ExpositoTOP

Generated by Doxygen 1.9.2

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 Package es.ull.esit.utilities

Classes

- class BellmanFord
- class ExpositoUtilities
- class PowerSet

5.2 Package es.ull.esit.utils

Classes

- class Pair

5.3 Package top

Classes

- class mainTOPTW
- class TOPTW
- class TOPTWEvaluator
- class TOPTWGRASP
- class TOPTWReader
- class TOPTWRoute
- class TOPTWSolution

Chapter 6

Class Documentation

6.1 es.ull.esit.utilities.BellmanFord Class Reference

Public Member Functions

- BellmanFord (int[ ][ ] distanceMatrix, int nodes, ArrayList< Integer > path)
- int[ ] getDistances ()
- int getValue ()
- void solve ()

6.1.1 Detailed Description

Definition at line 5 of file BellmanFord.java.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 BellmanFord()

```
es.ull.esit.utilities.BellmanFord.BellmanFord (
            int distanceMatrix[][],
            int nodes,
            ArrayList< Integer > path )
```

Parameters

distanceMatrix	
nodes	
path	

Definition at line 46 of file BellmanFord.java.

6.1.3 Member Function Documentation

6.1.3.1 getDistances()

```
int[] es.ull.esit.utilities.BellmanFord.getDistances ( )
```

Returns

Definition at line 74 of file BellmanFord.java.

6.1.3.2 getValue()

```
int es.ull.esit.utilities.BellmanFord.getValue ( )
```

Returns

Definition at line 82 of file BellmanFord.java.

6.1.3.3 solve()

```
void es.ull.esit.utilities.BellmanFord.solve ( )
```

Definition at line 89 of file BellmanFord.java.

The documentation for this class was generated from the following file:

- src/main/java/es/ull/esit/utilities/BellmanFord.java

6.2 es.ull.esit.utilities.ExpositoUtilities Class Reference

Static Public Member Functions

- static void printFile (String file)
- static String simplifyString (String string)
- static double[ ][ ] multiplyMatrices (double a[ ][ ], double b[ ][ ])
- static void writeTextToFile (String file, String text) throws IOException
- static String getFormat (String string)
- static String getFormat (double value)
- static String getFormat (double value, int zeros)
- static String getFormat (String string, int width)
- static String getFormat (String string, int width, int alignment)
- static String getFormat (ArrayList< String > strings, int width)
- static String getFormat (ArrayList< Integer > strings)
- static String getFormat (String[ ] strings, int width)
- static String getFormat (String[ ][ ] matrixStrings, int width)
- static String getFormat (String[ ] strings)
- static String getFormat (String[ ] strings, int[ ] width)
- static String getFormat (String[ ] strings, int[ ] width, int[ ] alignment)
- static boolean isInteger (String str)
- static boolean isDouble (String str)
- static boolean isAcyclic (int[ ][ ] distanceMatrix)
- static boolean thereIsPath (int[ ][ ] distanceMatrix, int node)

Static Public Attributes

- static final int DEFAULT_COLUMN_WIDTH = 10
- static final int ALIGNMENT_LEFT = 1
- static final int ALIGNMENT_RIGHT = 2

6.2.1 Detailed Description

Definition at line 17 of file ExpositoUtilities.java.

6.2.2 Member Function Documentation

6.2.2.1 getFormat() [1/12]

```
static String es.ull.esit.utilities.ExpositoUtilities.getFormat (
            ArrayList< Integer > strings )  [static]
```

Definition at line 147 of file ExpositoUtilities.java.

6.2.2.2 getFormat() [2/12]

```
static String es.ull.esit.utilities.ExpositoUtilities.getFormat (
            ArrayList< String > strings,
            int width )  [static]
```

Definition at line 135 of file ExpositoUtilities.java.

6.2.2.3 getFormat() [3/12]

```
static String es.ull.esit.utilities.ExpositoUtilities.getFormat (
            double value )  [static]
```

Definition at line 98 of file ExpositoUtilities.java.

6.2.2.4 getFormat() [4/12]

```
static String es.ull.esit.utilities.ExpositoUtilities.getFormat (
            double value,
            int zeros )  [static]
```

Definition at line 106 of file ExpositoUtilities.java.

6.2.2.5 getFormat() [5/12]

```
static String es.ull.esit.utilities.ExpositoUtilities.getFormat (
            String string )  [static]
```

Definition at line 88 of file ExpositoUtilities.java.

6.2.2.6 getFormat() [6/12]

```
static String es.ull.esit.utilities.ExpositoUtilities.getFormat (
            String string,
            int width )  [static]
```

Definition at line 118 of file ExpositoUtilities.java.

6.2.2.7 getFormat() [7/12]

```
static String es.ull.esit.utilities.ExpositoUtilities.getFormat (
            String string,
            int width,
            int alignment )  [static]
```

Definition at line 122 of file ExpositoUtilities.java.

6.2.2.8 getFormat() [8/12]

```
static String es.ull.esit.utilities.ExpositoUtilities.getFormat (
            String[] strings )  [static]
```

Definition at line 183 of file ExpositoUtilities.java.

6.2.2.9 getFormat() [9/12]

```
static String es.ull.esit.utilities.ExpositoUtilities.getFormat (
            String[] strings,
            int width )  [static]
```

Definition at line 159 of file ExpositoUtilities.java.

6.2.2.10 getFormat() [10/12]

```
static String es.ull.esit.utilities.ExpositoUtilities.getFormat (
            String[] strings,
            int[] width )  [static]
```

Definition at line 191 of file ExpositoUtilities.java.

6.2.2.11 getFormat() [11/12]

```
static String es.ull.esit.utilities.ExpositoUtilities.getFormat (
            String[] strings,
            int[] width,
            int[] alignment )  [static]
```

Definition at line 197 of file ExpositoUtilities.java.

6.2.2.12 getFormat() [12/12]

```
static String es.ull.esit.utilities.ExpositoUtilities.getFormat (
            String matrixStrings[][],
            int width )  [static]
```

Definition at line 167 of file ExpositoUtilities.java.

6.2.2.13 isAcyclic()

```
static boolean es.ull.esit.utilities.ExpositoUtilities.isAcyclic (
            int distanceMatrix[][] )  [static]
```

Definition at line 231 of file ExpositoUtilities.java.

6.2.2.14 isDouble()

```
static boolean es.ull.esit.utilities.ExpositoUtilities.isDouble (
            String str )  [static]
```

Definition at line 222 of file ExpositoUtilities.java.

6.2.2.15 isInteger()

```
static boolean es.ull.esit.utilities.ExpositoUtilities.isInteger (
            String str )  [static]
```

Definition at line 213 of file ExpositoUtilities.java.

6.2.2.16 multiplyMatrices()

```
static double[][] es.ull.esit.utilities.ExpositoUtilities.multiplyMatrices (
            double a[][],
            double b[][] )  [static]
```

Definition at line 60 of file ExpositoUtilities.java.

6.2.2.17 printFile()

```
static void es.ull.esit.utilities.ExpositoUtilities.printFile (
            String file )  [static]
```

Definition at line 32 of file ExpositoUtilities.java.

6.2.2.18 simplifyString()

```
static String es.ull.esit.utilities.ExpositoUtilities.simplifyString (
            String string )  [static]
```

Definition at line 51 of file ExpositoUtilities.java.

6.2.2.19 thereIsPath()

```
static boolean es.ull.esit.utilities.ExpositoUtilities.thereIsPath (
            int distanceMatrix[][],
            int node )  [static]
```

Definition at line 244 of file ExpositoUtilities.java.

6.2.2.20 writeTextToFile()

```
static void es.ull.esit.utilities.ExpositoUtilities.writeTextToFile (
            String file,
            String text ) throws IOException  [static]
```

Definition at line 81 of file ExpositoUtilities.java.

6.2.3 Member Data Documentation

6.2.3.1 ALIGNMENT_LEFT

```
final int es.ull.esit.utilities.ExpositoUtilities.ALIGNMENT_LEFT = 1  [static]
```

Definition at line 20 of file ExpositoUtilities.java.

6.2.3.2 ALIGNMENT_RIGHT

```
final int es.ull.esit.utilities.ExpositoUtilities.ALIGNMENT_RIGHT = 2  [static]
```

Definition at line 21 of file ExpositoUtilities.java.

6.2.3.3 DEFAULT_COLUMN_WIDTH

```
final int es.ull.esit.utilities.ExpositoUtilities.DEFAULT_COLUMN_WIDTH = 10  [static]
```

Definition at line 19 of file ExpositoUtilities.java.

The documentation for this class was generated from the following file:

- src/main/java/es/ull/esit/utilities/ExpositoUtilities.java

6.3 top.mainTOPTW Class Reference

Static Public Member Functions

- static void main (String[ ] args)

6.3.1 Detailed Description

Definition at line 3 of file mainTOPTW.java.

6.3.2 Member Function Documentation

6.3.2.1 main()

```
static void top.mainTOPTW.main (
            String[ ] args )  [static]
```

Definition at line 5 of file mainTOPTW.java.

The documentation for this class was generated from the following file:

- src/main/java/top/mainTOPTW.java

6.4 es.ull.esit.utils.Pair< F, S > Class Template Reference

Public Member Functions

- Pair (F first, S second)
- boolean equals (Object o)
- int hashCode ()

Static Public Member Functions

- static< A, B > Pair< A, B > create (A a, B b)

Public Attributes

- final F first
- final S second

6.4.1 Detailed Description

Definition at line 4 of file Pair.java.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 Pair()

```
es.ull.esit.utils.Pair< F, S >.Pair (
            F first,
            S second )
```

Definition at line 8 of file Pair.java.

6.4.3 Member Function Documentation

6.4.3.1 create()

```
static< A, B > Pair< A, B > es.ull.esit.utils.Pair< F, S >.create (
            A a,
            B b )  [static]
```

Definition at line 27 of file Pair.java.

6.4.3.2 equals()

```
boolean es.ull.esit.utils.Pair< F, S >.equals (
            Object o )
```

Definition at line 14 of file Pair.java.

6.4.3.3 hashCode()

```
int es.ull.esit.utils.Pair< F, S >.hashCode ( )
```

Definition at line 23 of file Pair.java.

6.4.4 Member Data Documentation

6.4.4.1 first

```
final F es.ull.esit.utils.Pair< F, S >.first
```

Definition at line 5 of file Pair.java.

6.4.4.2 second

```
final S es.ull.esit.utils.Pair< F, S >.second
```

Definition at line 6 of file Pair.java.

The documentation for this class was generated from the following file:

- src/main/java/es/ull/esit/utils/Pair.java

6.5 es.ull.esit.utilities.PowerSet< E > Class Template Reference

Inherits Iterator< Set< E > >, and Iterable< Set< E > >.

Public Member Functions

- PowerSet (Set< E > set)
- boolean hasNext ()
- Set< E > next ()
- void remove ()
- Iterator< Set< E > > iterator ()

6.5.1 Detailed Description

Definition at line 9 of file PowerSet.java.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 PowerSet()

```
es.ull.esit.utilities.PowerSet< E >.PowerSet (
            Set< E > set )
```

Definition at line 15 of file PowerSet.java.

6.5.3 Member Function Documentation

6.5.3.1 hasNext()

```
boolean es.ull.esit.utilities.PowerSet< E >.hasNext ( )
```

Definition at line 21 of file PowerSet.java.

6.5.3.2 iterator()

```
Iterator< Set< E > > es.ull.esit.utilities.PowerSet< E >.iterator ( )
```

Definition at line 50 of file PowerSet.java.

6.5.3.3 next()

```
Set< E > es.ull.esit.utilities.PowerSet< E >.next ( )
```

Definition at line 26 of file PowerSet.java.

6.5.3.4 remove()

```
void es.ull.esit.utilities.PowerSet< E >.remove ( )
```

Definition at line 45 of file PowerSet.java.

The documentation for this class was generated from the following file:

- src/main/java/es/ull/esit/utilities/PowerSet.java

6.6 top.TOPTW Class Reference

Public Member Functions

- TOPTW (int nodes, int routes)
- boolean isDepot (int a)
- double getDistance (int[ ] route)
- double getDistance (ArrayList< Integer > route)
- double getDistance (ArrayList< Integer >[ ] routes)
- void calculateDistanceMatrix ()
- double getMaxTimePerRoute ()
- void setMaxTimePerRoute (double maxTimePerRoute)
- double getMaxRoutes ()
- void setMaxRoutes (double maxRoutes)
- int getPOIs ()
- double getDistance (int i, int j)
- double getTime (int i, int j)
- int getNodes ()
- void setNodes (int nodes)
- double getX (int index)
- void setX (int index, double x)
- double getY (int index)
- void setY (int index, double y)
- double getScore (int index)
- double[ ] getScore ()
- void setScore (int index, double score)
- double getReadyTime (int index)
- void setReadyTime (int index, double readyTime)
- double getDueTime (int index)
- void setDueTime (int index, double dueTime)
- double getServiceTime (int index)
- void setServiceTime (int index, double serviceTime)
- int getVehicles ()
- String toString ()
- int addNode ()
- int addNodeDepot ()

6.6.1 Detailed Description

Definition at line 8 of file TOPTW.java.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 TOPTW()

```
top.TOPTW.TOPTW (
            int nodes,
            int routes )
```

Definition at line 22 of file TOPTW.java.

6.6.3 Member Function Documentation

6.6.3.1 addNode()

```
int top.TOPTW.addNode ( )
```

Definition at line 224 of file TOPTW.java.

6.6.3.2 addNodeDepot()

```
int top.TOPTW.addNodeDepot ( )
```

Definition at line 229 of file TOPTW.java.

6.6.3.3 calculateDistanceMatrix()

```
void top.TOPTW.calculateDistanceMatrix ( )
```

Definition at line 77 of file TOPTW.java.

6.6.3.4 getDistance() [1/4]

```
double top.TOPTW.getDistance (
            ArrayList< Integer > route )
```

Definition at line 58 of file TOPTW.java.

6.6.3.5 getDistance() [2/4]

```
double top.TOPTW.getDistance (
            ArrayList< Integer >[] routes )
```

Definition at line 68 of file TOPTW.java.

6.6.3.6 getDistance() [3/4]

```
double top.TOPTW.getDistance (
            int i,
            int j )
```

Definition at line 112 of file TOPTW.java.

6.6.3.7 getDistance() [4/4]

```
double top.TOPTW.getDistance (
            int[] route )
```

Definition at line 48 of file TOPTW.java.

6.6.3.8 getDueTime()

```
double top.TOPTW.getDueTime (
            int index )
```

Definition at line 172 of file TOPTW.java.

6.6.3.9 getMaxRoutes()

```
double top.TOPTW.getMaxRoutes ( )
```

Definition at line 100 of file TOPTW.java.

6.6.3.10 getMaxTimePerRoute()

```
double top.TOPTW.getMaxTimePerRoute ( )
```

Definition at line 92 of file TOPTW.java.

6.6.3.11 getNodes()

```
int top.TOPTW.getNodes ( )
```

Definition at line 124 of file TOPTW.java.

6.6.3.12 getPOIs()

```
int top.TOPTW.getPOIs ( )
```

Definition at line 108 of file TOPTW.java.

6.6.3.13 getReadyTime()

```
double top.TOPTW.getReadyTime (
            int index )
```

Definition at line 163 of file TOPTW.java.

6.6.3.14 getScore() [1/2]

```
double[] top.TOPTW.getScore ( )
```

Definition at line 155 of file TOPTW.java.

6.6.3.15 getScore() [2/2]

```
double top.TOPTW.getScore (
            int index )
```

Definition at line 150 of file TOPTW.java.

6.6.3.16 getServiceTime()

```
double top.TOPTW.getServiceTime (
            int index )
```

Definition at line 181 of file TOPTW.java.

6.6.3.17 getTime()

```
double top.TOPTW.getTime (
            int i,
            int j )
```

Definition at line 118 of file TOPTW.java.

6.6.3.18 getVehicles()

```
int top.TOPTW.getVehicles ( )
```

Definition at line 190 of file TOPTW.java.

6.6.3.19 getX()

```
double top.TOPTW.getX (
            int index )
```

Definition at line 132 of file TOPTW.java.

6.6.3.20 getY()

```
double top.TOPTW.getY (
            int index )
```

Definition at line 141 of file TOPTW.java.

6.6.3.21 isDepot()

```
boolean top.TOPTW.isDepot (
            int a )
```

Definition at line 41 of file TOPTW.java.

6.6.3.22 setDueTime()

```
void top.TOPTW.setDueTime (
            int index,
            double dueTime )
```

Definition at line 177 of file TOPTW.java.

6.6.3.23 setMaxRoutes()

```
void top.TOPTW.setMaxRoutes (
            double maxRoutes )
```

Definition at line 104 of file TOPTW.java.

6.6.3.24 setMaxTimePerRoute()

```
void top.TOPTW.setMaxTimePerRoute (
            double maxTimePerRoute )
```

Definition at line 96 of file TOPTW.java.

6.6.3.25 setNodes()

```
void top.TOPTW.setNodes (
            int nodes )
```

Definition at line 128 of file TOPTW.java.

6.6.3.26 setReadyTime()

```
void top.TOPTW.setReadyTime (
            int index,
            double readyTime )
```

Definition at line 168 of file TOPTW.java.

6.6.3.27 setScore()

```
void top.TOPTW.setScore (
            int index,
            double score )
```

Definition at line 159 of file TOPTW.java.

6.6.3.28 setServiceTime()

```
void top.TOPTW.setServiceTime (
            int index,
            double serviceTime )
```

Definition at line 186 of file TOPTW.java.

6.6.3.29 setX()

```
void top.TOPTW.setX (
            int index,
            double x )
```

Definition at line 137 of file TOPTW.java.

6.6.3.30 setY()

```
void top.TOPTW.setY (
            int index,
            double y )
```

Definition at line 146 of file TOPTW.java.

6.6.3.31 toString()

```
String top.TOPTW.toString ( )
```

Definition at line 195 of file TOPTW.java.

The documentation for this class was generated from the following file:

- src/main/java/top/TOPTW.java

6.7 top.TOPTWEvaluator Class Reference

Public Member Functions

- void evaluate (TOPTWSolution solution)

Static Public Attributes

- static double NO_EVALUATED = -1.0

6.7.1 Detailed Description

Definition at line 3 of file TOPTWEvaluator.java.

6.7.2 Member Function Documentation

6.7.2.1 evaluate()

```
void top.TOPTWEvaluator.evaluate (
            TOPTWSolution solution )
```

Definition at line 6 of file TOPTWEvaluator.java.

6.7.3 Member Data Documentation

6.7.3.1 NO_EVALUATED

```
double top.TOPTWEvaluator.NO_EVALUATED = -1.0  [static]
```

Definition at line 4 of file TOPTWEvaluator.java.

The documentation for this class was generated from the following file:

- src/main/java/top/TOPTWEvaluator.java

6.8 top.TOPTWGRASP Class Reference

Public Member Functions

- TOPTWGRASP (TOPTWSolution sol)
- void GRASP (int maxIterations, int maxSizeRCL)
- int aleatorySelectionRCL (int maxTRCL)
- int fuzzySelectionBestFDRCL (ArrayList< double[ ] > rcl)
- int fuzzySelectionAlphaCutRCL (ArrayList< double[ ] > rcl, double alpha)
- void computeGreedySolution (int maxSizeRCL)
- void updateSolution (double[ ] candidateSelected, ArrayList< ArrayList< Double > > departureTimes)
- ArrayList< double[ ] > comprehensiveEvaluation (ArrayList< Integer > customers, ArrayList< ArrayList< Double > > departureTimes)
- TOPTWSolution getSolution ()
- void setSolution (TOPTWSolution solution)
- int getSolutionTime ()
- void setSolutionTime (int solutionTime)
- double getMaxScore ()

Static Public Attributes

- static double NO_EVALUATED = -1.0

6.8.1 Detailed Description

Definition at line 8 of file TOPTWGRASP.java.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 TOPTWGRASP()

```
top.TOPTWGRASP.TOPTWGRASP (
            TOPTWSolution sol )
```

Definition at line 14 of file TOPTWGRASP.java.

6.8.3 Member Function Documentation

6.8.3.1 aleatorySelectionRCL()

```
int top.TOPTWGRASP.aleatorySelectionRCL (
            int maxTRCL )
```

Definition at line 70 of file TOPTWGRASP.java.

6.8.3.2 comprehensiveEvaluation()

```
ArrayList< double[] > top.TOPTWGRASP.comprehensiveEvaluation (
            ArrayList< Integer > customers,
            ArrayList< ArrayList< Double > > departureTimes )
```

Definition at line 228 of file TOPTWGRASP.java.

6.8.3.3 computeGreedySolution()

```
void top.TOPTWGRASP.computeGreedySolution (
            int maxSizeRCL )
```

Definition at line 116 of file TOPTWGRASP.java.

6.8.3.4 fuzzySelectionAlphaCutRCL()

```
int top.TOPTWGRASP.fuzzySelectionAlphaCutRCL (
            ArrayList< double[] > rcl,
            double alpha )
```

Definition at line 95 of file TOPTWGRASP.java.

6.8.3.5 fuzzySelectionBestFDRCL()

```
int top.TOPTWGRASP.fuzzySelectionBestFDRCL (
            ArrayList< double[] > rcl )
```

Definition at line 78 of file TOPTWGRASP.java.

6.8.3.6 getMaxScore()

```
double top.TOPTWGRASP.getMaxScore ( )
```

Definition at line 329 of file TOPTWGRASP.java.

6.8.3.7 getSolution()

```
TOPTWSolution top.TOPTWGRASP.getSolution ( )
```

Definition at line 313 of file TOPTWGRASP.java.

6.8.3.8 getSolutionTime()

```
int top.TOPTWGRASP.getSolutionTime ( )
```

Definition at line 321 of file TOPTWGRASP.java.

6.8.3.9 GRASP()

```
void top.TOPTWGRASP.GRASP (
            int maxIterations,
            int maxSizeRCL )
```

Definition at line 41 of file TOPTWGRASP.java.

6.8.3.10 setSolution()

```
void top.TOPTWGRASP.setSolution (
            TOPTWSolution solution )
```

Definition at line 317 of file TOPTWGRASP.java.

6.8.3.11 setSolutionTime()

```
void top.TOPTWGRASP.setSolutionTime (
            int solutionTime )
```

Definition at line 325 of file TOPTWGRASP.java.

6.8.3.12 updateSolution()

```
void top.TOPTWGRASP.updateSolution (
            double[] candidateSelected,
            ArrayList< ArrayList< Double > > departureTimes )
```

Definition at line 197 of file TOPTWGRASP.java.

6.8.4 Member Data Documentation

6.8.4.1 NO_EVALUATED

```
double top.TOPTWGRASP.NO_EVALUATED = -1.0  [static]
```

Definition at line 9 of file TOPTWGRASP.java.

The documentation for this class was generated from the following file:

- src/main/java/top/TOPTWGRASP.java

6.9 top.TOPTWReader Class Reference

Static Public Member Functions

- static TOPTW readProblem (String filePath)

6.9.1 Detailed Description

Definition at line 10 of file TOPTWReader.java.

6.9.2 Member Function Documentation

6.9.2.1 readProblem()

```
static TOPTW top.TOPTWReader.readProblem (
            String filePath )  [static]
```

Definition at line 12 of file TOPTWReader.java.

The documentation for this class was generated from the following file:

- src/main/java/top/TOPTWReader.java

6.10 top.TOPTWRoute Class Reference

Public Member Functions

- int getPredeccesor ()
- int getSuccesor ()
- int getId ()
- void setPredeccesor (int pre)
- void setSuccesor (int suc)
- void setId (int id)

6.10.1 Detailed Description

Definition at line 3 of file TOPTWRoute.java.

6.10.2 Member Function Documentation

6.10.2.1 getId()

```
int top.TOPTWRoute.getId ( )
```

Definition at line 26 of file TOPTWRoute.java.

6.10.2.2 getPredeccesor()

```
int top.TOPTWRoute.getPredeccesor ( )
```

Definition at line 18 of file TOPTWRoute.java.

6.10.2.3 getSuccesor()

```
int top.TOPTWRoute.getSuccesor ( )
```

Definition at line 22 of file TOPTWRoute.java.

6.10.2.4 setId()

```
void top.TOPTWRoute.setId (
            int id )
```

Definition at line 38 of file TOPTWRoute.java.

6.10.2.5 setPredeccesor()

```
void top.TOPTWRoute.setPredeccesor (
            int pre )
```

Definition at line 30 of file TOPTWRoute.java.

6.10.2.6 setSuccesor()

```
void top.TOPTWRoute.setSuccesor (
            int suc )
```

Definition at line 34 of file TOPTWRoute.java.

The documentation for this class was generated from the following file:

- src/main/java/top/TOPTWRoute.java

6.11 top.TOPTWSolution Class Reference

Public Member Functions

- TOPTWSolution (TOPTW problem)
- void initSolution ()
- boolean isDepot (int c)
- boolean equals (TOPTWSolution otherSolution)
- int getAvailableVehicles ()
- int getCreatedRoutes ()
- double getDistance (int x, int y)
- void setAvailableVehicles (int availableVehicles)
- int getPredecessor (int customer)
- int[ ] getPredecessors ()
- TOPTW getProblem ()
- double getObjectiveFunctionValue ()
- int getPositionInRoute (int customer)
- int getSuccessor (int customer)
- int[ ] getSuccessors ()
- int getIndexRoute (int index)
- double getWaitingTime (int customer)
- void setObjectiveFunctionValue (double objectiveFunctionValue)
- void setPositionInRoute (int customer, int position)
- void setPredecessor (int customer, int predecessor)
- void setSuccessor (int customer, int succesor)
- void setWaitingTime (int customer, int waitingTime)
- String getInfoSolution ()
- double evaluateFitness ()
- int addRoute ()
- double printSolution ()

Static Public Attributes

- static final int NO_INITIALIZED = -1

6.11.1 Detailed Description

Definition at line 7 of file TOPTWSolution.java.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 TOPTWSolution()

```
top.TOPTWSolution.TOPTWSolution (
            TOPTW problem )
```

Definition at line 19 of file TOPTWSolution.java.

6.11.3 Member Function Documentation

6.11.3.1 addRoute()

```
int top.TOPTWSolution.addRoute ( )
```

Definition at line 218 of file TOPTWSolution.java.

6.11.3.2 equals()

```
boolean top.TOPTWSolution.equals (
            TOPTWSolution otherSolution )
```

Definition at line 56 of file TOPTWSolution.java.

6.11.3.3 evaluateFitness()

```
double top.TOPTWSolution.evaluateFitness ( )
```

Definition at line 201 of file TOPTWSolution.java.

6.11.3.4 getAvailableVehicles()

```
int top.TOPTWSolution.getAvailableVehicles ( )
```

Definition at line 65 of file TOPTWSolution.java.

6.11.3.5 getCreatedRoutes()

```
int top.TOPTWSolution.getCreatedRoutes ( )
```

Definition at line 69 of file TOPTWSolution.java.

6.11.3.6 getDistance()

```
double top.TOPTWSolution.getDistance (
            int x,
            int y )
```

Definition at line 73 of file TOPTWSolution.java.

6.11.3.7 getIndexRoute()

```
int top.TOPTWSolution.getIndexRoute (
            int index )
```

Definition at line 109 of file TOPTWSolution.java.

6.11.3.8 getInfoSolution()

```
String top.TOPTWSolution.getInfoSolution ( )
```

Definition at line 137 of file TOPTWSolution.java.

6.11.3.9 getObjectiveFunctionValue()

```
double top.TOPTWSolution.getObjectiveFunctionValue ( )
```

Definition at line 93 of file TOPTWSolution.java.

6.11.3.10 getPositionInRoute()

```
int top.TOPTWSolution.getPositionInRoute (
            int customer )
```

Definition at line 97 of file TOPTWSolution.java.

6.11.3.11 getPredecessor()

```
int top.TOPTWSolution.getPredecessor (
            int customer )
```

Definition at line 81 of file TOPTWSolution.java.

6.11.3.12 getPredecessors()

```
int[] top.TOPTWSolution.getPredecessors ( )
```

Definition at line 85 of file TOPTWSolution.java.

6.11.3.13 getProblem()

```
TOPTW top.TOPTWSolution.getProblem ( )
```

Definition at line 89 of file TOPTWSolution.java.

6.11.3.14 getSuccessor()

```
int top.TOPTWSolution.getSuccessor (
            int customer )
```

Definition at line 101 of file TOPTWSolution.java.

6.11.3.15 getSuccessors()

```
int[] top.TOPTWSolution.getSuccessors ( )
```

Definition at line 105 of file TOPTWSolution.java.

6.11.3.16 getWaitingTime()

```
double top.TOPTWSolution.getWaitingTime (
            int customer )
```

Definition at line 113 of file TOPTWSolution.java.

6.11.3.17 initSolution()

```
void top.TOPTWSolution.initSolution ( )
```

Definition at line 34 of file TOPTWSolution.java.

6.11.3.18 isDepot()

```
boolean top.TOPTWSolution.isDepot (
            int c )
```

Definition at line 47 of file TOPTWSolution.java.

6.11.3.19 printSolution()

```
double top.TOPTWSolution.printSolution ( )
```

Definition at line 237 of file TOPTWSolution.java.

6.11.3.20 setAvailableVehicles()

```
void top.TOPTWSolution.setAvailableVehicles (
            int availableVehicles )
```

Definition at line 77 of file TOPTWSolution.java.

6.11.3.21 setObjectiveFunctionValue()

```
void top.TOPTWSolution.setObjectiveFunctionValue (
            double objectiveFunctionValue )
```

Definition at line 117 of file TOPTWSolution.java.

6.11.3.22 setPositionInRoute()

```
void top.TOPTWSolution.setPositionInRoute (
            int customer,
            int position )
```

Definition at line 121 of file TOPTWSolution.java.

6.11.3.23 setPredecessor()

```
void top.TOPTWSolution.setPredecessor (
            int customer,
            int predecessor )
```

Definition at line 125 of file TOPTWSolution.java.

6.11.3.24 setSuccessor()

```
void top.TOPTWSolution.setSuccessor (
            int customer,
            int succesor )
```

Definition at line 129 of file TOPTWSolution.java.

6.11.3.25 setWaitingTime()

```
void top.TOPTWSolution.setWaitingTime (
            int customer,
            int waitingTime )
```

Definition at line 133 of file TOPTWSolution.java.

6.11.4 Member Data Documentation

6.11.4.1 NO_INITIALIZED

```
final int top.TOPTWSolution.NO_INITIALIZED = -1  [static]
```

Definition at line 8 of file TOPTWSolution.java.

The documentation for this class was generated from the following file:

- src/main/java/top/TOPTWSolution.java

Chapter 7

File Documentation

7.1 src/main/java/es/ull/esit/utilities/BellmanFord.java File Reference

Classes

- class es.ull.esit.utilities.BellmanFord

Packages

- package es.ull.esit.utilities

7.2 BellmanFord.java

Go to the documentation of this file.

```
00001 package es.ull.esit.utilities;
00002
00003 import java.util.ArrayList;
00004
00005 public class BellmanFord {
00006
00010     private static final int INFINITY = 999999;
00014     private final int[][] distanceMatrix;
00018     private ArrayList<Integer> edges1 = null;
00022     private ArrayList<Integer> edges2 = null;
00026     private final int nodes;
00030     private final ArrayList<Integer> path;
00034     private int[] distances = null;
00038     private int value;
00039
00046     public BellmanFord(int[][] distanceMatrix, int nodes, ArrayList<Integer> path) {
00047         this.distanceMatrix = distanceMatrix;
00048         this.nodes = nodes;
00049         this.path = path;
00050         this.calculateEdges();
00051         this.value = BellmanFord.INFINITY;
00052     }
00053
00057     private void calculateEdges() {
00058         this.edges1 = new ArrayList<>();
00059         this.edges2 = new ArrayList<>();
00060         for (int i = 0; i < this.nodes; i++) {
00061             for (int j = 0; j < this.nodes; j++) {
00062                 if (this.distanceMatrix[i][j] != Integer.MAX_VALUE) {
00063                     this.edges1.add(i);
00064                     this.edges2.add(j);
00065                 }
00066             }
```

```
00067         }
00068     }
00069
00074     public int[] getDistances() {
00075         return this.distances;
00076     }
00077
00082     public int getValue() {
00083         return this.value;
00084     }
00085
00089     public void solve() {
00090         int numEdges = this.edges1.size();
00091         int[] predecessor = new int[this.nodes];
00092         this.distances = new int[this.nodes];
00093         for (int i = 0; i < this.nodes; i++) {
00094             this.distances[i] = BellmanFord.INFINITY;
00095             predecessor[i] = -1;
00096         }
00097         this.distances[0] = 0;
00098         for (int i = 0; i < (this.nodes - 1); i++) {
00099             for (int j = 0; j < numEdges; j++) {
00100                 int u = this.edges1.get(j);
00101                 int v = this.edges2.get(j);
00102                 if (this.distances[v] > this.distances[u] + this.distanceMatrix[u][v]) {
00103                     this.distances[v] = this.distances[u] + this.distanceMatrix[u][v];
00104                     predecessor[v] = u;
00105                 }
00106             }
00107         }
00108         this.path.add(this.nodes - 1);
00109         int pred = predecessor[this.nodes - 1];
00110         while (pred != -1) {
00111             this.path.add(pred);
00112             pred = predecessor[pred];
00113         }
00114         this.value = -this.distances[this.nodes - 1];
00115     }
00116 }
```

7.3 src/main/java/es/ull/esit/utilities/ExpositoUtilities.java File Reference

Classes

- class es.ull.esit.utilities.ExpositoUtilities

Packages

- package es.ull.esit.utilities

7.4 ExpositoUtilities.java

Go to the documentation of this file.

```
00001 package es.ull.esit.utilities;
00002
00003 import java.io.BufferedReader;
00004 import java.io.BufferedWriter;
00005 import java.io.FileReader;
00006 import java.io.FileWriter;
00007 import java.io.IOException;
00008 import java.text.DecimalFormat;
00009 import java.text.DecimalFormatSymbols;
00010 import java.util.ArrayList;
00011 import java.util.Arrays;
00012 import java.util.HashSet;
00013 import java.util.Iterator;
00014 import java.util.logging.Level;
00015 import java.util.logging.Logger;
00016
00017 public class ExpositoUtilities {
```

```
00018
00019     public static final int DEFAULT_COLUMN_WIDTH = 10;
00020     public static final int ALIGNMENT_LEFT = 1;
00021     public static final int ALIGNMENT_RIGHT = 2;
00022
00023     private static int getFirstAppearance(int[] vector, int element) {
00024         for (int i = 0; i < vector.length; i++) {
00025             if (vector[i] == element) {
00026                 return i;
00027             }
00028         }
00029         return -1;
00030     }
00031
00032     public static void printFile(String file) {
00033         BufferedReader reader = null;
00034         try {
00035             reader = new BufferedReader(new FileReader(file));
00036             String line = "";
00037             while ((line = reader.readLine()) != null) {
00038                 System.out.println(line);
00039             }
00040         } catch (Exception ex) {
00041             Logger.getLogger(ExpositoUtilities.class.getName()).log(Level.SEVERE, null, ex);
00042         } finally {
00043             try {
00044                 reader.close();
00045             } catch (IOException ex) {
00046                 Logger.getLogger(ExpositoUtilities.class.getName()).log(Level.SEVERE, null, ex);
00047             }
00048         }
00049     }
00050
00051     public static String simplifyString(String string) {
00052         string = string.replaceAll("\t", " ");
00053         for (int i = 0; i < 50; i++) {
00054             string = string.replaceAll("  ", " ");
00055         }
00056         string = string.trim();
00057         return string;
00058     }
00059
00060     public static double[][] multiplyMatrices(double a[][], double b[][]) {
00061         if (a.length == 0) {
00062             return new double[0][0];
00063         }
00064         if (a[0].length != b.length) {
00065             return null;
00066         }
00067         int n = a[0].length;
00068         int m = a.length;
00069         int p = b[0].length;
00070         double ans[][] = new double[m][p];
00071         for (int i = 0; i < m; i++) {
00072             for (int j = 0; j < p; j++) {
00073                 for (int k = 0; k < n; k++) {
00074                     ans[i][j] += a[i][k] * b[k][j];
00075                 }
00076             }
00077         }
00078         return ans;
00079     }
00080
00081     public static void writeTextToFile(String file, String text) throws IOException {
00082         BufferedWriter writer = new BufferedWriter(new FileWriter(file));
00083         writer.write(text);
00084         writer.flush();
00085         writer.close();
00086     }
00087
00088     public static String getFormat(String string) {
00089         if (!ExpositoUtilities.isInteger(string)) {
00090             if (ExpositoUtilities.isDouble(string)) {
00091                 double value = Double.parseDouble(string);
00092                 string = ExpositoUtilities.getFormat(value);
00093             }
00094         }
00095         return string;
00096     }
00097
00098     public static String getFormat(double value) {
00099         DecimalFormat decimalFormatter = new DecimalFormat("0.000");
00100         DecimalFormatSymbols symbols = new DecimalFormatSymbols();
00101         symbols.setDecimalSeparator('.');
00102         decimalFormatter.setDecimalFormatSymbols(symbols);
00103         return decimalFormatter.format(value);
00104     }
```

```
00105
00106     public static String getFormat(double value, int zeros) {
00107         String format = "0.";
00108         for (int i = 0; i < zeros; i++) {
00109             format += "0";
00110         }
00111         DecimalFormat decimalFormatter = new DecimalFormat(format);
00112         DecimalFormatSymbols symbols = new DecimalFormatSymbols();
00113         symbols.setDecimalSeparator('.');
00114         decimalFormatter.setDecimalFormatSymbols(symbols);
00115         return decimalFormatter.format(value);
00116     }
00117
00118     public static String getFormat(String string, int width) {
00119         return ExpositoUtilities.getFormat(string, width, ExpositoUtilities.ALIGNMENT_RIGHT);
00120     }
00121
00122     public static String getFormat(String string, int width, int alignment) {
00123         String format = "";
00124         if (alignment == ExpositoUtilities.ALIGNMENT_LEFT) {
00125             format = "%-" + width + "s";
00126         } else {
00127             format = "%" + 1 + "$" + width + "s";
00128         }
00129         DecimalFormatSymbols symbols = new DecimalFormatSymbols();
00130         symbols.setDecimalSeparator('.');
00131         String[] data = new String[]{string};
00132         return String.format(format, (Object[]) data);
00133     }
00134
00135     public static String getFormat(ArrayList<String> strings, int width) {
00136         String format = "";
00137         for (int i = 0; i < strings.size(); i++) {
00138             format += "%" + (i + 1) + "$" + width + "s";
00139         }
00140         String[] data = new String[strings.size()];
00141         for (int t = 0; t < strings.size(); t++) {
00142             data[t] = "" + ExpositoUtilities.getFormat(strings.get(t));
00143         }
00144         return String.format(format, (Object[]) data);
00145     }
00146
00147     public static String getFormat(ArrayList<Integer> strings) {
00148         String format = "";
00149         for (int i = 0; i < strings.size(); i++) {
00150             format += "%" + (i + 1) + "$" + DEFAULT_COLUMN_WIDTH + "s";
00151         }
00152         Integer[] data = new Integer[strings.size()];
00153         for (int t = 0; t < strings.size(); t++) {
00154             data[t] = strings.get(t);
00155         }
00156         return String.format(format, (Object[]) data);
00157     }
00158
00159     public static String getFormat(String[] strings, int width) {
00160         int[] alignment = new int[strings.length];
00161         Arrays.fill(alignment, ExpositoUtilities.ALIGNMENT_RIGHT);
00162         int[] widths = new int[strings.length];
00163         Arrays.fill(widths, width);
00164         return ExpositoUtilities.getFormat(strings, widths, alignment);
00165     }
00166
00167         public static String getFormat(String[][] matrixStrings, int width) {
00168         String result = "";
00169         for (int i = 0; i < matrixStrings.length; i++) {
00170             String[] strings = matrixStrings[i];
00171             int[] alignment = new int[strings.length];
00172             Arrays.fill(alignment, ExpositoUtilities.ALIGNMENT_RIGHT);
00173             int[] widths = new int[strings.length];
00174             Arrays.fill(widths, width);
00175             result += ExpositoUtilities.getFormat(strings, widths, alignment);
00176             if (i < (matrixStrings.length - 1)) {
00177                 result += "\n";
00178             }
00179         }
00180         return result;
00181     }
00182
00183     public static String getFormat(String[] strings) {
00184         int[] alignment = new int[strings.length];
00185         Arrays.fill(alignment, ExpositoUtilities.ALIGNMENT_RIGHT);
00186         int[] widths = new int[strings.length];
00187         Arrays.fill(widths, ExpositoUtilities.DEFAULT_COLUMN_WIDTH);
00188         return ExpositoUtilities.getFormat(strings, widths, alignment);
00189     }
00190
00191     public static String getFormat(String[] strings, int[] width) {
```

```
00192         int[] alignment = new int[strings.length];
00193         Arrays.fill(alignment, ExpositoUtilities.ALIGNMENT_RIGHT);
00194         return ExpositoUtilities.getFormat(strings, width, alignment);
00195     }
00196 
00197     public static String getFormat(String[] strings, int[] width, int[] alignment) {
00198         String format = "";
00199         for (int i = 0; i < strings.length; i++) {
00200             if (alignment[i] == ExpositoUtilities.ALIGNMENT_LEFT) {
00201                 format += "%" + (i + 1) + "$-" + width[i] + "s";
00202             } else {
00203                 format += "%" + (i + 1) + "$" + width[i] + "s";
00204             }
00205         }
00206         String[] data = new String[strings.length];
00207         for (int t = 0; t < strings.length; t++) {
00208             data[t] = "" + ExpositoUtilities.getFormat(strings[t]);
00209         }
00210         return String.format(format, (Object[]) data);
00211     }
00212 
00213     public static boolean isInteger(String str) {
00214         try {
00215             Integer.parseInt(str);
00216             return true;
00217         } catch (Exception e) {
00218         }
00219         return false;
00220     }
00221 
00222     public static boolean isDouble(String str) {
00223         try {
00224             Double.parseDouble(str);
00225             return true;
00226         } catch (Exception e) {
00227         }
00228         return false;
00229     }
00230 
00231     public static boolean isAcyclic(int[][] distanceMatrix) {
00232         int numRealTasks = distanceMatrix.length - 2;
00233         int node = 1;
00234         boolean acyclic = true;
00235         while (acyclic && node <= numRealTasks) {
00236             if (ExpositoUtilities.thereIsPath(distanceMatrix, node)) {
00237                 return false;
00238             }
00239             node++;
00240         }
00241         return true;
00242     }
00243 
00244     public static boolean thereIsPath(int[][] distanceMatrix, int node) {
00245         HashSet<Integer> visits = new HashSet<>();
00246         HashSet<Integer> noVisits = new HashSet<>();
00247         for (int i = 0; i < distanceMatrix.length; i++) {
00248             if (i != node) {
00249                 noVisits.add(i);
00250             }
00251         }
00252         visits.add(node);
00253         while (!visits.isEmpty()) {
00254             Iterator<Integer> it = visits.iterator();
00255             int toCheck = it.next();
00256             visits.remove(toCheck);
00257             for (int i = 0; i < distanceMatrix.length; i++) {
00258                 if (toCheck != i && distanceMatrix[toCheck][i] != Integer.MAX_VALUE) {
00259                     if (i == node) {
00260                         return true;
00261                     }
00262                     if (noVisits.contains(i)) {
00263                         noVisits.remove(i);
00264                         visits.add(i);
00265                     }
00266                 }
00267             }
00268         }
00269         return false;
00270     }
00271 }
```

7.5 src/main/java/es/ull/esit/utilities/PowerSet.java File Reference

Classes

- class es.ull.esit.utilities.PowerSet< E >

Packages

- package es.ull.esit.utilities

7.6 PowerSet.java

Go to the documentation of this file.

```
00001 package es.ull.esit.utilities;
00002
00003 import java.util.BitSet;
00004 import java.util.Iterator;
00005 import java.util.Set;
00006 import java.util.TreeSet;
00007
00008 // Sirve para calcular todos los subconjuntos de un conjunto dado
00009 public class PowerSet<E> implements Iterator<Set<E>>, Iterable<Set<E>> {
00010
00011     private E[] arr = null;
00012     private BitSet bset = null;
00013
00014     @SuppressWarnings("unchecked")
00015     public PowerSet(Set<E> set) {
00016         this.arr = (E[]) set.toArray();
00017         this.bset = new BitSet(this.arr.length + 1);
00018     }
00019
00020     @Override
00021     public boolean hasNext() {
00022         return !this.bset.get(this.arr.length);
00023     }
00024
00025     @Override
00026     public Set<E> next() {
00027         Set<E> returnSet = new TreeSet<>();
00028         for (int i = 0; i < this.arr.length; i++) {
00029             if (this.bset.get(i)) {
00030                 returnSet.add(this.arr[i]);
00031             }
00032         }
00033         for (int i = 0; i < this.bset.size(); i++) {
00034             if (!this.bset.get(i)) {
00035                 this.bset.set(i);
00036                 break;
00037             } else {
00038                 this.bset.clear(i);
00039             }
00040         }
00041         return returnSet;
00042     }
00043
00044     @Override
00045     public void remove() {
00046         throw new UnsupportedOperationException("Not Supported!");
00047     }
00048
00049     @Override
00050     public Iterator<Set<E>> iterator() {
00051         return this;
00052     }
00053 }
```

7.7 src/main/java/es/ull/esit/utils/Pair.java File Reference

Classes

- class es.ull.esit.utils.Pair< F, S >

Packages

- package es.ull.esit.utils

7.8 Pair.java

Go to the documentation of this file.

```
00001 package es.ull.esit.utils;
00002 import java.util.Objects;
00003
00004 public class Pair<F, S> {
00005     public final F first;
00006     public final S second;
00007
00008     public Pair(F first, S second) {
00009         this.first = first;
00010         this.second = second;
00011     }
00012
00013     @Override
00014     public boolean equals(Object o) {
00015         if (!(o instanceof Pair)) {
00016             return false;
00017         }
00018         Pair<?, ?> p = (Pair<?, ?>) o;
00019         return Objects.equals(p.first, first) && Objects.equals(p.second, second);
00020     }
00021
00022     @Override
00023     public int hashCode() {
00024         return (first == null ? 0 : first.hashCode()) ^ (second == null ? 0 : second.hashCode());
00025     }
00026
00027     public static <A, B> Pair <A, B> create(A a, B b) {
00028         return new Pair<A, B>(a, b);
00029     }
00030 }
```

7.9 src/main/java/top/mainTOPTW.java File Reference

Classes

- class top.mainTOPTW

Packages

- package top

7.10 mainTOPTW.java

Go to the documentation of this file.

```
00001 package top;
00002
00003 public class mainTOPTW {
00004
00005     public static void main(String[] args) {
00006
00007         String[] instances = new String[29];
00008
00009         instances[0] = "c101.txt"; instances[3] = "c104.txt"; instances[6] = "c107.txt";
00010         instances[1] = "c102.txt"; instances[4] = "c105.txt"; instances[7] = "c108.txt";
00011         instances[2] = "c103.txt"; instances[5] = "c106.txt"; instances[8] = "c109.txt";
00012
```

```
00013         instances[9] = "r101.txt";  instances[12] = "r104.txt"; instances[15] = "r107.txt";
00014         instances[10] = "r102.txt"; instances[13] = "r105.txt"; instances[16] = "r108.txt";
00015         instances[11] = "r103.txt"; instances[14] = "r106.txt"; instances[17] = "r109.txt";
00016         instances[18] = "r110.txt"; instances[19] = "r111.txt"; instances[20] = "r112.txt";
00017
00018         instances[21] = "rc101.txt"; instances[24] = "rc104.txt"; instances[27] = "rc107.txt";
00019         instances[22] = "rc102.txt"; instances[25] = "rc105.txt"; instances[28] = "rc108.txt";
00020         instances[23] = "rc103.txt"; instances[26] = "rc106.txt";
00021
00022         for(int i = 0; i < instances.length; i++) {
00023             String INSTANCE = "Instances/TOPTW/"+instances[i];
00024             TOPTW problem = TOPTWReader.readProblem(INSTANCE);
00025             TOPTWSolution solution = new TOPTWSolution(problem);
00026             TOPTWGRASP grasp = new TOPTWGRASP(solution);
00027
00028             System.out.println(" --> Instance: "+instances[i]);
00029             grasp.GRASP(10000, 3);
00030             grasp.GRASP(10000, 5);
00031             grasp.GRASP(10000, 7);
00032             System.out.println("");
00033         }
00034     }
00035
00036 }
```

7.11 src/main/java/top/TOPTW.java File Reference

Classes

- class top.TOPTW

Packages

- package top

7.12 TOPTW.java

Go to the documentation of this file.

```
00001 package top;
00002
00003 import java.util.ArrayList;
00004 import java.util.Arrays;
00005
00006 import es.ull.esit.utilities.ExpositoUtilities;
00007
00008 public class TOPTW {
00009     private int nodes;
00010     private double[] x;
00011     private double[] y;
00012     private double[] score;
00013     private double[] readyTime;
00014     private double[] dueTime;
00015     private double[] serviceTime;
00016     private int vehicles;
00017     private int depots;
00018     private double maxTimePerRoute;
00019     private double maxRoutes;
00020     private double[][] distanceMatrix;
00021
00022     public TOPTW(int nodes, int routes) {
00023         this.nodes = nodes;
00024         this.depots = 0;
00025         this.x = new double[this.nodes + 1];
00026         this.y = new double[this.nodes + 1];
00027         this.score = new double[this.nodes + 1];
00028         this.readyTime = new double[this.nodes + 1];
00029         this.dueTime = new double[this.nodes + 1];
00030         this.serviceTime = new double[this.nodes + 1];
00031         this.distanceMatrix = new double[this.nodes + 1][this.nodes + 1];
00032         for (int i = 0; i < this.nodes + 1; i++) {
```

```
00033             for (int j = 0; j < this.nodes + 1; j++) {
00034                 this.distanceMatrix[i][j] = 0.0;
00035             }
00036         }
00037         this.maxRoutes = routes;
00038         this.vehicles = routes;
00039     }
00040 
00041     public boolean isDepot(int a) {
00042         if(a > this.nodes) {
00043             return true;
00044         }
00045         return false;
00046     }
00047 
00048     public double getDistance(int[] route) {
00049         double distance = 0.0;
00050         for (int i = 0; i < route.length - 1; i++) {
00051             int node1 = route[i];
00052             int node2 = route[i + 1];
00053             distance += this.getDistance(node1, node2);
00054         }
00055         return distance;
00056     }
00057 
00058     public double getDistance(ArrayList<Integer> route) {
00059         double distance = 0.0;
00060         for (int i = 0; i < route.size() - 1; i++) {
00061             int node1 = route.get(i);
00062             int node2 = route.get(i + 1);
00063             distance += this.getDistance(node1, node2);
00064         }
00065         return distance;
00066     }
00067 
00068     public double getDistance(ArrayList<Integer>[] routes) {
00069         double distance = 0.0;
00070         for (ArrayList<Integer> route : routes) {
00071             distance += this.getDistance(route);
00072         }
00073         return distance;
00074     }
00075 
00076 
00077     public void calculateDistanceMatrix() {
00078         for (int i = 0; i < this.nodes + 1; i++) {
00079             for (int j = 0; j < this.nodes + 1; j++) {
00080                 if (i != j) {
00081                     double diffXs = this.x[i] - this.x[j];
00082                     double diffYs = this.y[i] - this.y[j];
00083                     this.distanceMatrix[i][j] = Math.sqrt(diffXs * diffXs + diffYs * diffYs);
00084                     this.distanceMatrix[j][i] = this.distanceMatrix[i][j];
00085                 } else {
00086                     this.distanceMatrix[i][j] = 0.0;
00087                 }
00088             }
00089         }
00090     }
00091 
00092     public double getMaxTimePerRoute() {
00093         return maxTimePerRoute;
00094     }
00095 
00096     public void setMaxTimePerRoute(double maxTimePerRoute) {
00097         this.maxTimePerRoute = maxTimePerRoute;
00098     }
00099 
00100     public double getMaxRoutes() {
00101         return maxRoutes;
00102     }
00103 
00104     public void setMaxRoutes(double maxRoutes) {
00105         this.maxRoutes = maxRoutes;
00106     }
00107 
00108     public int getPOIs() {
00109         return this.nodes;
00110     }
00111 
00112     public double getDistance(int i, int j) {
00113         if(this.isDepot(i)) { i=0; }
00114         if(this.isDepot(j)) { j=0; }
00115         return this.distanceMatrix[i][j];
00116     }
00117 
00118     public double getTime(int i, int j) {
00119         if(this.isDepot(i)) { i=0; }
```

```
00120         if(this.isDepot(j)) { j=0; }
00121         return this.distanceMatrix[i][j];
00122     }
00123 
00124     public int getNodes() {
00125         return this.nodes;
00126     }
00127 
00128     public void setNodes(int nodes) {
00129         this.nodes = nodes;
00130     }
00131 
00132     public double getX(int index) {
00133         if(this.isDepot(index)) { index=0; }
00134         return this.x[index];
00135     }
00136 
00137     public void setX(int index, double x) {
00138         this.x[index] = x;
00139     }
00140 
00141     public double getY(int index) {
00142         if(this.isDepot(index)) { index=0; }
00143         return this.y[index];
00144     }
00145 
00146     public void setY(int index, double y) {
00147         this.y[index] = y;
00148     }
00149 
00150     public double getScore(int index) {
00151         if(this.isDepot(index)) { index=0; }
00152         return this.score[index];
00153     }
00154 
00155     public double[] getScore() {
00156         return this.score;
00157     }
00158 
00159     public void setScore(int index, double score) {
00160         this.score[index] = score;
00161     }
00162 
00163     public double getReadyTime(int index) {
00164         if(this.isDepot(index)) { index=0; }
00165         return this.readyTime[index];
00166     }
00167 
00168     public void setReadyTime(int index, double readyTime) {
00169         this.readyTime[index] = readyTime;
00170     }
00171 
00172     public double getDueTime(int index) {
00173         if(this.isDepot(index)) { index=0; }
00174         return this.dueTime[index];
00175     }
00176 
00177     public void setDueTime(int index, double dueTime) {
00178         this.dueTime[index] = dueTime;
00179     }
00180 
00181     public double getServiceTime(int index) {
00182         if(this.isDepot(index)) { index=0; }
00183         return this.serviceTime[index];
00184     }
00185 
00186     public void setServiceTime(int index, double serviceTime) {
00187         this.serviceTime[index] = serviceTime;
00188     }
00189 
00190     public int getVehicles() {
00191         return this.vehicles;
00192     }
00193 
00194     @Override
00195     public String toString() {
00196         final int COLUMN_WIDTH = 15;
00197         String text = "Nodes: " + this.nodes + "\n";
00198         String[] strings = new String[]{"CUST NO.", "XCOORD.", "YCOORD.", "SCORE", "READY TIME", "DUE
    DATE", "SERVICE TIME"};
00199         int[] width = new int[strings.length];
00200         Arrays.fill(width, COLUMN_WIDTH);
00201         text += ExpositoUtilities.getFormat(strings, width) + "\n";
00202         for (int i = 0; i < this.nodes; i++) {
00203             strings = new String[strings.length];
00204             int index = 0;
00205             //strings[index++] = Integer.toString("" + i);
```

```
00206             strings[index++] = Integer.toString(i);
00207             strings[index++] = "" + this.x[i];
00208             strings[index++] = "" + this.y[i];
00209             strings[index++] = "" + this.score[i];
00210             strings[index++] = "" + this.readyTime[i];
00211             strings[index++] = "" + this.dueTime[i];
00212             strings[index++] = "" + this.serviceTime[i];
00213             text += ExpositoUtilities.getFormat(strings, width);
00214             text += "\n";
00215         }
00216         text += "Vehicles: " + this.vehicles + "\n";
00217         strings = new String[]{"VEHICLE", "CAPACITY"};
00218         width = new int[strings.length];
00219         Arrays.fill(width, COLUMN_WIDTH);
00220         text += ExpositoUtilities.getFormat(strings, width) + "\n";
00221         return text;
00222     }
00223
00224     public int addNode() {
00225         this.nodes++;
00226         return this.nodes;
00227     }
00228
00229     public int addNodeDepot() {
00230         this.depots++;
00231         return this.depots;
00232     }
00233 }
```

7.13 src/main/java/top/TOPTWEvaluator.java File Reference

Classes

- class top.TOPTWEvaluator

Packages

- package top

7.14 TOPTWEvaluator.java

Go to the documentation of this file.

```
00001 package top;
00002
00003 public class TOPTWEvaluator {
00004     public static double NO_EVALUATED = -1.0;
00005
00006     public void evaluate(TOPTWSolution solution) {
00007         /*CumulativeCVRP problem = solution.getProblem();
00008         double objectiveFunctionValue = 0.0;
00009         for (int i = 0; i < solution.getIndexDepot().size(); i++) {
00010             double cumulative = 0;
00011             int depot = solution.getAnIndexDepot(i);
00012             int actual = depot;
00013             actual = solution.getSuccessor(actual);
00014             cumulative += problem.getDistanceMatrix(0, actual);
00015             objectiveFunctionValue += problem.getDistanceMatrix(0, actual);
00016             System.out.println("Desde " + 0 + " a " + actual + " = " + cumulative);
00017             while (actual != depot) {
00018                 int ant = actual;
00019                 actual = solution.getSuccessor(actual);
00020                 if (actual != depot) {
00021                     cumulative += problem.getDistanceMatrix(ant, actual);
00022                     objectiveFunctionValue += cumulative;
00023                     System.out.println("Desde " + ant + " a " + actual + " = " + cumulative);
00024                 } else {
00025                     cumulative += problem.getDistanceMatrix(ant, 0);
00026                     objectiveFunctionValue += cumulative;
00027                     System.out.println("Desde " + ant + " a " + 0 + " = " + cumulative);
00028                 }
00029             }
00030             System.out.println("");
00031         }
00032         solution.setObjectiveFunctionValue(objectiveFunctionValue);*/
00033     }
00034 }
```

7.15 src/main/java/top/TOPTWGRASP.java File Reference

Classes

- class top.TOPTWGRASP

Packages

- package top

7.16 TOPTWGRASP.java

Go to the documentation of this file.

```
00001 package top;
00002
00003 import java.util.ArrayList;
00004 import java.util.Collections;
00005 import java.util.Comparator;
00006 import java.security.SecureRandom;
00007
00008 public class TOPTWGRASP {
00009  public static double NO_EVALUATED = -1.0;
00010
00011     private TOPTWSolution solution;
00012     private int solutionTime;
00013
00014     public TOPTWGRASP(TOPTWSolution sol){
00015         this.solution = sol;
00016         this.solutionTime = 0;
00017     }
00018
00019     /*procedure GRASP(Max Iterations,Seed)
00020         1 Read Input();
00021         2 for k = 1, . . . , Max Iterations do
00022             3 Solution ← Greedy Randomized Construction(Seed);
00023             4 Solution ← Local Search(Solution);
00024             5 Update Solution(Solution,Best Solution);
00025         6 end;
00026         7 return Best Solution;
00027     end GRASP*/
00028
00029     /*procedure Greedy Randomized Construction(Seed)
00030         Solution ← ;
00031         Evaluate the incremental costs of the candidate elements;
00032         while Solution is not a complete solution do
00033             Build the restricted candidate list (RCL);
00034             Select an element s from the RCL at random;
00035             Solution ← Solution  {s};
00036             Reevaluate the incremental costs;
00037         end;
00038         return Solution;
00039     end Greedy Randomized Construction.*/
00040
00041     public void GRASP(int maxIterations, int maxSizeRCL) {
00042         double averageFitness = 0.0;
00043         double bestSolution = 0.0;
00044         for(int i = 0; i < maxIterations; i++) {
00045
00046             this.computeGreedySolution(maxSizeRCL);
00047
00048             // IMPRIMIR SOLUCION
00049             double fitness = this.solution.evaluateFitness();
00050             System.out.println(this.solution.getInfoSolution());
00051             //System.out.println("Press Any Key To Continue...");
00052             //new java.util.Scanner(System.in).nextLine();
00053             averageFitness += fitness;
00054             if(bestSolution < fitness) {
00055                 bestSolution = fitness;
00056             }
00057             //double fitness = this.solution.printSolution();
00058
00059             /******
00060             *
00061             * BÚSQUEDA LOCAL
```

```
00062             *
00063             */
00064         }
00065         averageFitness = averageFitness/maxIterations;
00066         System.out.println(" --> MEDIA: "+averageFitness);
00067         System.out.println(" --> MEJOR SOLUCION: "+bestSolution);
00068     }
00069
00070     public int aleatorySelectionRCL(int maxTRCL) {
00071        SecureRandom r = new SecureRandom();
00072        int low = 0;
00073        int high = maxTRCL;
00074        int posSelected = r.nextInt(high-low) + low;
00075        return posSelected;
00076     }
00077
00078     public int fuzzySelectionBestFDRCL(ArrayList< double[] > rcl) {
00079         double[] membershipFunction = new double[rcl.size()];
00080         double maxSc = this.getMaxScore();
00081         for(int j=0; j < rcl.size(); j++) {
00082             membershipFunction[j] = 1 - ((rcl.get(j)[4])/maxSc);
00083         }
00084         double minMemFunc = Double.MAX_VALUE;
00085         int posSelected = -1;
00086         for(int i = 0; i < rcl.size(); i++) {
00087             if(minMemFunc > membershipFunction[i]) {
00088                 minMemFunc = membershipFunction[i];
00089                 posSelected = i;
00090             }
00091         }
00092         return posSelected;
00093     }
00094
00095     public int fuzzySelectionAlphaCutRCL(ArrayList< double[] > rcl, double alpha) {
00096         ArrayList< double[] > rclAlphaCut = new ArrayList< double[] >();
00097         ArrayList< Integer > rclPos = new ArrayList< Integer >();
00098         double[] membershipFunction = new double[rcl.size()];
00099         double maxSc = this.getMaxScore();
00100         for(int j=0; j < rcl.size(); j++) {
00101             membershipFunction[j] = 1 - ((rcl.get(j)[4])/maxSc);
00102             if(membershipFunction[j] <= alpha) {
00103                 rclAlphaCut.add(rcl.get(j));
00104                 rclPos.add(j);
00105             }
00106         }
00107         int posSelected = -1;
00108         if(rclAlphaCut.size() > 0) {
00109             posSelected = rclPos.get(aleatorySelectionRCL(rclAlphaCut.size()));
00110         } else {
00111             posSelected = aleatorySelectionRCL(rcl.size());
00112         }
00113         return posSelected;
00114     }
00115
00116     public void computeGreedySolution(int maxSizeRCL) {
00117         // inicialización
00118         this.solution.initSolution();
00119
00120         // tiempo de salida y score por ruta y cliente
00121         ArrayList<ArrayList<Double» departureTimesPerClient = new ArrayList<ArrayList<Double»();
00122         ArrayList<Double> init = new ArrayList<Double>();
00123         for(int z = 0; z <
    this.solution.getProblem().getPOIs()+this.solution.getProblem().getVehicles(); z++) {init.add(0.0);}
00124         departureTimesPerClient.add(0, init);
00125
00126         // clientes
00127         ArrayList<Integer> customers = new ArrayList<Integer>();
00128         for(int j = 1; j <= this.solution.getProblem().getPOIs(); j++) { customers.add(j); }
00129
00130         // Evaluar coste incremental de los elementos candidatos
00131         ArrayList< double[] > candidates = this.comprehensiveEvaluation(customers,
    departureTimesPerClient);
00132
00133         Collections.sort(candidates, new Comparator<double[]>() {
00134             public int compare(double[] a, double[] b) {
00135                 return Double.compare(a[a.length-2], b[b.length-2]);
00136             }
00137         });
00138
00139         int maxTRCL = maxSizeRCL;
00140         boolean existCandidates = true;
00141
00142         while(!customers.isEmpty() && existCandidates) {
00143             if(!candidates.isEmpty()) {
00144                 //Construir lista restringida de candidatos
00145                 ArrayList< double[] > rcl = new ArrayList< double[] >();
00146                 maxTRCL = maxSizeRCL;
```

```
00147                 if(maxTRCL > candidates.size()) { maxTRCL = candidates.size(); }
00148                 for(int j=0; j < maxTRCL; j++) { rcl.add(candidates.get(j)); }
00149 
00150                 //Selección aleatoria o fuzzy de candidato de la lista restringida
00151                 int posSelected = -1;
00152                 int selection = 3;
00153                 double alpha = 0.8;
00154                 switch (selection) {
00155                     case 1:  posSelected = this.aleatorySelectionRCL(maxTRCL);  // Selección aleatoria
00156                              break;
00157                     case 2:  posSelected = this.fuzzySelectionBestFDRCL(rcl);   // Selección fuzzy con
      mejor valor de alpha
00158                              break;
00159                     case 3:  posSelected = this.fuzzySelectionAlphaCutRCL(rcl, alpha); // Selección
      fuzzy con alpha corte aleatoria
00160                              break;
00161                     default: posSelected = this.aleatorySelectionRCL(maxTRCL);  // Selección aleatoria
      por defecto
00162                              break;
00163                 }
00164 
00165                 double[] candidateSelected = rcl.get(posSelected);
00166                 for(int j=0; j < customers.size(); j++) {
00167                     if(customers.get(j)==candidateSelected[0]) {
00168                         customers.remove(j);
00169                     }
00170                 }
00171 
00172                 updateSolution(candidateSelected, departureTimesPerClient);
00173 
00174             } else { // No hay candidatos a insertar en la solución, crear otra ruta
00175                 if(this.solution.getCreatedRoutes() < this.solution.getProblem().getVehicles()) {
00176                     int newDepot = this.solution.addRoute();
00177                     ArrayList<Double> initNew = new ArrayList<Double>();
00178                     for(int z = 0; z <
      this.solution.getProblem().getPOIs()+this.solution.getProblem().getVehicles(); z++)
      {initNew.add(0.0);}
00179                     departureTimesPerClient.add(initNew);
00180                 }
00181                 else {
00182                     existCandidates = false;
00183                 }
00184             }
00185             //Reevaluar coste incremental de los elementos candidatos
00186             candidates.clear();
00187             candidates = this.comprehensiveEvaluation(customers, departureTimesPerClient);
00188             Collections.sort(candidates, new Comparator<double[]>() {
00189                 public int compare(double[] a, double[] b) {
00190                     return Double.compare(a[a.length-2], b[b.length-2]);
00191                 }
00192             });
00193         }
00194 
00195     }
00196 
00197     public void updateSolution(double[] candidateSelected, ArrayList< ArrayList< Double > >
      departureTimes) {
00198         // Inserción del cliente en la ruta  return: cliente, ruta, predecesor, coste
00199         this.solution.setPredecessor((int)candidateSelected[0], (int)candidateSelected[2]);
00200         this.solution.setSuccessor((int)candidateSelected[0],
      this.solution.getSuccessor((int)candidateSelected[2]));
00201         this.solution.setSuccessor((int)candidateSelected[2], (int)candidateSelected[0]);
00202         this.solution.setPredecessor(this.solution.getSuccessor((int)candidateSelected[0]),
      (int)candidateSelected[0]);
00203 
00204         // Actualización de las estructuras de datos y conteo a partir de la posición a insertar
00205         double costInsertionPre =
      departureTimes.get((int)candidateSelected[1]).get((int)candidateSelected[2]);
00206         ArrayList<Double> route = departureTimes.get((int)candidateSelected[1]);
00207         int pre=(int)candidateSelected[2], suc=-1;
00208         int depot = this.solution.getIndexRoute((int)candidateSelected[1]);
00209         do {
00210             suc = this.solution.getSuccessor(pre);
00211             costInsertionPre += this.solution.getDistance(pre, suc);
00212 
00213             if(costInsertionPre < this.solution.getProblem().getReadyTime(suc)) {
00214                 costInsertionPre = this.solution.getProblem().getReadyTime(suc);
00215             }
00216             costInsertionPre += this.solution.getProblem().getServiceTime(suc);
00217 
00218             if(!this.solution.isDepot(suc))
00219                 route.set(suc, costInsertionPre);
00220             pre = suc;
00221         } while((suc != depot));
00222 
00223         // Actualiza tiempos
00224         departureTimes.set((int)candidateSelected[1], route);
```

```
00225     }
00226
00227     //return: cliente, ruta, predecesor, coste tiempo, score
00228     public ArrayList< double[] > comprehensiveEvaluation(ArrayList<Integer> customers, ArrayList<
      ArrayList< Double > > departureTimes) {
00229         ArrayList< double[] > candidatesList = new ArrayList< double[] >();
00230         double[] infoCandidate = new double[5];
00231         boolean validFinalInsertion = true;
00232         infoCandidate[0] = -1;
00233         infoCandidate[1] = -1;
00234         infoCandidate[2] = -1;
00235         infoCandidate[3] = Double.MAX_VALUE;
00236         infoCandidate[4] = -1;
00237
00238         for(int c = 0; c < customers.size(); c++) { // clientes disponibles
00239             for(int k = 0; k < this.solution.getCreatedRoutes(); k++) { // rutas creadas
00240                 validFinalInsertion = true;
00241                 int depot = this.solution.getIndexRoute(k);
00242                 int pre=-1, suc=-1;
00243                 double costInsertion = 0;
00244                 pre = depot;
00245                 int candidate = customers.get(c);
00246                 do {                                                    // recorremos la ruta
00247                     validFinalInsertion = true;
00248                     suc = this.solution.getSuccessor(pre);
00249                     double timesUntilPre = departureTimes.get(k).get(pre) +
      this.solution.getDistance(pre, candidate);
00250                     if(timesUntilPre < (this.solution.getProblem().getDueTime(candidate))) {
00251                         double costCand = 0;
00252                         if(timesUntilPre < this.solution.getProblem().getReadyTime(candidate)) {
00253                             costCand = this.solution.getProblem().getReadyTime(candidate);
00254                         } else { costCand = timesUntilPre; }
00255                         costCand +=  this.solution.getProblem().getServiceTime(candidate);
00256                         if(costCand > this.solution.getProblem().getMaxTimePerRoute()) {
      validFinalInsertion = false; }
00257
00258                         // Comprobar TW desde candidate hasta sucesor
00259                         double timesUntilSuc = costCand + this.solution.getDistance(candidate, suc);
00260                         if(timesUntilSuc < (this.solution.getProblem().getDueTime(suc))) {

00261                             double costSuc = 0;
00262                             if(timesUntilSuc < this.solution.getProblem().getReadyTime(suc)) {
00263                                 costSuc = this.solution.getProblem().getReadyTime(suc);
00264                             } else { costSuc = timesUntilSuc; }
00265                             costSuc +=  this.solution.getProblem().getServiceTime(suc);
00266                             costInsertion = costSuc;
00267                             if(costSuc > this.solution.getProblem().getMaxTimePerRoute()) {
      validFinalInsertion = false;}
00268
00269                             int pre2=suc, suc2 = -1;
00270                             if(suc != depot)
00271                                 do {
00272                                     suc2 = this.solution.getSuccessor(pre2);
00273                                     double timesUntilSuc2 = costInsertion +
      this.solution.getDistance(pre2, suc2);
00274                                     if(timesUntilSuc2 < (this.solution.getProblem().getDueTime(suc2)))
      {
00275                                         if(timesUntilSuc2 <
      this.solution.getProblem().getReadyTime(suc2)) {
00276                                             costInsertion =
      this.solution.getProblem().getReadyTime(suc2);
00277                                         } else { costInsertion = timesUntilSuc2; }
00278                                         costInsertion +=
      this.solution.getProblem().getServiceTime(suc2);
00279                                         if(costInsertion >
      this.solution.getProblem().getMaxTimePerRoute()) { validFinalInsertion = false; }
00280                                     } else { validFinalInsertion = false; }
00281                                     pre2 = suc2;
00282                                 } while((suc2 != depot) && validFinalInsertion);
00283                         } else { validFinalInsertion = false; }
00284                     } else { validFinalInsertion = false; }
00285
00286                     if(validFinalInsertion==true) { // cliente, ruta, predecesor, coste
00287                         if(costInsertion < infoCandidate[3]) {
00288                             infoCandidate[0] = candidate; infoCandidate[1] = k; infoCandidate[2] =
      pre; infoCandidate[3] = costInsertion; infoCandidate[4] =
      this.solution.getProblem().getScore(candidate); // cliente, ruta, predecesor, coste, score
00289                         }
00290                     }
00291
00292                     pre = suc;
00293                 } while(suc != depot);
00294             } //rutas creadas
00295
00296             // almacenamos en la lista de candidatos la mejor posición de inserción para el cliente
00297             if(infoCandidate[0]!=-1 && infoCandidate[1]!=-1 && infoCandidate[2]!=-1 &&
      infoCandidate[3] != Double.MAX_VALUE && infoCandidate[4]!=-1) {
```

```
00298                 double[] infoCandidate2 = new double[5];
00299                 infoCandidate2[0] = infoCandidate[0];  infoCandidate2[1] = infoCandidate[1];
00300                 infoCandidate2[2] = infoCandidate[2];  infoCandidate2[3] = infoCandidate[3];
00301                 infoCandidate2[4] = infoCandidate[4];
00302                 candidatesList.add(infoCandidate2);
00303             }
00304             validFinalInsertion = true;
00305             infoCandidate[0] = -1;  infoCandidate[1] = -1;
00306             infoCandidate[2] = -1;  infoCandidate[3] = Double.MAX_VALUE;
00307             infoCandidate[4] = -1;
00308         } // cliente
00309
00310         return candidatesList;
00311     }
00312
00313     public TOPTWSolution getSolution() {
00314         return solution;
00315     }
00316
00317     public void setSolution(TOPTWSolution solution) {
00318         this.solution = solution;
00319     }
00320
00321     public int getSolutionTime() {
00322         return solutionTime;
00323     }
00324
00325     public void setSolutionTime(int solutionTime) {
00326         this.solutionTime = solutionTime;
00327     }
00328
00329     public double getMaxScore() {
00330         double maxSc = -1.0;
00331         for(int i = 0; i < this.solution.getProblem().getScore().length; i++) {
00332             if(this.solution.getProblem().getScore(i) > maxSc)
00333                 maxSc = this.solution.getProblem().getScore(i);
00334         }
00335         return maxSc;
00336     }
00337
00338 }
```

7.17 src/main/java/top/TOPTWReader.java File Reference

Classes

- class top.TOPTWReader

Packages

- package top

7.18 TOPTWReader.java

Go to the documentation of this file.

```
00001 package top;
00002
00003 import java.io.BufferedReader;
00004 import java.io.File;
00005 import java.io.FileReader;
00006 import java.io.IOException;
00007
00008 import es.ull.esit.utilities.ExpositoUtilities;
00009
00010 public class TOPTWReader {
00011
00012     public static TOPTW readProblem(String filePath) {
00013         TOPTW problem = null;
00014         BufferedReader reader = null;
00015         try {
```

```
00016             File instaceFile = new File(filePath);
00017             reader = new BufferedReader(new FileReader(instaceFile));
00018             String line = reader.readLine();
00019             line = ExpositoUtilities.simplifyString(line);
00020             String[] parts =line.split(" ");
00021             problem = new TOPTW(Integer.parseInt(parts[2]), Integer.parseInt(parts[1]));
00022             line = reader.readLine();
00023             line = null; parts = null;
00024             for (int i = 0; i < problem.getPOIs()+1; i++) {
00025                 line = reader.readLine();
00026                 line = ExpositoUtilities.simplifyString(line);
00027                 parts = line.split(" ");
00028                 problem.setX(i, Double.parseDouble(parts[1]));
00029                 problem.setY(i, Double.parseDouble(parts[2]));
00030                 problem.setServiceTime(i, Double.parseDouble(parts[3]));
00031                 problem.setScore(i, Double.parseDouble(parts[4]));
00032                 if(i==0) {
00033                     problem.setReadyTime(i, Double.parseDouble(parts[7]));
00034                     problem.setDueTime(i, Double.parseDouble(parts[8]));
00035                 }
00036                 else {
00037                     problem.setReadyTime(i, Double.parseDouble(parts[8]));
00038                     problem.setDueTime(i, Double.parseDouble(parts[9]));
00039                 }
00040                 line = null; parts = null;
00041             }
00042             problem.calculateDistanceMatrix();
00043         } catch (IOException e) {
00044             System.err.println(e);
00045             System.exit(0);
00046         } finally {
00047             if (reader != null) {
00048                 try {
00049                     reader.close();
00050                 } catch (IOException ex) {
00051                     System.err.println(ex);
00052                     System.exit(0);
00053                 }
00054             }
00055         }
00056         problem.setMaxTimePerRoute(problem.getDueTime(0));
00057         return problem;
00058     }
00059
00060 }
```

7.19 src/main/java/top/TOPTWRoute.java File Reference

Classes

- class top.TOPTWRoute

Packages

- package top

7.20 TOPTWRoute.java

Go to the documentation of this file.

```
00001 package top;
00002
00003 public class TOPTWRoute {
00004     int predecessor;
00005     int succesor;
00006     int id;
00007
00008     TOPTWRoute() {
00009
00010     }
00011
```

```
00012     TOPTWRoute(int pre, int succ, int id) {
00013         this.predecessor = pre;
00014         this.succesor = succ;
00015         this.id = id;
00016     }
00017 
00018     public int getPredeccesor() {
00019         return this.predecessor;
00020     }
00021 
00022     public int getSuccesor() {
00023         return this.succesor;
00024     }
00025 
00026     public int getId() {
00027         return this.id;
00028     }
00029 
00030     public void setPredeccesor(int pre) {
00031         this.predecessor = pre;
00032     }
00033 
00034     public void setSuccesor(int suc) {
00035         this.succesor = suc;
00036     }
00037 
00038     public void setId(int id) {
00039         this.id = id;
00040     }
00041 }
```

7.21 src/main/java/top/TOPTWSolution.java File Reference

Classes

- class top.TOPTWSolution

Packages

- package top

7.22 TOPTWSolution.java

Go to the documentation of this file.

```
00001 package top;
00002 
00003 import java.util.Arrays;
00004 
00005 import es.ull.esit.utilities.ExpositoUtilities;
00006 
00007 public class TOPTWSolution {
00008     public static final int NO_INITIALIZED = -1;
00009     private TOPTW problem;
00010     private int[] predecessors;
00011     private int[] successors;
00012     private double[] waitingTime;
00013     private int[] positionInRoute;
00014 
00015     private int[] routes;
00016     private int availableVehicles;
00017     private double objectiveFunctionValue;
00018 
00019     public TOPTWSolution(TOPTW problem) {
00020         this.problem = problem;
00021         this.availableVehicles = this.problem.getVehicles();
00022         this.predecessors = new int[this.problem.getPOIs()+this.problem.getVehicles()];
00023         this.successors = new int[this.problem.getPOIs()+this.problem.getVehicles()];
00024         this.waitingTime = new double[this.problem.getPOIs()];
00025         this.positionInRoute = new int[this.problem.getPOIs()];
00026         Arrays.fill(this.predecessors, TOPTWSolution.NO_INITIALIZED);
```

```
00027         Arrays.fill(this.successors, TOPTWSolution.NO_INITIALIZED);
00028         Arrays.fill(this.waitingTime, TOPTWSolution.NO_INITIALIZED);
00029         Arrays.fill(this.positionInRoute, TOPTWSolution.NO_INITIALIZED);
00030         this.routes = new int[this.problem.getVehicles()];
00031         this.objectiveFunctionValue = TOPTWEvaluator.NO_EVALUATED;
00032     }
00033
00034     public void initSolution() {
00035         this.predecessors = new int[this.problem.getPOIs()+this.problem.getVehicles()];
00036         this.successors = new int[this.problem.getPOIs()+this.problem.getVehicles()];
00037         Arrays.fill(this.predecessors, TOPTWSolution.NO_INITIALIZED);
00038         Arrays.fill(this.successors, TOPTWSolution.NO_INITIALIZED);
00039         this.routes = new int[this.problem.getVehicles()];
00040         Arrays.fill(this.routes, TOPTWSolution.NO_INITIALIZED);
00041         this.routes[0] = 0;
00042         this.predecessors[0] = 0;
00043         this.successors[0] = 0;
00044         this.availableVehicles = this.problem.getVehicles() - 1;
00045     }
00046
00047     public boolean isDepot(int c) {
00048         for(int i = 0; i < this.routes.length; i++) {
00049             if(c==this.routes[i]) {
00050                 return true;
00051             }
00052         }
00053         return false;
00054     }
00055
00056     public boolean equals(TOPTWSolution otherSolution) {
00057         for (int i = 0; i < this.predecessors.length; i++) {
00058             if (this.predecessors[i] != otherSolution.predecessors[i]) {
00059                 return false;
00060             }
00061         }
00062         return true;
00063     }
00064
00065     public int getAvailableVehicles() {
00066         return this.availableVehicles;
00067     }
00068
00069     public int getCreatedRoutes() {
00070         return this.problem.getVehicles() - this.availableVehicles;
00071     }
00072
00073     public double getDistance(int x, int y) {
00074         return this.problem.getDistance(x, y);
00075     }
00076
00077     public void setAvailableVehicles(int availableVehicles) {
00078         this.availableVehicles = availableVehicles;
00079     }
00080
00081     public int getPredecessor(int customer) {
00082         return this.predecessors[customer];
00083     }
00084
00085     public int[] getPredecessors() {
00086         return this.predecessors;
00087     }
00088
00089     public TOPTW getProblem() {
00090         return this.problem;
00091     }
00092
00093     public double getObjectiveFunctionValue() {
00094         return this.objectiveFunctionValue;
00095     }
00096
00097     public int getPositionInRoute(int customer) {
00098         return this.positionInRoute[customer];
00099     }
00100
00101     public int getSuccessor(int customer) {
00102         return this.successors[customer];
00103     }
00104
00105     public int[] getSuccessors() {
00106         return this.successors;
00107     }
00108
00109     public int getIndexRoute(int index) {
00110         return this.routes[index];
00111     }
00112
00113     public double getWaitingTime(int customer) {
```

```
00114         return this.waitingTime[customer];
00115     }
00116 
00117     public void setObjectiveFunctionValue(double objectiveFunctionValue) {
00118         this.objectiveFunctionValue = objectiveFunctionValue;
00119     }
00120 
00121     public void setPositionInRoute(int customer, int position) {
00122         this.positionInRoute[customer] = position;
00123     }
00124 
00125     public void setPredecessor(int customer, int predecessor) {
00126         this.predecessors[customer] = predecessor;
00127     }
00128 
00129     public void setSuccessor(int customer, int succesor) {
00130         this.successors[customer] = succesor;
00131     }
00132 
00133     public void setWaitingTime(int customer, int waitingTime) {
00134         this.waitingTime[customer] = waitingTime;
00135     }
00136 
00137     public String getInfoSolution() {
00138         final int COLUMN_WIDTH = 15;
00139         String text = "\n"+"NODES: " + this.problem.getPOIs() + "\n" + "MAX TIME PER ROUTE: " +
      this.problem.getMaxTimePerRoute() + "\n" + "MAX NUMBER OF ROUTES: " + this.problem.getMaxRoutes() +
      "\n";
00140         String textSolution = "\n"+"SOLUTION: "+"\n";
00141         double costTimeSolution = 0.0, fitnessScore = 0.0;
00142         boolean validSolution = true;
00143         for(int k = 0; k < this.getCreatedRoutes(); k++) { // rutas creadas
00144             String[] strings = new String[]{"\n" + "ROUTE " + k };
00145             int[] width = new int[strings.length];
00146             Arrays.fill(width, COLUMN_WIDTH);
00147             text += ExpositoUtilities.getFormat(strings, width) + "\n";
00148             strings = new String[]{"CUST NO.", "X COORD.", "Y. COORD.", "READY TIME", "DUE DATE",
      "ARRIVE TIME", " LEAVE TIME", "SERVICE TIME"};
00149             width = new int[strings.length];
00150             Arrays.fill(width, COLUMN_WIDTH);
00151             text += ExpositoUtilities.getFormat(strings, width) + "\n";
00152             strings = new String[strings.length];
00153             int depot = this.getIndexRoute(k);
00154             int pre=-1, suc=-1;
00155             double costTimeRoute = 0.0, fitnessScoreRoute = 0.0;
00156             pre = depot;
00157             int index = 0;
00158             strings[index++] = "" + pre;
00159             strings[index++] = "" + this.getProblem().getX(pre);
00160             strings[index++] = "" + this.getProblem().getY(pre);
00161             strings[index++] = "" + this.getProblem().getReadyTime(pre);
00162             strings[index++] = "" + this.getProblem().getDueTime(pre);
00163             strings[index++] = "" + 0;
00164             strings[index++] = "" + 0;
00165             strings[index++] = "" + this.getProblem().getServiceTime(pre);
00166             text += ExpositoUtilities.getFormat(strings, width);
00167             text += "\n";
00168             do {                // recorremos la ruta
00169                 index = 0;
00170                 suc = this.getSuccessor(pre);
00171                 textSolution += pre+" - ";
00172                 strings[index++] = "" + suc;
00173                 strings[index++] = "" + this.getProblem().getX(suc);
00174                 strings[index++] = "" + this.getProblem().getY(suc);
00175                 strings[index++] = "" + this.getProblem().getReadyTime(suc);
00176                 strings[index++] = "" + this.getProblem().getDueTime(suc);
00177                 costTimeRoute += this.getDistance(pre, suc);
00178                 if(costTimeRoute < (this.getProblem().getDueTime(suc))) {
00179                     if(costTimeRoute < this.getProblem().getReadyTime(suc)) {
00180                         costTimeRoute = this.getProblem().getReadyTime(suc);
00181                     }
00182                     strings[index++] = "" + costTimeRoute;
00183                     costTimeRoute +=  this.getProblem().getServiceTime(suc);
00184                     strings[index++] = "" + costTimeRoute;
00185                     strings[index++] = "" + this.getProblem().getServiceTime(pre);
00186                     if(costTimeRoute > this.getProblem().getMaxTimePerRoute()) { validSolution =
      false; }
00187                     fitnessScoreRoute += this.problem.getScore(suc);
00188                 } else { validSolution = false; }
00189                 pre = suc;
00190                 text += ExpositoUtilities.getFormat(strings, width);
00191                 text += "\n";
00192             } while(suc != depot);
00193             textSolution += suc+"\n";
00194             costTimeSolution += costTimeRoute;
00195             fitnessScore += fitnessScoreRoute;
00196         }
```

```
00197         textSolution += "FEASIBLE SOLUTION: "+validSolution+"\n"+"SCORE: "+fitnessScore+"\n"+"TIME
      COST: "+costTimeSolution+"\n";
00198         return textSolution+text;
00199     }
00200
00201     public double evaluateFitness() {
00202         double objectiveFunction = 0.0;
00203         double objectiveFunctionPerRoute = 0.0;
00204         for(int k = 0; k < this.getCreatedRoutes(); k++) {
00205             int depot = this.getIndexRoute(k);
00206             int pre=depot, suc = -1;
00207             do {
00208                 suc = this.getSuccessor(pre);
00209                 objectiveFunctionPerRoute = objectiveFunctionPerRoute + this.problem.getScore(suc);
00210                 pre = suc;
00211             } while((suc != depot));
00212             objectiveFunction = objectiveFunction + objectiveFunctionPerRoute;
00213             objectiveFunctionPerRoute = 0.0;
00214         }
00215         return objectiveFunction;
00216     }
00217
00218     public int addRoute() {
00219         int depot = this.problem.getPOIs();
00220         depot++;
00221         int routePos = 1;
00222         for(int i = 0; i < this.routes.length; i++) {
00223             if(this.routes[i] != -1 && this.routes[i] != 0) {
00224                 depot = this.routes[i];
00225                 depot++;
00226                 routePos = i+1;
00227             }
00228         }
00229         this.routes[routePos] = depot;
00230         this.availableVehicles--;
00231         this.predecessors[depot] = depot;
00232         this.successors[depot] = depot;
00233         this.problem.addNodeDepot();
00234         return depot;
00235     }
00236
00237     public double printSolution() {
00238         for(int k = 0; k < this.getCreatedRoutes(); k++) {
00239                 int depot = this.getIndexRoute(k);
00240                 int pre=depot, suc = -1;
00241                 do {
00242                     suc = this.getSuccessor(pre);
00243                     System.out.print(pre+" - ");
00244                     pre = suc;
00245                 } while((suc != depot));
00246                 System.out.println(suc+"  ");
00247         }
00248         double fitness = this.evaluateFitness();
00249         System.out.println("SC="+fitness);
00250         return fitness;
00251     }
00252
00253 }
```

Index