Reference Manual

Generated by Doxygen 1.7.2

Thu Dec 16 2010 15:53:58

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 MTRand Class Reference

Mersenne Twister random number generator.

`#include <MersenneTwister.h>`

Public Types

- enum { **N** = 624 }
- enum { **SAVE** = N + 1 }
- typedef unsigned long **uint32**

Public Member Functions

- **MTRand** (const uint32 &oneSeed)
- **MTRand** (uint32 *const bigSeed, uint32 const seedLength=N)
- double **rand** ()
- double **rand** (const double &n)
- double **randExc** ()
- double **randExc** (const double &n)
- double **randDblExc** ()
- double **randDblExc** (const double &n)
- uint32 **randInt** ()
- uint32 **randInt** (const uint32 &n)
- double **operator()** ()
- double **rand53** ()
- double **randNorm** (const double &mean=0.0, const double &variance=0.0)
- void **seed** (const uint32 oneSeed)
- void **seed** (uint32 *const bigSeed, const uint32 seedLength=N)
- void **seed** ()

- void **save** (uint32 *saveArray) const
- void **load** (uint32 *const loadArray)

Protected Types

- enum { **M** = 397 }

Protected Member Functions

- void **initialize** (const uint32 oneSeed)
- void **reload** ()
- uint32 **hiBit** (const uint32 &u) const
- uint32 **loBit** (const uint32 &u) const
- uint32 **loBits** (const uint32 &u) const
- uint32 **mixBits** (const uint32 &u, const uint32 &v) const
- uint32 **twist** (const uint32 &m, const uint32 &s0, const uint32 &s1) const

Static Protected Member Functions

- static uint32 **hash** (time_t t, clock_t c)

Protected Attributes

- uint32 **state** [N]
- uint32 * **pNext**
- int **left**

Friends

- std::ostream & **operator**<< (std::ostream &os, const MTRand &mtrand)
- std::istream & **operator**>> (std::istream &is, MTRand &mtrand)

2.1.1 Detailed Description

Mersenne Twister random number generator. MersenneTwister.h Mersenne Twister random number generator -- a C++ class MTRand Based on code by Makoto Matsumoto, Takuji Nishimura, and Shawn Cokus Richard J. Wagner v1.0 15 May 2003 rjwagner@writeme.com

The Mersenne Twister is an algorithm for generating random numbers. It was designed with consideration of the flaws in various other generators. The period, 2^19937-1, and the order of equidistribution, 623 dimensions, are far greater. The generator is also fast; it avoids multiplication and division, and it benefits from caches and pipelines. For more information see the inventors' web page at http://www.math.keio.ac.jp/~matumoto/emt.html

Reference M. Matsumoto and T. Nishimura, "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator", ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, pp 3-30.

The original code included the following notice:

When you use this, send an email to: matumoto@math.keio.ac.jp with an appropriate reference to your work.

It would be nice to CC: rjwagner@writeme.com and Cokus@math.washington.edu when you write.

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

- MersenneTwister.h

2.2 tData Struct Reference

`#include <dataTypes.h>`

Public Attributes

- unsigned **from**

- unsigned **to**
- double **length**

2.2.1 Detailed Description

Cost between nodes: Structure to store the cost between pairs of node.

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

- dataTypes.h

2.3 tDemand Struct Reference

Public Attributes

- unsigned **delta**
- std::vector< tDemandCostumer > **demandsCostumers**

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

- dataTypes.h

2.4 tDemandCostumer Struct Reference

`#include <dataTypes.h>`

Public Attributes

- double **type**
- unsigned **probability**

2.4.1 Detailed Description

Demand Specification Structure: "delta" is defined as the percentage change in capacity of the fleet of vehicles. This value will be bounded between 0 and 100. The bigger this value is, the more capacity the vehicles will have. Moreover, we will also specify the demands with their respective probabilites.

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

- dataTypes.h

2.5 tPos Struct Reference

`#include <dataTypes.h>`

Public Attributes

- unsigned **id**
- double **lat**
- double **lng**

2.5.1 Detailed Description

Position of the costumers: For each costumer, we will hold the the original id and its position (lat, lng).

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

- dataTypes.h

2.6 tServiceTimeCostumer Struct Reference

`#include <dataTypes.h>`

Public Attributes

- double **type**
- unsigned **probability**

2.6.1 Detailed Description

Service Time Specification Structure For each costumer, a type of time service is defined with its probability

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

- dataTypes.h

2.7 tSolomon Struct Reference

`#include <dataTypes.h>`

Public Attributes

- unsigned **id**
- double **X**
- double **Y**
- double **demand**
- unsigned long **opens**
- unsigned long **closes**
- unsigned long **serviceTime**

2.7.1 Detailed Description

Solomon-like Specification Structure In order to output the general specifications of the costumers, we will use the structure emplied in the VRP Solomon's instance-set.

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

- dataTypes.h

2.8 tTimeWindow Struct Reference

Public Attributes

- tTimeWindowDepot **timeWindowDepot**
- std::vector< tTimeWindowCostumer > **timeWindowsCostumers**

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

- dataTypes.h

2.9 tTimeWindowCostumer Struct Reference

Public Attributes

- double **opens**
- double **closes**
- unsigned **probability**

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

- dataTypes.h

2.10 tTimeWindowDepot Struct Reference

`#include <dataTypes.h>`

Public Attributes

- double **opens**
- double **closes**

2.10.1 Detailed Description

Time Window Scepecifications Structure: In order to avoid the use of inheritance and reduce the possible overhead to the minimum, we will use two structs, one for the depot and another for the costumers.

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

- dataTypes.h

2.11 VRPTWInstanceGenerator Class Reference

Public Member Functions

- VRPTWInstanceGenerator ()
  Default Ctor.
- ∼VRPTWInstanceGenerator ()
  Default Destructor.
- void readDistancesFile (const char ∗distanceFileName)
  Method that reads the distance between each par of costumers.
- void readTimesFile (const char ∗timeFilename)
  Method that reads the time between each par of costumers.
- void readIds (const char ∗idsFileName)
  Method that reads the real ids of the costumers.
- void readPositions (const char ∗idslatlngFileName)
  Method that reads the position of the costumers.
- void readTimeWindowsFile (const char ∗fileName)
  Method that parsers the time windows specifications.

- void readDemandsFile (const char *fileName)

  Method that parsers the demands specifications.

- void readServiceTimesFile (const char *fileName)

  Method that parsers the service times specifications.

- void setSize (unsigned)

  Sets the size of the instance.

- void setSeed (const std::string &, unsigned)

  Sets the seeds to generate the data for this instance.

- void setPrefix (const std::string &)

  Sets the prefix for the output files.

- void generateMatrices ()

  Method that generates the distance and time windows matrices with random costumers.

- void generateTimeWindows ()

  Method that associates a type of time window to each costumer.

- void generateDemands ()

  Method that associates a type of demand to each costumer.

- void generateServiceTimes ()

  Method that associates a type of service time to each costumer.

- void generateAll ()

  Method that call all generates.

- void printParameters ()

  Method to print the parameters.

- void printTimeMatrix ()

  Method to print the time matrix.

- void printDistanceMatrix ()

  Method to print the distance matrix.

- void printSpecifications ()

  Method to print the specifications.

- void print ()

  Method that prints all.

- void printAll ()

  Method that prints all.

- void writeDistanceMatrix ()

  Method that writes the distance matrix to a file.

- void writeTimeMatrix ()

  Method that writes the time matrix to a file.

- void writeSpecifications ()

  Method that writes the specification to a file.

- void writeAll ()

  Method that call all writes.

Protected Member Functions

- double getData (const rawInfoVector &, unsigned, unsigned)

  Method that, given a table and 'from' 'to' information, returns the length in time or distance.

- unsigned getCategoryRandomly (double, const accProbabilityType &)

  This method will receive a random number generator and a vector of accumulative probability and will return a category.

- void error (const std::string &)

  Method to output errors.

- void warning (const std::string &)

  Method to output warnings.

- unsigned getPosition (unsigned)

  Method to get the position of a costumer within vector.

- void generateDistanceMatrix ()

  Method to generate the distance matrix for this instance.

- void generateTimeMatrix ()

  Method to generate the time matrix for this instance.

2.11.1 Constructor & Destructor Documentation

2.11.1.1 VRPTWInstanceGenerator::VRPTWInstanceGenerator ( )

Default Ctor.

Ctor. It sets default values for data members

2.11.1.2 VRPTWInstanceGenerator::~VRPTWInstanceGenerator ( )

Default Destructor.

Dtor. It does nothing

2.11.2 Member Function Documentation

2.11.2.1 void VRPTWInstanceGenerator::error (const std::string & *errorMessage*) [protected]

Method to output errors.

Method that ouputs message and force the exit of the app.

Parameters

errorMessage	is the message to be output.

2.11.2.2 void VRPTWInstanceGenerator::generateAll ( )

Method that call all generates.

Method that invokes serveral other methods to generate the instance.

2.11.2.3 void VRPTWInstanceGenerator::generateDemands ( )

Method that associates a type of demand to each costumer.

Method that generates a vector of size (size) with all the demands.

2.11.2.4 void VRPTWInstanceGenerator::generateDistanceMatrix ( ) [protected]

Method to generate the distance matrix for this instance.

Method that generates the distance matrix using random ids.

2.11.2.5 void VRPTWInstanceGenerator::generateMatrices ( )

Method that generates the distance and time windows matrices with random costumers.

Method that generates a vector of random ids (size value random ids) and invokes generateDistanceMatrix and generateTimeMatrix().

2.11.2.6 void VRPTWInstanceGenerator::generateServiceTimes ( )

Method that associates a type of service time to each costumer.

Method that generates a vector of size (size) with all the time services.

2.11.2.7 void VRPTWInstanceGenerator::generateTimeMatrix ( ) [protected]

Method to generate the time matrix for this instance.

Method that generates the travel time matrix using random ids.

2.11.2.8 void VRPTWInstanceGenerator::generateTimeWindows ( )

Method that associates a type of time window to each costumer.

Method that generates a vector of size (size) with all the time windows.

2.11.2.9 unsigned VRPTWInstanceGenerator::getCategoryRandomly (double *randNumber,* const accProbabilityType & *accProbability*) [protected]

This method will receive a random number generator and a vector of accumulative probability and will return a category.

Method that given a vector with accumulate probabilities returns the section in which a random number falls.

Parameters

randNumber	is the random number we want to know the place it falls in
accProbability	is the vector containing the accumuate probabilities.

Returns

the section in which the random number falls.

2.11.2.10 double VRPTWInstanceGenerator::getData (const rawInfoVector & *table,* unsigned *from,* unsigned *to*) [protected]

Method that, given a table and 'from' 'to' information, returns the length in time or distance.

Method that given the two first two elements of a table of nx3 returns the third element.

Parameters

table	is the matrix in which we want to look for the third element.
from	is the first element or index.
to	is the second element or index.

Returns

the value of the third element.

2.11.2.11 unsigned VRPTWInstanceGenerator::getPosition (unsigned *id*) [protected]

Method to get the position of a costumer within vector.

Method that returns the index of a given element (id) within the positions vector

Parameters

id	is the id of the element we are looking for

Returns

the position (index) within the vector

2.11.2.12 void VRPTWInstanceGenerator::print ( )

Method that prints all.

Method that invokes all printing methods.

2.11.2.13 void VRPTWInstanceGenerator::printAll ( )

Method that prints all.

Method that invokes all printing methods.

2.11.2.14 void VRPTWInstanceGenerator::printDistanceMatrix ( )

Method to print the distance matrix.

Method that prints the distance matrix.

2.11.2.15 void VRPTWInstanceGenerator::printParameters ( )

Method to print the parameters.

Method that prints the parameters given by the user (debug purposes).

2.11.2.16 void VRPTWInstanceGenerator::printSpecifications ( )

Method to print the specifications.

Method that prints the specifications.

2.11.2.17 void VRPTWInstanceGenerator::printTimeMatrix ( )

Method to print the time matrix.

Method that prints the time matrix.

2.11.2.18 void VRPTWInstanceGenerator::readDemandsFile (const char * *fileName*)

Method that parsers the demands specifications.

Parameters

fileName	is the name of the file containing the demands specifications.

Method that reads the demand specifications

Parameters

fileName	is the path to the file containing the demands specifications.

2.11.2.19 void VRPTWInstanceGenerator::readDistancesFile (const char * *distanceFileName*)

Method that reads the distance between each par of costumers.

Parameters

distanceFile-Name	is the name of the file containing the distances.

Method that reads the distances from a file between pairs of costumers.

Parameters

distanceFi-lanem	is the path to the file containing the distances information.

2.11.2.20 void VRPTWInstanceGenerator::readIds (const char * *idsFileName*)

Method that reads the real ids of the costumers.

Parameters

distanceFile-Name	is the name of the file containing the distances.

Method that reads the real ids of the costumers.

Parameters

idsFilename	is the path to the file containing the real ids.

2.11.2.21 void VRPTWInstanceGenerator::readPositions (const char * *idslatlngFileName*)

Method that reads the position of the costumers.

Parameters

idslatlngFile-Name	is the name of the file containing the distances.

Method that reads the real position of the costumers (lat, lng).

Parameters

idslatlngFile-name	is the path to the file containing the information.

2.11.2.22 void VRPTWInstanceGenerator::readServiceTimesFile (const char * *fileName*)

Method that parsers the service times specifications.

Parameters

fileName	is the name of the file containing the service times specifications.

Method that reads the service times specifications.

Parameters

fieName	is the path to the file containing the specifications.

2.11.2.23 void VRPTWInstanceGenerator::readTimesFile (const char * *timeFileName*)

Method that reads the time between each par of costumers.

Parameters

timeFile-Name	is the name of the file containing the times.

Method that reads the travel times from a file between pairs of costumers.

Parameters

timeFileName	is the path to the file containing the travel time information.

2.11.2.24 void VRPTWInstanceGenerator::readTimeWindowsFile (const char * *fileName*)

Method that parsers the time windows specifications.

Parameters

fileName	is the name of the file containing the time windows specifications.

Method that reads the time windows specification fileName is the path to the file containing the time windows specification

2.11.2.25 void VRPTWInstanceGenerator::setPrefix (const std::string & *prefix*)

Sets the prefix for the output files.

Method to set the prefix for the output files

Parameters

prefix	is the prefix of the output files.

2.11.2.26 void VRPTWInstanceGenerator::setSeed (const std::string & *object,* unsigned *value*)

Sets the seeds to generate the data for this instance.

Method to set the value of a seed.

Parameters

object	is the name of seed to be set up.
value	is the value to be set.

2.11.2.27 void VRPTWInstanceGenerator::setSize (unsigned *size*)

Sets the size of the instance.

Method to set the number of costumers of the instance.

Parameters

size	if the number of costumers.

2.11.2.28 void VRPTWInstanceGenerator::warning (const std::string & *warningMessage*) [protected]

Method to output warnings.

Method that ouputs a warning message.

Parameters

warningMessage	is the message to be output.

2.11.2.29 void VRPTWInstanceGenerator::writeAll ( )

Method that call all writes.

Method that invokes all writing methods.

2.11.2.30 void VRPTWInstanceGenerator::writeDistanceMatrix ( )

Method that writes the distance matrix to a file.

Method that creates the file containing the distance matrix.

2.11.2.31 void VRPTWInstanceGenerator::writeSpecifications ( )

Method that writes the specification to a file.

Method that creates the file containing the specifications.

2.11.2.32 void VRPTWInstanceGenerator::writeTimeMatrix ( )

Method that writes the time matrix to a file.

Method that creates the file containing the time matrix.

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

- vrptwinstancegenerator.h
- vrptwinstancegenerator.cpp

Index