A solver for the bi--objective cost--bottleneck location problem

Generated by Doxygen 1.8.6

Thu Jan 28 2016 14:49:40

Contents

Chapter 1

An introduction to the rdDat class

Author

Sune Lauth Gadegaard

Version

1.0.0

1.1 License

1.2 Description

The two classes contained in ths manual (BOCBLPsolver and rdDat) is used to solve the bi–objective cost–bottleneck location problem for different kinds of location problems. The class rdDat implements a data reader for different kinds of discrete facility location problems while the class BOCBLPsolver implements the actual solution algorithm.

1.3 Compiling

The codes were compiled using the GNU GCC compiler on a Linux Ubuntu 14.04 machine. The following flags were used: -Wall -O3 -std=c++11 -DIL_STD. The Code::blocks IDE was used as well.

1.4 Change log for rdDat.h and rdDat.cpp

FILE:	rdDat.h, rdDat.cpp, BOCBLPsolver.h and BOCBLPsolver.cpp		
Version:	1.0.0		
- -			
CHANGE LOG:	DATE	VER.-NO.	CHANGES MADE
	2016–01–28	1.0.0	First implementation

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 BOCBLPsolver Class Reference

`#include <BOCBLPsolver.h>`

Public Member Functions

- BOCBLPsolver (std::string DataFile, int TheProblemType)
- ∼BOCBLPsolver ()
- void run ()

Public Attributes

- unsigned long NrNodes

  Number of branching nodes. Only used when using the lexicographic branch and bound.
- double MedianObjective

  Holds the current best value of the min-cost objective when using lexicographic branch and bound.
- double CenterObjective

  Holds the value of the bottleneck objective of the current best solution to the min-cost problem when using lexicographic branch and bound.
- std::string outputfile

  File used to write info of the solution process to. Overwrites content if file already exists.
- std::string SummaryFile

  File used to a sumary data to. Appends to the file if it already exists.
- int InstanceNumber

  Holds the id-number of the current data instance. It is used to name the outputfile as well as the corresponding line in the summaryfile.
- int CostMethod

  Holds the id-number of the current cost stucture. It is used to name the outputfile as well as the corresponding line in the summaryfile.

Cplex

This section contains all the cplex gear needed for the algorithm to run.

- IloEnv env

  The Ilo environment used throughout the lifetime of the object.
- IloModel model

  The IloModel used to build the model.
- IloCplex cplex

The IloCplex environmnet used to solve the model.

- IloNumVarArray y

    The location variables.
- IloVarMatrix x

    The assignment variables.
- IloObjective OBJ

Data

This section contains all the data for describing the location problems.

- int n

    Number of facilities.
- int m

    Number of customrs.
- int $*$ d

    Demands.
- int $*$ s

    Capacities.
- int $*$ f

    Fixed opening cost.
- double $**$ c

    Travel cost.
- int $**$ t

    Travel time.
- int p

    Number of open facilies. Used when the problem is of the p-median type.
- int TD

Private Member Functions

- void BuildModel ()

    Contains the total execution time of the whole algorithm.
- int EvaluateCenter ()

Private Attributes

- int ProblemType

    0 = p-median, 1 = CFLP, 2 = UFLP, 3 = SSCFLP
- std::vector< std::pair< double,
  int > > Frontier
- std::vector< std::pair< double,
  int > > WeakPoints

    Vector of pairs containing the frontier.
- std::vector< double > Times

    Vector of pairs containing the weakly domoniated points and the frontier (Superset of Frontier)
- int NumOfPP

    Vector containing all the execution time of each problem Times[t] holds the time of the problem resulting in point WeakPoints[t].
- int NumOfWP

    The size of Fontier. That is NumOfPP = Frontier.size ( )
- double TotalTime

    The size of WeakPoints. That is NumOfWP = WeakPoints.size ( )

4.1.1 Constructor & Destructor Documentation

4.1.1.1 BOCBLPsolver::BOCBLPsolver (std::string DataFile, int TheProblemType)

The constructor of the class. Takes a data–file and reads the data using rdDat classed defined in rdDat.h and an integer specifying the problem type.

Parameters

DataFile	string. Must contain the address of a valid data file
TheProblemType	integer. Integer between 0 and 3 specifying the class of location problem you want to solve. 0 = p-median, 1 = CFLP, 2 = UFLP, 3 = SSCFLP

4.1.1.2 BOCBLPsolver::∼BOCBLPsolver ( )

The destructor of the class. Releases the memory allocated to cplex and internal data–structures.

4.1.2 Member Function Documentation

4.1.2.1 void BOCBLPsolver::BuildModel ( ) [private]

Contains the total execution time of the whole algorithm.

Build the location model specified by the integer ProblemType (can be either 0, 1, 2, or 3)

4.1.2.2 int BOCBLPsolver::EvaluateCenter ( ) [private]

Evaluates the bottleneck cost of the current solution.

4.1.2.3 void BOCBLPsolver::run ( )

Function working as the API for the user.

4.1.3 Member Data Documentation

4.1.3.1 double∗∗ BOCBLPsolver::c

Travel cost.

4.1.3.2 double BOCBLPsolver::CenterObjective

Holds the value of the bottleneck objective of the current best solution to the min-cost problem when using lexicographic branch and bound.

4.1.3.3 int BOCBLPsolver::CostMethod

Holds the id-number of the current cost stucture. It is used to name the outputfile as well as the corresponding line in the summaryfile.

4.1.3.4 IloCplex BOCBLPsolver::cplex

The IloCplex environmnet used to solve the model.

4.1.3.5 int∗ BOCBLPsolver::d

Demands.

4.1.3.6 IloEnv BOCBLPsolver::env

The Ilo environment used throughout the lifetime of the object.

4.1.3.7 int* BOCBLPsolver::f

Fixed opening cost.

4.1.3.8 std::vector<std::pair<double,int> > BOCBLPsolver::Frontier [private]

4.1.3.9 int BOCBLPsolver::InstanceNumber

Holds the id-number of the current data instance. It is used to name the outputfile as well as the corresponding line in the summaryfile.

4.1.3.10 int BOCBLPsolver::m

Number of customrs.

4.1.3.11 double BOCBLPsolver::MedianObjective

Holds the current best value of the min-cost objective when using lexicographic branch and bound.

4.1.3.12 IloModel BOCBLPsolver::model

The IloModel used to build the model.

4.1.3.13 int BOCBLPsolver::n

Number of facilities.

4.1.3.14 unsigned long BOCBLPsolver::NrNodes

Number of branching nodes. Only used when using the lexicographic branch and bound.

4.1.3.15 int BOCBLPsolver::NumOfPP [private]

Vector containing all the execution time of each problem Times[t] holds the time of the problem resulting in point WeakPoints[t].

4.1.3.16 int BOCBLPsolver::NumOfWP [private]

The size of Fontier. That is NumOfPP = Frontier.size ( )

4.1.3.17 IloObjective BOCBLPsolver::OBJ

The IloObjective extractable used to hold the objective function.

4.1.3.18 std::string BOCBLPsolver::outputfile

File used to write info of the solution process to. Overwrites content if file already exists.

4.1.3.19 int BOCBLPsolver::p

Number of open facilies. Used when the problem is of the p-median type.

4.1.3.20 int BOCBLPsolver::ProblemType [private]

0 = p-median, 1 = CFLP, 2 = UFLP, 3 = SSCFLP

4.1.3.21 int* BOCBLPsolver::s

Capacities.

4.1.3.22 std::string BOCBLPsolver::SummaryFile

File used to a sumary data to. Appends to the file if it already exists.

4.1.3.23 int** BOCBLPsolver::t

Travel time.

4.1.3.24 int BOCBLPsolver::TD

Total demand. That is TD = {j J} d_j

4.1.3.25 std::vector<double> BOCBLPsolver::Times [private]

Vector of pairs containing the weakly domoniated points and the frontier (Superset of Frontier)

4.1.3.26 double BOCBLPsolver::TotalTime [private]

The size of WeakPoints. That is NumOfWP = WeakPoints.size ( )

4.1.3.27 std::vector<std::pair<double,int> > BOCBLPsolver::WeakPoints [private]

Vector of pairs containing the frontier.

4.1.3.28 **IloVarMatrix** BOCBLPsolver::x

The assignment variables.

4.1.3.29 IloNumVarArray BOCBLPsolver::y

The location variables.

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

- BOCBLPsolver.h
- BOCBLPsolver.cpp

4.2 rdDat Class Reference

```
#include <rdDat.h>
```

Public Member Functions

- rdDat (std::string Filename, int ProblemType)
- ∼rdDat ()
- void rdPmed (std::string DataFile)
- void rdUFLP (std::string DataFile)
- void rdCFLP (std::string DataFile)
- void rdSSCFLP (std::string DataFile)
- int getNumFac ()
- int getNumCust ()
- int getP ()
- int getD (int j)
- int getS (int i)
- int getF (int i)
- int getC (int i, int j)
- int getT (int i, int j)
- int * getAllD ()
- int * getAllS ()
- int * getAllF ()
- double ** getAllC ()
- int ** getAllT ()

Private Attributes

- int n

  Number of facilities.
- int m

  Number of customers.
- int p

  Number of open faiclities in a solution to the p-median problem.
- int * d

  Demands. d[j] demand of customer.
- int * s

  Capacities. s[i] capacity of facility i.
- int * f

  Fixed opening cost. f[i] cost of opening facility i.
- double ** c

  Assingment cost. c[i][j] is the cost of supplying all of customer j's demand from facility i.
- int ** t

  Travel time from facility i to customer j.
- int TheProblemType

  Integer indicating which problem type is in qustion.

4.2.1 Constructor & Destructor Documentation

4.2.1.1 rdDat::rdDat (std::string Filename, int ProblemType)

Constructor of the rdDat class.

Parameters

Filename	String. Contains the path to a data file of appropriate format

4.2.1.2 rdDat::∼rdDat ( )

Destructor of the class. Cleans up after the us.

4.2.2 Member Function Documentation

4.2.2.1 double∗∗ rdDat::getAllC ( ) [inline]

Returns a pointer to a pointer to the an integer array containing the assignment costs

4.2.2.2 int∗ rdDat::getAllD ( ) [inline]

Returns a pointer to the first element in the integer array containing the demands

4.2.2.3 int∗ rdDat::getAllF ( ) [inline]

Returns a pointer to the first element in the integer array containing the fixed opening costs

4.2.2.4 int∗ rdDat::getAllS ( ) [inline]

Returns a pointer to the first element in the integer array containing the capacities

4.2.2.5 int∗∗ rdDat::getAllT ( ) [inline]

Returns a pointer to a pointer to the an integer array containing the travel times

4.2.2.6 int rdDat::getC (int i, int j) [inline]

Returns the assignment cost of the the facility–customer pair (i,j)

Parameters

i	integer. Index of the facility
j	integer. Index of the customer

4.2.2.7 int rdDat::getD (int j) [inline]

Returns the demand of customer j

Parameters

j	integer. Index of the customer who's demand you want

4.2.2.8 int rdDat::getF (int i) [inline]

Returns the fixed opening cost of facility i

Parameters

ji	integer. Index of the facility who's fixed cost you want

4.2.2.9 int rdDat::getNumCust ( ) [inline]

Returns the number of customers

4.2.2.10 int rdDat::getNumFac ( ) [inline]

Returns the number of facilities

4.2.2.11 int rdDat::getP ( ) [inline]

Returns the number facilities which must be open in a p-median problem

4.2.2.12 int rdDat::getS (int i) [inline]

Returns the capacity of facility i

Parameters

i	integer. Index of the facility who's capacity you want

4.2.2.13 int rdDat::getT (int i, int j) [inline]

Returns the travel time between facility i and customer j

Parameters

i	integer. Index of the facility
j	integer. Index of the customer

4.2.2.14 void rdDat::rdCFLP (std::string DataFile)

Reads the data of a capacitated facility location problem problem

4.2.2.15 void rdDat::rdPmed (std::string DataFile)

Reads the data of a p-median problem

4.2.2.16 void rdDat::rdSSCFLP (std::string DataFile)

Reads the data of a single source capacitated facility location problem problem

4.2.2.17 void rdDat::rdUFLP (std::string DataFile)

Reads the data of an uncapacitated facility location problem problem

4.2.3 Member Data Documentation

4.2.3.1 double** rdDat::c [private]

Assingment cost. c[i][j] is the cost of supplying all of customer j's demand from facility i.

4.2.3.2 int* rdDat::d [private]

Demands. d[j] demand of customer.

4.2.3.3 int* rdDat::f [private]

Fixed opening cost. f[i] cost of opening facility i.

4.2.3.4 int rdDat::m [private]

Number of customers.

4.2.3.5 int rdDat::n [private]

Number of facilities.

4.2.3.6 int rdDat::p [private]

Number of open faiclities in a solution to the p-median problem.

4.2.3.7 int* rdDat::s [private]

Capacities. s[i] capacity of facility i.

4.2.3.8 int** rdDat::t [private]

Travel time from facility i to customer j.

4.2.3.9 int rdDat::TheProblemType [private]

Integer indicating which problem type is in qustion.

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

- rdDat.h
- rdDat.cpp

Chapter 5

File Documentation

5.1 BOCBLPsolver.cpp File Reference

```
#include "PureCplex.h"
```

Functions

- ILOBRANCHCALLBACK1 (lexibrancher, BOCBLPsolver &, lm)
- ILOINCUMBENTCALLBACK1 (IncUpdate, BOCBLPsolver &, lm)
- ILONODECALLBACK1 (NodeCount, PureCplex &, lm)
- ILOMIPCALLBACK1 (Terminator, BOCBLPsolver &, lm)

Variables

- const bool DoLexiBrancing = false
- const double tol = 1E-6
- const bool fToZero = false

5.1.1 Function Documentation

5.1.1.1 ILOBRANCHCALLBACK1 (lexibrancher , **BOCBLPsolver** & , lm)

5.1.1.2 ILOINCUMBENTCALLBACK1 (IncUpdate , **BOCBLPsolver** & , lm)

5.1.1.3 ILOMIPCALLBACK1 (Terminator , **BOCBLPsolver** & , lm)

5.1.1.4 ILONODECALLBACK1 (NodeCount , PureCplex & , lm)

5.1.2 Variable Documentation

5.1.2.1 const bool DoLexiBrancing = false

5.1.2.2 const bool fToZero = false

5.1.2.3 const double tol = 1E-6

5.2 BOCBLPsolver.h File Reference

```
#include <ilcplex/ilocplex.h>
#include <exception>
#include <stdexcept>
#include <vector>
#include <algorithm>
#include <random>
#include <chrono>
#include <ratio>
#include <fstream>
#include "rdDat.h"
```

Classes

- class BOCBLPsolver

Typedefs

- typedef IloArray< IloNumVarArray > IloVarMatrix

  An IloArray of IloNumVarArrays.
- typedef std::chrono::high_resolution_clock CPUclock

5.2.1 Typedef Documentation

5.2.1.1 typedef std::chrono::high_resolution_clock **CPUclock**

5.2.1.2 typedef IloArray<IloNumVarArray> **IloVarMatrix**

An IloArray of IloNumVarArrays.

5.3 rdDat.cpp File Reference

```
#include "rdDat.h"
```

5.4 rdDat.h File Reference

```
#include <random>
#include <exception>
#include <stdexcept>
#include <iostream>
#include <fstream>
#include <vector>
#include <sstream>
#include <string>
```

Classes

- class rdDat