MOrepo converter

1.0

Generated by Doxygen 1.8.13

Contents

Chapter 1

An introduction to the MOrepo converter

Author

Sune Lauth Gadegaard

Version

1.0.0

1.1 License

If you use the software in any academic work, please make a reference to

“An appropriate reference should go here!"

1.2 Description

This software provides a "converter" that given appropritate input, generates a result file compatible witht format used on MOrepo. The software basically consists of a single class "converter". After constructing an object of the class, the simple "set" methods should be used to set the appropriate entries, and finally, after setting atleast all required entries, the `createResultsFile (...)` function should be called.

1.3 Compiling

The codes were compiled using the Visual Studio 2015 compiler on a Windows 10 machine. The following flags were used: /W3 /Ox /std:[c++14|c++latest]

1.4 Example of usage

This section contains an example showing how the `converter` class can be used to generate a MOrepo compatible json file. In this example we assume the instance is named "instance1", that the contribution is "Foo et al. 2017", and that the class dataReader has the appropriate get methods.

```
    // main.cpp
  #include"converter.h"
  #include"dataReader.h" // Assume you have written this yourself
  #include<vector>
    int main(int argc, char** argv){
        try{

          std::vector<std::string> allArgs(argv, argv + argc); // retrieve arguments

            if ( argc < 2 )
          {
              throw std::runtime_error ( "Two few arguments. No input file was specified" );
          }
          else if ( argc < 3 )
          {
              outputFile = "./results.json";
          }
          else
          {
              outputFile = allArgs[2];
          }
          inputFile = allArgs[1];

          dataReader dr = dataReader ( inputFile ); //Read your own result file

          converter conv = converter(); // Create a converter object
          conv.setVersion ( "1.0" );
          conv.setInstanceName ( "instance1" );
          conv.setContributionName ( "Foo et al. 2017" );
          conv.setObjectives ( dr.getNumberOfObjectives ( ) );
          conv.setObjectiveTypes ( dr.getObjectiveTypes ( ) );
          conv.setOptimal ( dr.isItOptimal ( ) );
          conv.setCardinality ( dr.getCardinalityOfFrontier ( ) );
          conv.setPoints ( dr.getPoints ( ) , dr.getPointTypes ( ) );
          conv.setValidity ( dr.getValidityOfSolution ( ) );
          conv.createResultsFile ( outputFile );

          return 0;
      }
      catch ( std::runtime_error& re )
      {
          std::cerr << re.what () << "\n";
        }
      catch(std::exception &e)
      {
            std::cerr << "Exception: " << e.what() << "\n";
        }
      catch (...)
      {
              std::cerr << "An unexpected exception was thrown. Caught in main.\n";
        }
      return 0;
    }
```

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 converter Class Reference

Public Member Functions

- bool createResultsFile (const std::string &outputFile, const std::string &inputFile="")

  Creates a results file.
- void setVersion (const std::string &versionNumber)

  Set the version of the resultfile.
- void setInstanceName (std::string &theInstanceName)
- void setContributionName (const std::string &contName)

  Sets the name of the contribution.
- void setObjectives (int numOfObjectives)
- void setObjectiveTypes (std::vector< std::string > &objTypes)

  Sets the objective types.
- void setDirections (std::vector< std::string > &theDirections)

  Sets the directions of the objective functions.
- void setOptimal (bool itsOptimal)
- void setCardinality (int theCardinality)
- void setPoints (std::vector< std::vector< double >> &thePoints, std::vector< std::string > &pointTypes)
- void setValidity (bool isValid)

  Sets the validity of the solution to either true or false.
- void setComments (std::string &comment)
- void setCPU (double executionTime, std::string &machineSpecs)
- void setExtremeCardinality (int extremeCardinality)
- void setSupportedCardinality (int supportedCardinality)
- void setMisc (std::string &theMISC)

3.1.1 Member Function Documentation

3.1.1.1 createResultsFile()

```
bool converter::createResultsFile (
             const std::string & outputFile,
             const std::string & inputFile = "" )
```

Creates a results file.

Creates a results file with the path "outputFile". The file is only created if all the required entries has been set using the setMethods.

Parameters

outputFile	reference to a constant string. outputFile contains the path to the results file. If no file with that path exists a new file is created. If the file exists, it will be overwritten!
inputFile	reference to a constant string. inputFile contains the path to the file you want to read from (this functionality is not yet implemented).

Returns

If the putput file is succesfully created the function returns true, otherwise false (an exception/runtime error will be thrown)

3.1.1.2 setCardinality()

```
void converter::setCardinality (
             int theCardinality )  [inline]
```

Sets the cardinality of the non–dominated frontier

Parameters

theCardinality	integer specifying the number of points on the efficient frontier.

3.1.1.3 setComments()

```
void converter::setComments (
             std::string & comment )  [inline]
```

Sets the comments entry

Parameters

comment	reference to a string. Contains the comment that should be attached to the result file.

3.1.1.4 setContributionName()

```
void converter::setContributionName (
             const std::string & contName )  [inline]
```

Sets the name of the contribution.

Sets the contribution name. It should be a string with the name of the contribution in which the instances and results have been published.

Parameters

constName	const reference to a string. Contains the name of the constribution, e.g. "Pedersen08".

3.1.1.5 setCPU()

```
void converter::setCPU (
            double executionTime,
            std::string & machineSpecs )
```

Sets the CPU information along with the specs of the machine the experiments was carried out on.

Parameters

executionTime	double containing the number of seconds used to compute the efficient frontier
machineSpecs	reference to a string containing the specifics of the machine used to carry out the experiments, e.g. "Intel Core i7-4785T 2.2 GHz, 16 GB RAM, Linux Ubuntu 64bit"

3.1.1.6 setDirections()

```
void converter::setDirections (
            std::vector< std::string > & theDirections )
```

Sets the directions of the objective functions.

Sets the directions of the objective functions. The directions can either be "min" or "max".

Parameters

theDirections	reference to a vector of strings. If forexample there are three objective functions where the two first are of the minimization-kind and the last is a maximization, we should specify a vector { "min" , "min" , "max" } as the function argument.

3.1.1.7 setExtremeCardinality()

```
void converter::setExtremeCardinality (
            int extremeCardinality )  [inline]
```

Sets the cardinality of the set of extreme supported non–dominated solutions

Parameters

extremeCardinality	integer containing the number of extreme supported non–dominated solutions.

3.1.1.8 setInstanceName()

```
void converter::setInstanceName (
            std::string & theInstanceName )  [inline]
```

Set the name of the instance for which the result file contains information

Parameters

theInstanceName	reference to a string containing the name of the instance for which the results are for.

3.1.1.9 setMisc()

```
void converter::setMisc (
            std::string & theMISC )  [inline]
```

Sets the misc entry

Parameters

theMISC	reference to a string containing the misc that should be attached to the result file

Note

This entry may be used as you like. It could e.g. contain an object with more detailed entries about the experiment.

3.1.1.10 setObjectives()

```
void converter::setObjectives (
            int numOfObjectives )  [inline]
```

Sets the number of objectives

Parameters

numOfObjectives	integer specifiying the number of objective functions of the multiobjective optimization problem.

3.1.1.11 setObjectiveTypes()

```
void converter::setObjectiveTypes (
            std::vector< std::string > & objTypes )
```

Sets the objective types.

Sets the objective types to either int, float, or null (if unknown).

Parameters

objTypes	vector of strings containing the type of each objective. That is if the i'th objective is integral, then objType[i] = "int"

Note

The function setObjectives must be called before setObjectiveTypes. Otherwise a runtime error is thrown.

3.1.1.12 setOptimal()

```
void converter::setOptimal (
            bool itsOptimal )  [inline]
```

Specifies whether the solution is an optimal solution to the specific instance or not.

Parameters

itsOptimal	boolean. If itsOptimal = true, it is assumed that the solutions is optimal solution, and if itsOptimal = false, it has not been verified optimal, or it is known to be suboptimal

3.1.1.13 setPoints()

```
void converter::setPoints (
            std::vector< std::vector< double >> & thePoints,
            std::vector< std::string > & pointTypes )
```

Sets the points and the point types

Parameters

thePoints	reference to a vector of vectors of doubles. thePoints[i] contains the i'th point on the frontier and thePoints[i][j] contains the j'th entry of the i'th non–dominated point.
pointTypes	reference to a vector of strings. Contains a specification of the type of each point. type can be either extreme supported ("se"), non-extreme supported ("sne"), supported (my be extreme or non–extreme) ("s"), unsuported ("un") or if this information is unknown ("null").

3.1.1.14 setSupportedCardinality()

```
void converter::setSupportedCardinality (
             int supportedCardinality )  [inline]
```

Sets the cardinality of the set of supported non–dominated solutions

Parameters

supportedCardinality	integer containing the number of supported non–dominated solutions.

3.1.1.15 setValidity()

```
void converter::setValidity (
             bool isValid )  [inline]
```

Sets the validity of the solution to either true or false.

Sets the validity of the solution to either true or false. If isValid is false, the solution might be in conflict with another solution on MOrepo. This will be sorted out eventually

Parameters

isValid	boolean. If true, the solution is not in conflict with other known solutions. If false, it is in conflict with a known solution.

3.1.1.16 setVersion()

```
void converter::setVersion (
             const std::string & versionNumber )  [inline]
```

Set the version of the resultfile.

Sets the version fo the results file using the provided string

Parameters

versionNumber	reference to a constant string. If the version is 5.4 the input should be a string "5.4"

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

- converter.h

- converter.cpp