Documentation for using the Webservice

„Routeoptimization with AntColonyOptimization as Single-Depot-Algorithm“

1. Config-File

The configurations file is called „config.properties“ and is found on the highest folder hierarchy of the project (same level as src, target, etc.).

Specified values:

* mDbHost is the Host of the MySQL-Server which contains the distance-matrixes and the trained pheromone-levels between the locations. Possible values are like „111.111.111.111“ or „www.wikipedia.org“.
* mDbName ist he name oft he used MySQL-Database. At the moment the database is called „distances“.
* mDbUser is a user which has the right to access and modify the database specified in mDbName.
* mDbPwd is the password of mDbUser to access the database.

1. Starting parameters

Setting the port of the application (examples):

* java -Dserver.port=8080 -jar executable.jar
* java -jar executable.jar –server.port=8080

Other parameters are specified in class src/main/java/de/adrianwilke/acotspjava/Application.java

There are 3 working starting-parameters:

-- min (int): Minimum number of threads used by the webservice before another server may start it’s corresponding webservice for route calculations. (default: 1)

-- max (int): Maximum number of threads used by the webservice so we don’t overload the server (default: 2)

-- opt (int): Optimum number of threads running at the same time (default: 4)

1. Parameter of the REST-Service

Accessing the REST-Service: *host:port***/single-depot/aco/***set\_parameters*

Specified in Class src/main/java/de/adrianwilke/acotspjava/RESTController.java

* acs (boolean): apply ant\_colony colony system (default: true)
* tries (int): number of independent trials (default: 1)
* maxSecondsPerTry (int): maximum time for each trial (default: 5)
* quietMode (boolean): reduce output to a minimum, no extra files are created (default: true)
* startLocationId (long): locationId of the start location as the single depot (optional)
* locationsToVisit (String i.e. “1,3,5,6,7,17“ visit these locations, must include start location) – non-optional & no default-value

the needed time of the algorithm = tries \* maxSecondsPerTry

1. Return type of the REST-Service

JSON-String:

* tourLength (long): distance in meters
* route (String i.e. "[4,2,6,9,4]") with the optimized order of the locations.
* done (boolean): true if the optimization finished correctly
* cancelled (boolean): false if the optimization wasn’t aborted