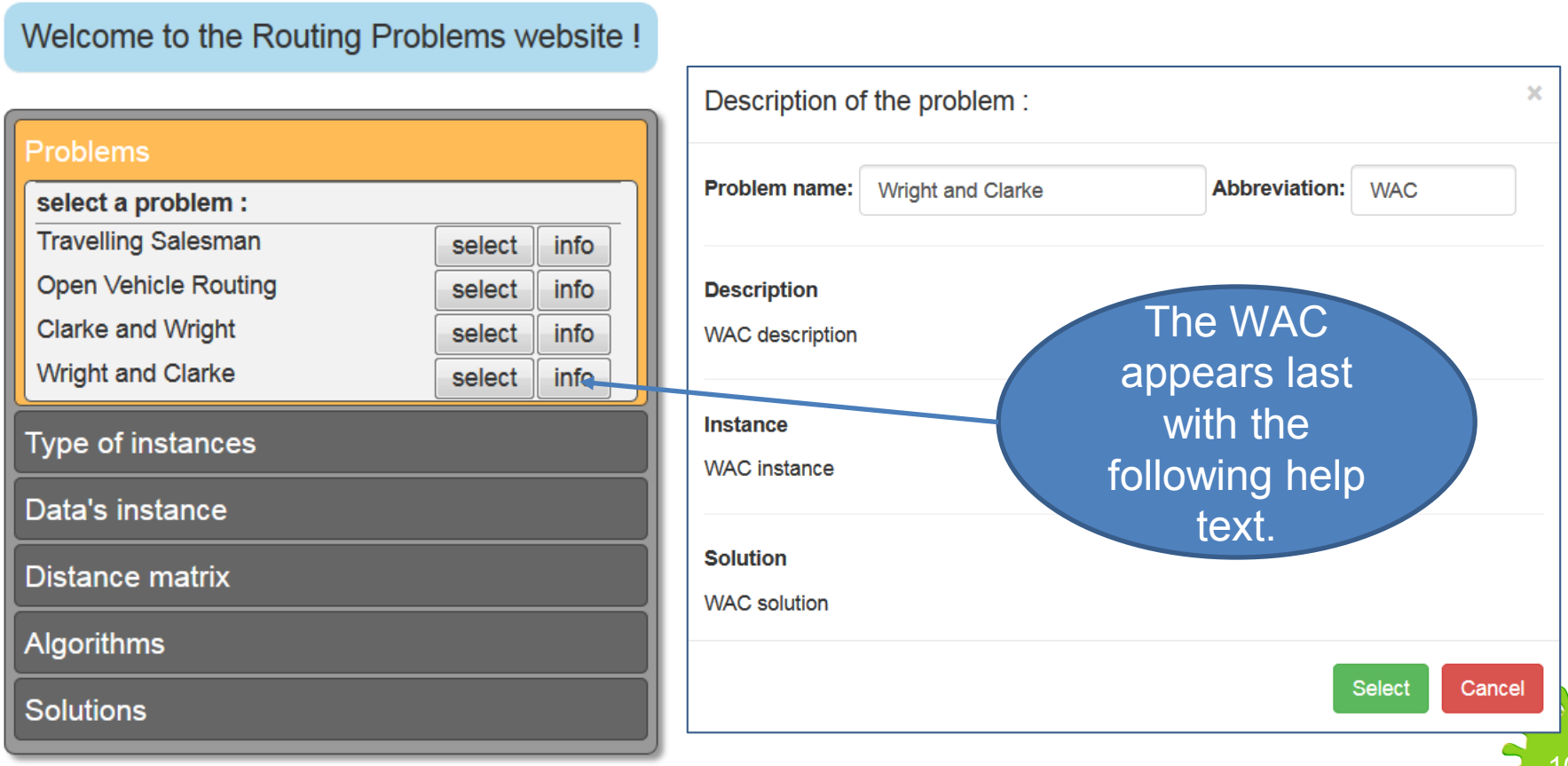
/d/projects/RP/RPWS/services

```
        "description":"<i>In the Open Vehicle Routing Problem (OVRP), the objective is
to minimize the total route cost to deliver all the client without taking into account
the cost of véhicle when it leaved or come back to the depot.</i>"
},
{       "name":"CAW",
        "full_name": "Clarke and Wright",
        "problem_location":"services/problems/CAW/",
        "instance_location":"services/problems/CAW/instance.json",
        "algorithms_location":"services/problems/CAW/algorithms.json",
        "description":"<i>The classroom traditionnal Clarke and Wright algorithm.</i>",
        "instance_txt":"An instance of this problem is a set of points, a distance matr
ix,  a depot and a capacity.",
        "solution_txt":"A solution is a set of trips from the depot to some points and
back to the depot. The number of points in a single trip is limited by the vehicule's c
apacity because the sum of the delivered goods during the trip cannot exceed the vehicu
le's capacity."
},
{       "name":"WAC",
        "full_name": "Wright and Clarke",
        "problem_location":"services/problems/WAC/",
        "instance_location":"services/problems/WAC/instance.json",
        "algorithms_location":"services/problems/WAC/algorithms.json",
        "description":"WAC description",
        "instance_txt":"WAC instance",
        "solution_txt":"WAC solution"
}
]
                                                                    20,1-8        Bas
```

Don’t forget to add a column here !

JSON customization - problem

Restart your web-browser.
It should look like this now.



JSON customization - algorithm

Because `WAC` is a strict copy of `CAW`, everything will be identical to `CAW` if you decide to play with the `WAC` problem. So, no need to test this.

Don't forget to add a column here !

To deeper customize our web-services, instead of adding more problems, we now want to add more algorithms. We want to add a second algorithm to `WAC`. Modify the `WAC\algorithms.json` file like this.

/d/projects/RP/RPWS/services/problems/WAC

```
[
        {       "name":"WAC1",
                "full_name":"Wright and Clarke 1",
                "description":"Single vehicule, limited capacity, customer delivery planning of goods",
                "time":"<10s",
                "rank":"1",
                "optimality":"no",
                "url":"services/problems/WAC/WAC1.php"
        },
        {       "name":"WAC2",
                "full_name":"Wright and Clarke 2",
                "description":"Single vehicule, limited capacity, customer delivery planning of goods",
                "time":"<10s",
                "rank":"1",
                "optimality":"no",
                "url":"services/problems/WAC/WAC2.php"
        }
]
                                                                2,17-30       Tout
```

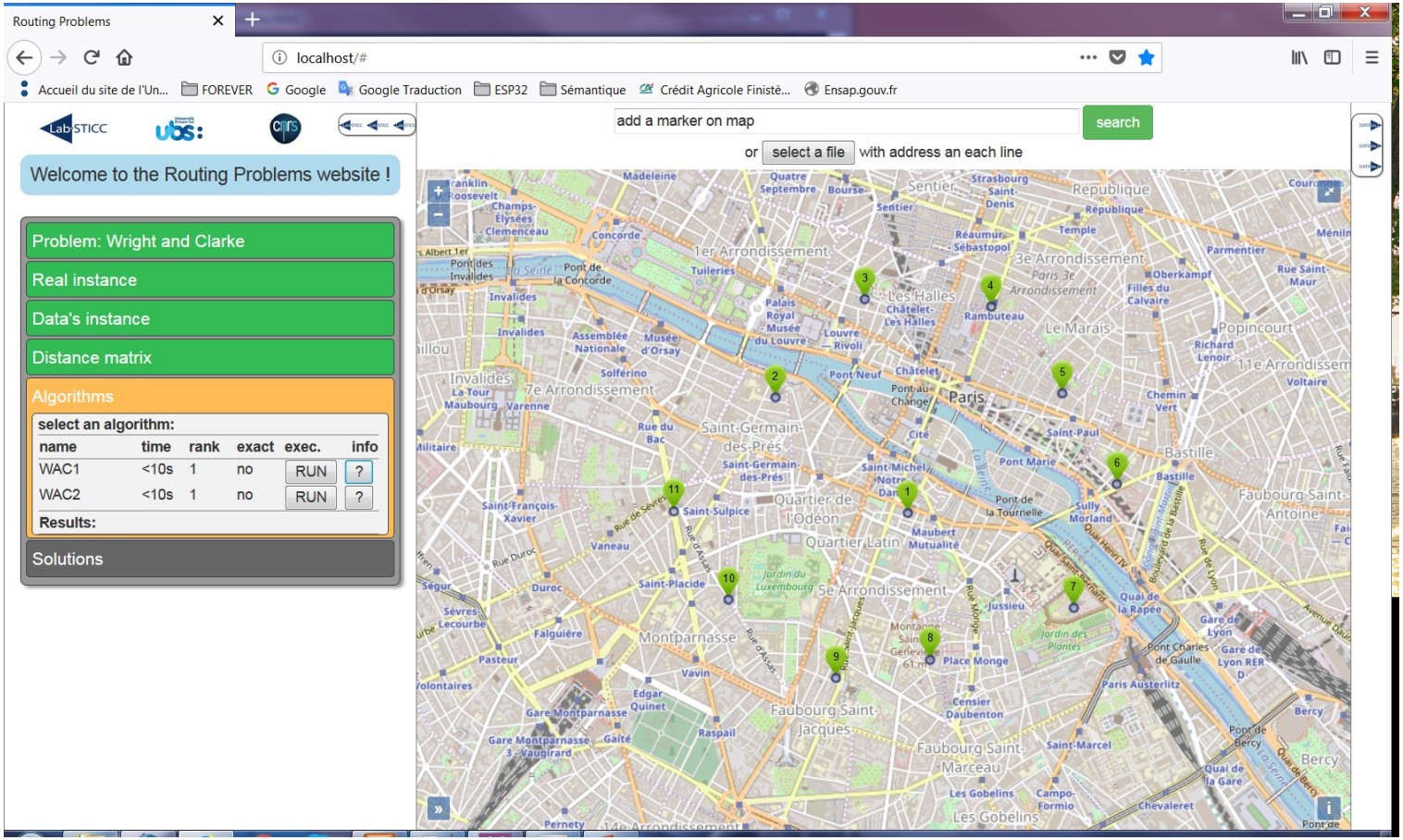
JSON customization - algorithm

Rename `CAW.php` into `WAC1.php`
Duplicate `WAC1.php` into `WAC2.php`
Don't modify `WAC1.php` and `WAC2.php`,
for they execute the same `CAW` algorithm located under `program`.

```
/cygdrive/d/projects/RP/RPWS/services/problems/WAC
bomel@UBS-19760 /cygdrive/d/projects/RP/RPWS/services/problems/WAC
$ ls
algorithms.json  CAW.php  instance.json  program

bomel@UBS-19760 /cygdrive/d/projects/RP/RPWS/services/problems/WAC
$ ll
total 17
-rwxr-xr-x  1 bomel None  253  3 avr.  13:41 algorithms.json
-rwxr-xr-x  1 bomel None 5502  3 avr.  13:41 CAW.php
-rwxr-xr-x  1 bomel None  675  3 avr.  13:41 instance.json
drwxr-xr-x+ 1 bomel None    0  3 avr.  13:41 program

bomel@UBS-19760 /cygdrive/d/projects/RP/RPWS/services/problems/WAC
$ mv CAW.php WAC1.php

bomel@UBS-19760 /cygdrive/d/projects/RP/RPWS/services/problems/WAC
$ cp WAC1.php WAC2.php

bomel@UBS-19760 /cygdrive/d/projects/RP/RPWS/services/problems/WAC
$ ls
algorithms.json  instance.json  program  WAC1.php  WAC2.php

bomel@UBS-19760 /cygdrive/d/projects/RP/RPWS/services/problems/WAC
$
```

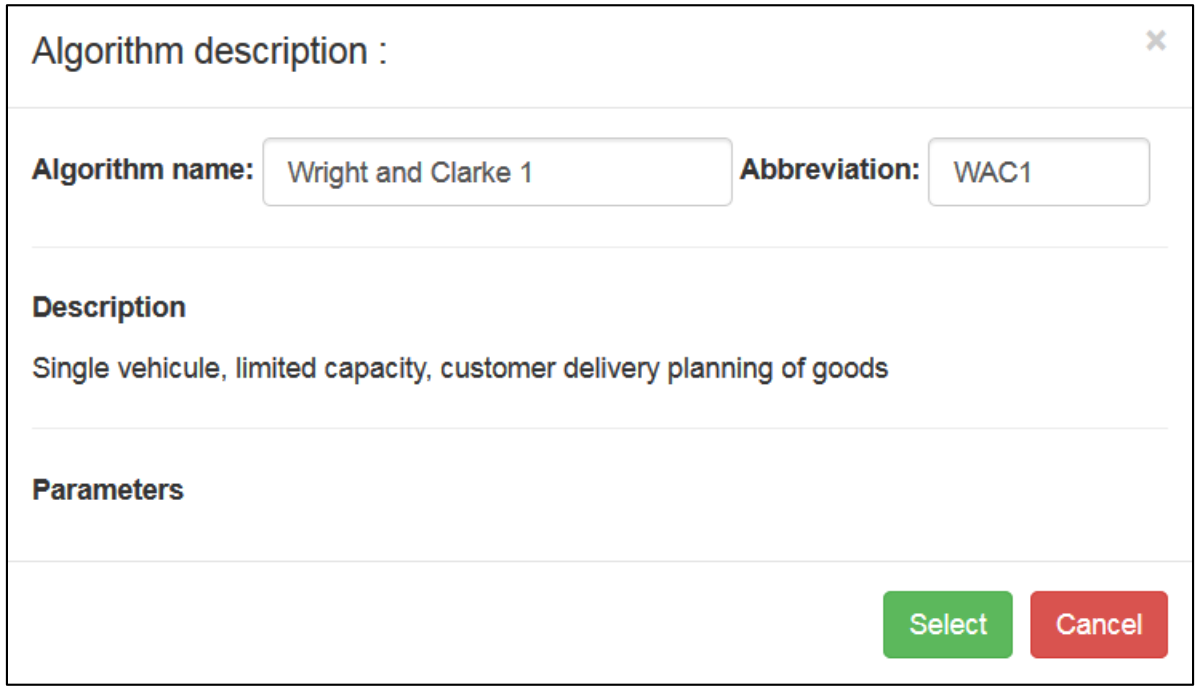
JSON customization - algorithm

Restart your web-browser, and select the WAC problem until you reach the algorithms sub-window. You should see this now.



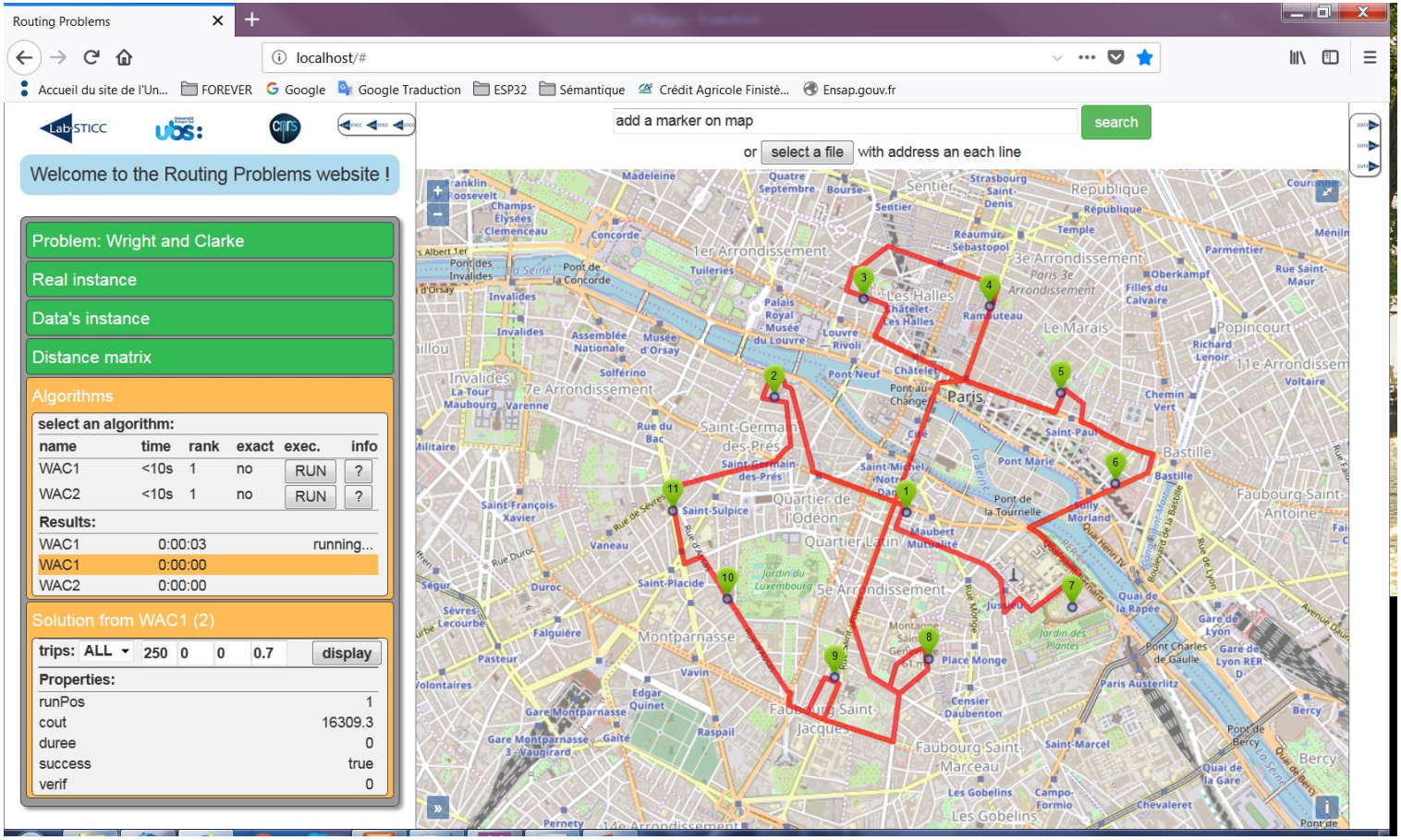
JSON customization - algorithm

If you click on « ? » you will see some helping text displayed.



JSON customization - algorithm

Click on « run » for WAC1 and WAC2 to see the resulting paths.
Of course, both will display the same paths, because they execute the same code.



JSON customization - algorithm

We want to create a different executable for `WAC2`

Duplicate `WAC\program` into `program2`

```
/cygdrive/d/projects/RP/RPWS/services/problems/...
$ ls
algorithms.json  algorithms.json~  instance.json  pro

bomel@UBS-19760 /cygdrive/d/projects/RP/RPWS/services
$ cp -r program program2

bomel@UBS-19760 /cygdrive/d/projects/RP/RPWS/services
$ ls
algorithms.json   instance.json  program2  WAC2.php
algorithms.json~  program        WAC1.php

bomel@UBS-19760 /cygdrive/d/projects/RP/RPWS/services
$
```

Modify `WAC\program\main.cc` to call `Clarke_and_Wrigth()`
Modify `WAC\program2\main.cc` to call `Random_Clarke_and_Wrigth()`
Compile and test them.

```
/d/projects/RP/RPWS/services/problems/WAC/program

        // INPUTS
        IO::load_params(param, merges, depot_id, goods, paths);

        // COMPUTATION
        // keep the one you want to test !!!!
        Clarke_and_Wright();
        //Random_Clarke_and_Wright();
        //My_Clarke_and_Wright();

        // OUTPUTS
        IO::store_results(res, depot_id, goods, paths);

        // CLEAN END
        param.close();
                                                296,1-8       98%
```

```
/d/projects/RP/RPWS/services/problems/WAC/program

        // INPUTS
        IO::load_params(param, merges, depot_id, goods, paths);

        // COMPUTATION
        // keep the one you want to test !!!!
        //Clarke_and_Wright();
        Random_Clarke_and_Wright();
        //My_Clarke_and_Wright();

        // OUTPUTS
        IO::store_results(res, depot_id, goods, paths);

        // CLEAN END
        param.close();
<ain.cc" [unix] 309L, 10977C écrit(s)           297,2-9       98%
```

JSON customization - algorithm

Modify `WAC\WAC2.php` file like this to point on the good executable, which is now located under `WAC\program2`.

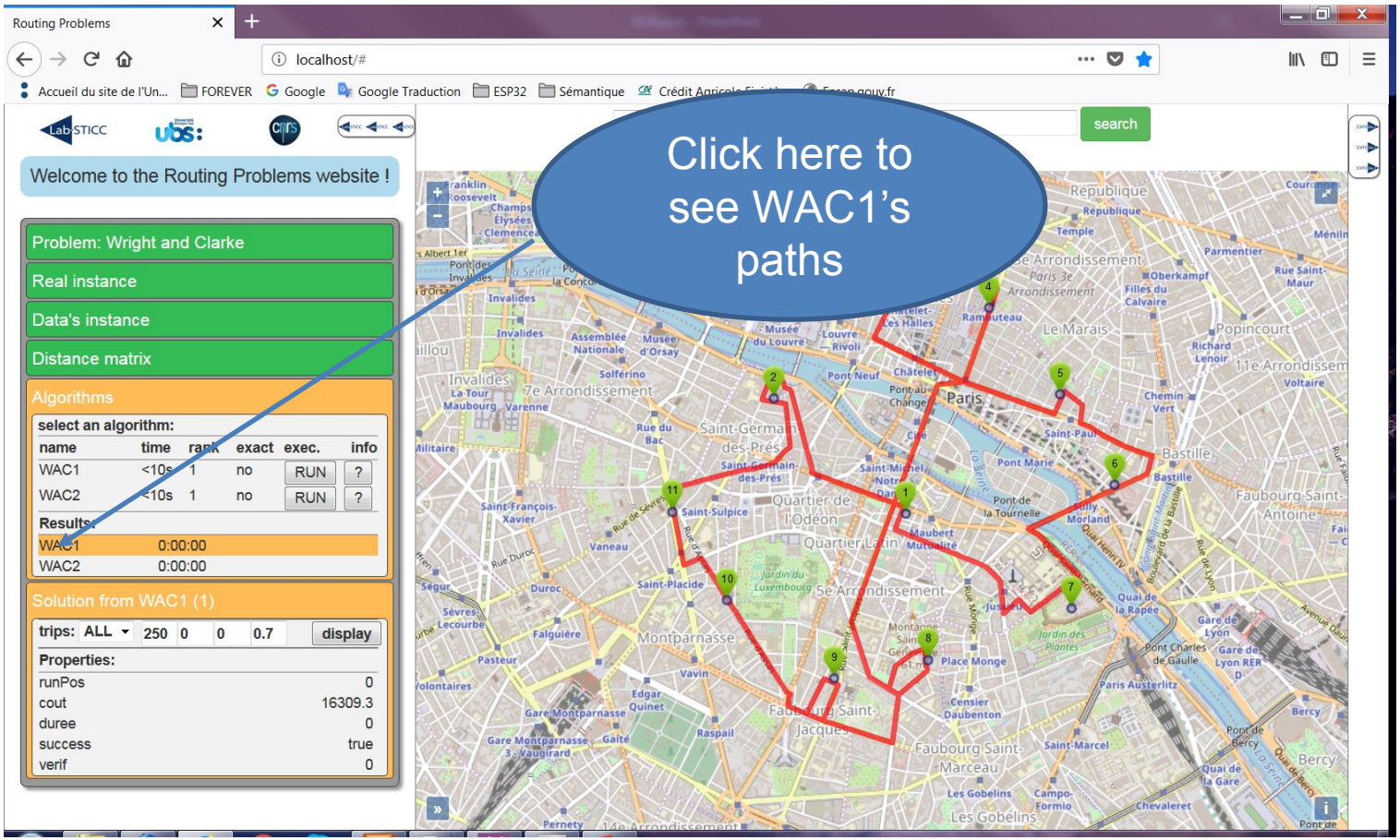
/d/projects/RP/RPWS/services/problems/WAC

```
                for ($j = $min; $j < $max; $j++) {
                        fputs($fileinstance, "".$matrix[$i][$j]->distance." ");
                }
                fputs($fileinstance, "\n") ;
        }
        # Close the file
        fclose($fileinstance);

        // RUN the CAW executable: Linux only
        //exec("program2/caw ".$name_inst." ".$name_sol);
        // RUN the CAW executable: Windows only
        exec("program2\caw.exe ".$name_inst." ".$name_sol);

        if (file_exists($name_sol)) {

                // init solution with default values
                $sol = new stdClass;
                $sol->runPos = $json_data->runPos;
                $sol->cout = 0;
                                                        110,16-30      59%
```

(You will learn more about PHP coding, just after the JSON part)

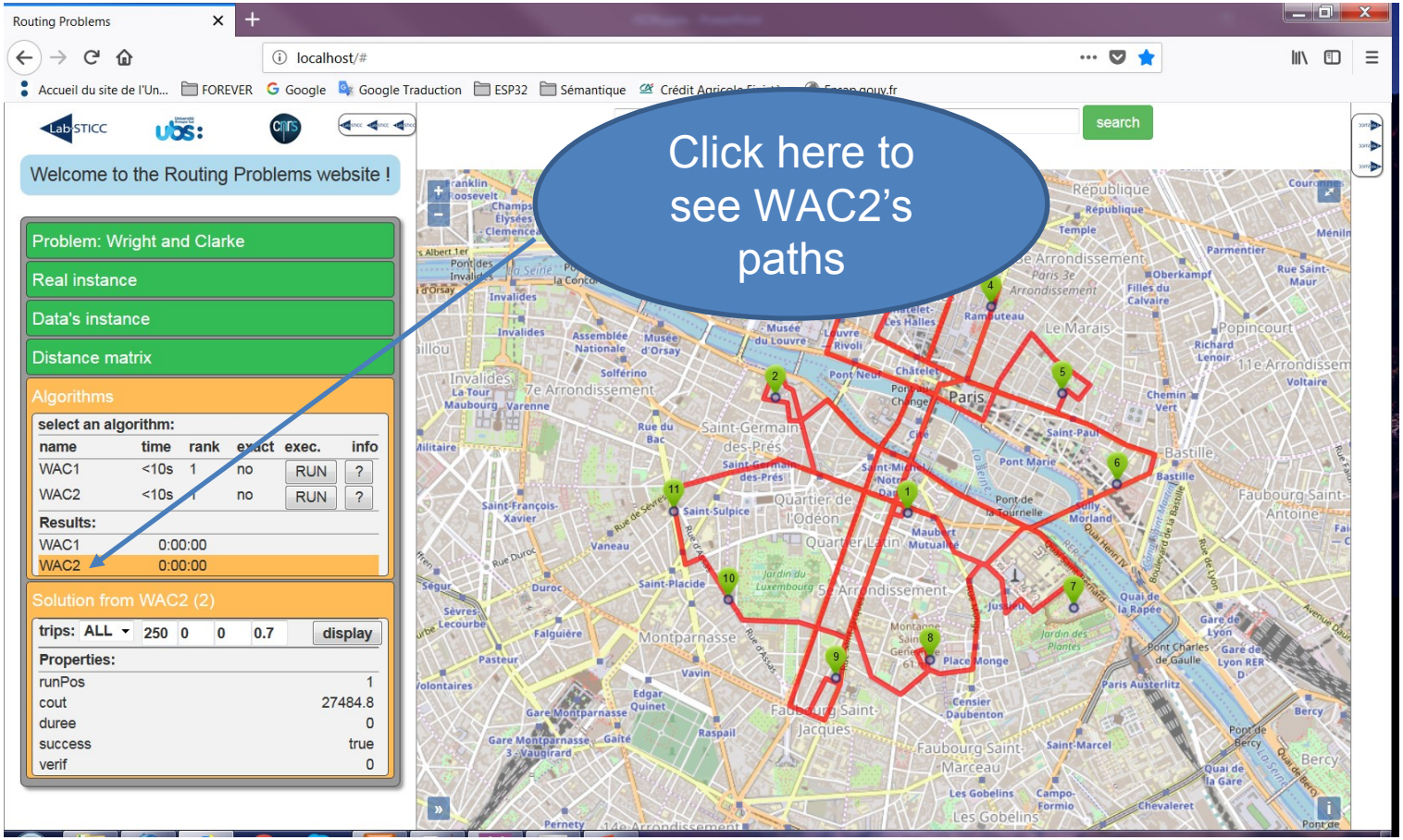
JSON customization - algorithm

Restart your browser, and click on « run »
for WAC1 and WAC2 to see the resulting paths.
Now the paths are different and you can compare them easily.



JSON customization - algorithm

Restart your browser, and click on « run »
for WAC1 and WAC2 to see the resulting paths.
Now the paths are different and you can compare them easily.



Spring School on Integrated Operational Problems

May 14-16, 2018, Troyes, France

PLAN

1rst session

Introduction (15-30 minutes)

C++ (1 hour)

- The Clarke and Wright algorithm
- Modifying the CAW and testing it
- Adding your own strategy to the CAW
- How to debug ?

2nd session

JSON (30 minutes)

- Understanding the role of the JSON files
- Adding two new algorithms to the CAW

PHP (30 minutes)

- Adding a new problem to the web-site

Conclusion will be yours !

Pierre Bomel & Marc Sevaux
Université de Bretagne Sud,
Lab-STICC, Lorient, France, 2018