BiUFLv2012

1.1

Generated by Doxygen 1.7.6.1

Fri Aug 31 2012 09:38:17

# **Contents**

1	Mod	ule Inde	ex		1
	1.1	Module	es		1
2	Clas	s Index			3
	2.1	Class I	List		3
3	File	Index			5
	3.1	File Lis	st		5
4	Mod	ule Doc	cumentatio	on	7
	4.1	Method	ds of Pavin	ng	7
		4.1.1	Function	Documentation	7
			4.1.1.1	addChildren	7
			4.1.1.2	boxFiltering	8
			4.1.1.3	createBox	8
			4.1.1.4	filter	8
			4.1.1.5	recomposition	9
			4.1.1.6	weightedSumOneStep	9
	4.2	Method	ds of Gene	erating	0
		4.2.1	Function	Documentation	0
			4.2.1.1	filterListSolution	0
			4.2.1.2	runLabelSetting	0
	4.3	Others	Methods		1
		4.3.1	Function	Documentation	1
			4.3.1.1	computeCorrelation	1
			4.3.1.2	time ms Diff	1

ii CONTENTS

			4.3.1.3	time_s_Diff	1
	4.4	Global	Variables		3
		4.4.1	Variable I	Documentation	3
			4.4.1.1	modeExport	3
			4.4.1.2	modeVerbose	3
	4.5	Main .			4
		4.5.1	Function	Documentation	4
			4.5.1.1	main	4
5	Clas	s Docui	mentation	15	5
•	5.1		ass Refere		
	0.1	5.1.1		Description	
		5.1.2		tor & Destructor Documentation	
		· · · · ·	5.1.2.1	Box	
			5.1.2.2	Box	
			5.1.2.3	Box	8
			5.1.2.4	~Box	8
		5.1.3	Member I	Function Documentation	8
			5.1.3.1	computeBox	8
			5.1.3.2	getData	8
			5.1.3.3	getHasMoreStepWS	9
			5.1.3.4	getHasNeighbor	9
			5.1.3.5	getld	9
			5.1.3.6	getMaxZ1	9
			5.1.3.7	getMaxZ2	9
			5.1.3.8	getMinZ1	0
			5.1.3.9	getMinZ2	0
			5.1.3.10	getnbCustomerNotAffected	0
			5.1.3.11	getNbFacilityOpen	0
			5.1.3.12	getOriginZ1	0
			5.1.3.13	getOriginZ2	0
			5.1.3.14	isAssigned	1
			5.1.3.15	isOpened	1
			5.1.3.16	openFacility	1

CONTENTS iii

		5.1.3.17	print
		5.1.3.18	setHasMoreStepWS 21
		5.1.3.19	setHasNeighbor
		5.1.3.20	setId
		5.1.3.21	setMaxZ1 22
		5.1.3.22	setMaxZ2
		5.1.3.23	setMinZ1
		5.1.3.24	setMinZ2
	5.1.4	Friends A	and Related Function Documentation
		5.1.4.1	filterDominatedBoxes
		5.1.4.2	isDominatedBetweenOrigins
		5.1.4.3	isDominatedBetweenTwoBoxes 23
		5.1.4.4	isDominatedByItsBox
		5.1.4.5	isDominatedByItsOrigin
	5.1.5	Member	Data Documentation
		5.1.5.1	data
		5.1.5.2	facility
		5.1.5.3	hasMoreStepWS
		5.1.5.4	hasNeighbor
		5.1.5.5	id
		5.1.5.6	isAssigned
		5.1.5.7	maxZ1
		5.1.5.8	maxZ2
		5.1.5.9	minZ1
		5.1.5.10	minZ2
		5.1.5.11	nbCustomerNotAffected
		5.1.5.12	originZ1
		5.1.5.13	originZ2
5.2	Custon	ner Class I	Reference
	5.2.1	Detailed	Description
	5.2.2	Construc	tor & Destructor Documentation
		5.2.2.1	Customer
	5.2.3	Member	Function Documentation
		5.2.3.1	getCoordX

iv CONTENTS

		5.2.3.2	getCoordY	27
	5.2.4	Friends A	and Related Function Documentation	27
		5.2.4.1	operator<<	27
	5.2.5	Member	Data Documentation	27
		5.2.5.1	$coordX\_{}\ldots\ldots\ldots\ldots\ldots\ldots$	28
		5.2.5.2	$coordY\_\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots$	28
5.3	Data C	lass Refer	ence	28
	5.3.1	Detailed	Description	29
	5.3.2	Member	Typedef Documentation	29
		5.3.2.1	ListCustomers	29
		5.3.2.2	ListFacilities	29
	5.3.3	Construc	tor & Destructor Documentation	30
		5.3.3.1	Data	30
		5.3.3.2	~Data	30
	5.3.4	Member	Function Documentation	30
		5.3.4.1	addCustomer	30
		5.3.4.2	addFacility	30
		5.3.4.3	getAllocationObj1Cost	30
		5.3.4.4	getAllocationObj2Cost	31
		5.3.4.5	getFacility	31
		5.3.4.6	getFileName	31
		5.3.4.7	getnbCustomer	31
		5.3.4.8	getnbFacility	32
		5.3.4.9	setAllocationObj1Cost	32
		5.3.4.10	setAllocationObj2Cost	32
		5.3.4.11	setFileName	32
	5.3.5	Member	Data Documentation	32
		5.3.5.1	allocationObj1Cost	32
		5.3.5.2	allocationObj2Cost	33
		5.3.5.3	customerList	33
		5.3.5.4	facilityList	33
		5.3.5.5	fileName	33
5.4	Facility	Class Ref	erence	33
	5.4.1	Detailed	Description	34

CONTENTS

	5.4.2	Construc	tor & Destructor Documentation	4
		5.4.2.1	Facility	4
	5.4.3	Member	Function Documentation	4
		5.4.3.1	getCoordX	4
		5.4.3.2	getCoordY	5
		5.4.3.3	getLocationObj1Cost	5
		5.4.3.4	getLocationObj2Cost	5
		5.4.3.5	setLocationObj1Cost	5
		5.4.3.6	setLocationObj2Cost	5
	5.4.4	Friends A	and Related Function Documentation	5
		5.4.4.1	operator<< 3	6
	5.4.5	Member	Data Documentation	6
		5.4.5.1	coordX	6
		5.4.5.2	coordY	6
		5.4.5.3	locationObj1Cost	6
		5.4.5.4	locationObj2Cost	6
5.5	LabelS	etting Clas	ss Reference	6
	5.5.1	Detailed	Description	7
	5.5.1 5.5.2		Description	
			•	7
		Construc	tor & Destructor Documentation	7 7
		Construction 5.5.2.1 5.5.2.2	tor & Destructor Documentation	7 7 8
	5.5.2	Construction 5.5.2.1 5.5.2.2	tor & Destructor Documentation	7 7 8
	5.5.2	5.5.2.1 5.5.2.2 Member	tor & Destructor Documentation       3         LabelSetting       3         ~LabelSetting       3         Function Documentation       3	7 7 8 8
	5.5.2	Construct 5.5.2.1 5.5.2.2 Member 5.5.3.1	tor & Destructor Documentation       3         LabelSetting       3         ~LabelSetting       3         Function Documentation       3         compute       3	7 8 8 8
	5.5.2	Construct 5.5.2.1 5.5.2.2 Member 5.5.3.1 5.5.3.2	tor & Destructor Documentation       3         LabelSetting       3         ~LabelSetting       3         Function Documentation       3         compute       3         computeNode       3	7 7 8 8 8 8
	5.5.2	Construct 5.5.2.1 5.5.2.2 Member 1 5.5.3.1 5.5.3.2 5.5.3.3 5.5.3.4	tor & Destructor Documentation       3         LabelSetting       3         ~LabelSetting       3         Function Documentation       3         compute       3         computeNode       3         getRank       3	7 7 8 8 8 8
	5.5.2	Construct 5.5.2.1 5.5.2.2 Member 1 5.5.3.1 5.5.3.2 5.5.3.3 5.5.3.4	tor & Destructor Documentation       3         LabelSetting       3         ~LabelSetting       3         Function Documentation       3         compute       3         computeNode       3         getRank       3         print       3	7 7 8 8 8 8 8
	5.5.2	Construct 5.5.2.1 5.5.2.2 Member 5.5.3.1 5.5.3.2 5.5.3.3 5.5.3.4 Member	tor & Destructor Documentation       3         LabelSetting       3         ~LabelSetting       3         Function Documentation       3         compute       3         computeNode       3         getRank       3         print       3         Data Documentation       3	7 7 8 8 8 8 8 8
	5.5.2	Construct 5.5.2.1 5.5.2.2 Member 5.5.3.1 5.5.3.2 5.5.3.3 5.5.3.4 Member 6 5.5.4.1	tor & Destructor Documentation       3         LabelSetting       3         ~LabelSetting       3         Function Documentation       3         compute       3         computeNode       3         getRank       3         print       3         Data Documentation       3         boundZ1_       3	7 7 8 8 8 8 8 8 8
	5.5.2	Construct 5.5.2.1 5.5.2.2 Member 1 5.5.3.1 5.5.3.2 5.5.3.3 5.5.3.4 Member 1 5.5.4.1 5.5.4.2	tor & Destructor Documentation       3         LabelSetting       3         ~LabelSetting       3         Function Documentation       3         compute       3         computeNode       3         getRank       3         print       3         Data Documentation       3         boundZ1_       3         boundZ2_       3	7 7 8 8 8 8 8 8 8 8 8
	5.5.2	Construct 5.5.2.1 5.5.2.2 Member 1 5.5.3.1 5.5.3.2 5.5.3.3 5.5.3.4 Member 1 5.5.4.1 5.5.4.2 5.5.4.3	tor & Destructor Documentation       3         LabelSetting       3         ~LabelSetting       3         Function Documentation       3         compute       3         computeNode       3         getRank       3         print       3         Data Documentation       3         boundZ1_       3         boundZ2_       3         data_       3	7 7 8 8 8 8 8 8 8 8 9
5.6	5.5.2 5.5.3 5.5.4	Construct 5.5.2.1 5.5.2.2 Member 1 5.5.3.1 5.5.3.2 5.5.3.3 5.5.3.4 Member 1 5.5.4.1 5.5.4.2 5.5.4.3 5.5.4.4 5.5.4.5	tor & Destructor Documentation       3         LabelSetting       3         ~LabelSetting       3         Function Documentation       3         compute       3         computeNode       3         getRank       3         print       3         Data Documentation       3         boundZ1_       3         boundZ2_       3         data_       3         nbRank_       3	7 7 8 8 8 8 8 8 8 8 8 9 9

vi CONTENTS

	5.6.2	Construc	tor & Destructor Documentation
		5.6.2.1	Node
		5.6.2.2	Node
		5.6.2.3	~Node
	5.6.3	Member	Function Documentation
		5.6.3.1	clearLabels
		5.6.3.2	getCostToEnterZ1 40
		5.6.3.3	getCostToEnterZ2
		5.6.3.4	getSize
		5.6.3.5	print
		5.6.3.6	setSize
		5.6.3.7	setValues
	5.6.4	Member	Data Documentation
		5.6.4.1	costToEnterZ1
		5.6.4.2	costToEnterZ2
		5.6.4.3	labels
		5.6.4.4	size
5.7	Parser	Class Ref	erence
	5.7.1	Detailed	Description
	5.7.2	Member	Function Documentation
			Tunction Documentation
		5.7.2.1	ignoreChar
		5.7.2.1	ignoreChar
5.8	Solutio	5.7.2.1 5.7.2.2 5.7.2.3	ignoreChar       43         ignoreLine       43
5.8	Solutio 5.8.1	5.7.2.1 5.7.2.2 5.7.2.3 n Class R	ignoreChar       43         ignoreLine       43         Parsing       43
5.8		5.7.2.1 5.7.2.2 5.7.2.3 n Class R Detailed	ignoreChar       43         ignoreLine       43         Parsing       43         eference       44
5.8	5.8.1	5.7.2.1 5.7.2.2 5.7.2.3 n Class R Detailed	ignoreChar       43         ignoreLine       43         Parsing       43         eference       44         Description       45
5.8	5.8.1	5.7.2.1 5.7.2.2 5.7.2.3 n Class R Detailed Construc	ignoreChar       43         ignoreLine       43         Parsing       43         eference       44         Description       45         etor & Destructor Documentation       45
5.8	5.8.1	5.7.2.1 5.7.2.2 5.7.2.3 n Class R Detailed Construct 5.8.2.1	ignoreChar       43         ignoreLine       43         Parsing       43         eference       44         Description       45         stor & Destructor Documentation       45         Solution       45
5.8	5.8.1	5.7.2.1 5.7.2.2 5.7.2.3 n Class R Detailed Construct 5.8.2.1 5.8.2.2 5.8.2.3	ignoreChar       43         ignoreLine       43         Parsing       43         eference       44         Description       45         ctor & Destructor Documentation       45         Solution       45         Solution       45         Solution       45
5.8	5.8.1 5.8.2	5.7.2.1 5.7.2.2 5.7.2.3 n Class R Detailed Construct 5.8.2.1 5.8.2.2 5.8.2.3	ignoreChar       43         ignoreLine       43         Parsing       43         eference       44         Description       45         stor & Destructor Documentation       45         Solution       45         Solution       45         Solution       45         ~Solution       45         ~Solution       45
5.8	5.8.1 5.8.2	5.7.2.1 5.7.2.2 5.7.2.3 n Class R Detailed Construct 5.8.2.1 5.8.2.2 5.8.2.3 Member	ignoreChar       43         ignoreLine       43         Parsing       43         eference       44         Description       45         stor & Destructor Documentation       45         Solution       45         Solution       45         Solution       45         Function Documentation       45
5.8	5.8.1 5.8.2	5.7.2.1 5.7.2.2 5.7.2.3 n Class R Detailed Construct 5.8.2.1 5.8.2.2 5.8.2.3 Member 5.8.3.1	ignoreChar       43         ignoreLine       43         Parsing       43         eference       44         Description       45         stor & Destructor Documentation       45         Solution       45         Solution       45         Solution       45         Function Documentation       45         getObj1       45

CONTENTS vii

		5.8.4	Friends A	and Related Function Documentation	46
			5.8.4.1	operator<	46
		5.8.5	Member I	Data Documentation	46
			5.8.5.1	obj1	46
			5.8.5.2	obj2	46
	5.9	ToFile (	Class Refe	erence	47
		5.9.1	Detailed I	Description	47
		5.9.2	Member	Function Documentation	47
			5.9.2.1	removeFiles	47
			5.9.2.2	saveCorrelation	47
			5.9.2.3	saveFilteringBoxes	48
			5.9.2.4	saveInitialBoxes	48
			5.9.2.5	saveReconstructionBoxes	48
			5.9.2.6	saveYN	48
6	File I	Docume	entation		51
_	6.1	Documentation /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-		Computing/syn/bjuflp2012/trunk/wolsevaward/1.1/src/-	
				erence	51
		6.1.1	Function	Documentation	51
			6.1.1.1	filterDominatedBoxes	51
			6.1.1.2	isDominatedBetweenOrigins	51
			6.1.1.3	isDominatedBetweenTwoBoxes	51
			6.1.1.4	isDominatedByItsBox	51
			6.1.1.5	isDominatedByItsOrigin	51
	6.2			Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/erence	52
		6.2.1	Detailed I	Description	52
	6.3			Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-e Reference	52
		6.3.1	Function	Documentation	53
			6.3.1.1	operator<<	53
	6.4			Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/e Reference	53
		6.4.1	• • •	Description	53
	6.5			Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-	55
	0.0			erence	54

viii CONTENTS

6.6		axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-pp File Reference	54
	6.6.1	Detailed Description	54
6.7		axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/cpp File Reference	55
	6.7.1	Function Documentation	55
		6.7.1.1 operator<<	55
6.8		axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-hpp File Reference	55
	6.8.1	Detailed Description	55
6.9		axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-	56
6.10		axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-	56
	6.10.1	Detailed Description	57
6.11		axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/etting.cpp File Reference	58
6.12		axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/etting.hpp File Reference	58
	6.12.1	Detailed Description	58
6.13		axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-pp File Reference	59
	6.13.1	Detailed Description	60
6.14		axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-pp File Reference	60
6.15		axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-pp File Reference	60
	6.15.1	Detailed Description	61
6.16		axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-cpp File Reference	61
6.17		axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-hpp File Reference	61
	6.17.1	Detailed Description	62
6.18		axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-n.cpp File Reference	62
	6.18.1	Function Documentation	62
		6.18.1.1 operator<	62

CONTENTS ix

6.19	/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-	
	Solution.hpp File Reference	62
	6.19.1 Detailed Description	63
6.20	/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-ToFile.cpp File Reference	63
6.21	/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-ToFile.hpp File Reference	63
	6.21.1 Detailed Description	64

# **Chapter 1**

# **Module Index**

# 1.1 Modules

Here	is a	list c	nf all	modu	ıles

Methods of Pavin	g.																	7
Methods of Gene	rat	ing	J															10
Others Methods																		11
Global Variables																		13
Main																		14

2 Module Index

# **Chapter 2**

# **Class Index**

# 2.1 Class List

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

Rox		
	Class to represent a Box	15
Custome	r	
	Class to represent a Customer	26
Data		
	Class to represent a Data	28
Facility		
	Class to represent a Facility	33
LabelSet		
	Class to represent a LabelSetting	36
Node	Class to represent a N. e.d.	20
Parser	Class to represent a Node	39
raisei	Class to represent a Parser	42
Solution	Olass to represent a farser	72
Colution	Class to represent a Solution	44
ToFile		
	Class to represent a ToFile	47

4 Class Index

# **Chapter 3**

# File Index

# 3.1 File List

Here is a list of all files w	vith brief descrip	otions
-------------------------------	--------------------	--------

/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Box	
cpp	51
/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Box	
hpp	
Class of the Box	52
/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-	
Customer.cpp	52
/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-	
Customer.hpp	
Class of the Customer	53
/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Data	
cpp	54
/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Data	
hpp	
Class of the Data	54
/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-	
Facility.cpp	55
/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-	
Facility.hpp	
Class of the Facility	55
/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-	
Functions.cpp	56
/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-	
Functions.hpp	
A set of functions usefull for our software	56
/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Label-	
Setting.cpp	58

6 File Index

/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Label- Setting.hpp	
Class of the Box	58
/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Main	
cpp Main of the software	59
/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Node	
cpp	60
/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Node hpp	
Class of the Node	60
/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-	
Parser.cpp	61
/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-	
Parser.hpp	
Class of the Parser	61
/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-	
Solution.cpp	62
/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-	
Solution.hpp	
Class of the Solution	62
/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/ToFile	
cpp	63
/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/ToFile	
Class of the ToFile	63

# **Chapter 4**

# **Module Documentation**

# 4.1 Methods of Paving

#### **Functions**

long int createBox (vector< Box \* > &vectorBox, Data &data)

This method computes all the initial Boxes of our algorithm.

void addChildren (Box \*boxMother, vector< Box \* > &vBox)

This method adds children of a Box into a vector of Boxes.

- void filter (vector<  $\mbox{Box} * > \mbox{\&vectorBox}, \mbox{long int &nbToCompute, long int &nbWithNeighbor)}$ 

This method filters all the Boxes of a vector of Box.

 void boxFiltering (vector< Box \* > &vectorBox, Data &data, long int &nbTo-Compute, long int &nbWithNeighbor)

This method splits a Box into two Boxes.

 void recomposition (vector< Box \* > &vectorBox, vector< Box \* > &vectorBox-Final, Data &data, long int &nbToCompute, long int &nbWithNeighbor)

This method recomposes a group of Boxes into an unique one.

void weightedSumOneStep (vector< Box \* > &vectorBox, Data &data)

Compute the weighted sum method to split a Box in two Boxes.

#### 4.1.1 Function Documentation

```
4.1.1.1 addChildren ( Box * boxMother, vector < Box * > & vBox )
```

This method adds children of a Box into a vector of Boxes.

This method computes all the children Boxes of a Box into a vector of Boxes. A children is defined by a combination of Facility in which indices have not yet been opened.

#### **Parameters**

in		: A Box for which ones wants to add children Boxes.
in,out	vBox	: A vector of Box in whichone ones add all the children -
		Boxes at the end (enqueue at the end of the vector).

4.1.1.2 boxFiltering ( vector< Box \* > & vectorBox, Data & data, long int & nbToCompute, long int & nbWithNeighbor )

This method splits a Box into two Boxes.

Using a weighted sum method to find a supported point, ones splits a into two Boxes.

#### **Parameters**

in,out	vectorBox	: A vector of Box in which one ones adds all the new Boxes
		computed.
in	data	: A Data object which contains all the values of the in-
		stance.
in,out	nbTo-	: A long int that takes the number of Boxes with a available
	Compute	decomposition.
in,out	nbWith-	: A long int that takes the number of Boxes overlapped by
	Neighbor	another one.

4.1.1.3 long int createBox ( vector < Box \* > & vectorBox, Data & data )

This method computes all the initial  ${\tt Boxes}$  of our algorithm.

This method computes all the Boxes in whichones the Label Setting algorithm will runs. This method uses a smart Branch&Bound (Breadth First Search, splitting on – Facility setup variables) to compute all the feasible and important Boxes. Useless Boxes are avoided by the Branch&Bound.

#### **Parameters**

in,out	vectorBox	: A vector of Box, empty a the beginning, and containing all
		the Boxes at the end of this method.
in	data	: A Data object which contains all the values of the in-
		stance.

#### Returns

A long int which represents the number of Boxes computed by the method.

4.1.1.4 filter ( vector< Box \* > & vectorBox, long int & nbToCompute, long int & nbWithNeighbor )

This method filters all the Boxes of a vector of Box.

This method filters Boxes by eliminating all Box that are dominated by an other one.

#### **Parameters**

in,out	nbTo-	: A long int that takes the number of Boxes with a available
	Compute	decomposition.
in,out	nbWith-	: A long int that takes the number of Boxes overlapped by
	Neighbor	another one.

4.1.1.5 recomposition ( vector< Box \* > & vectorBox, vector< Box \* > & vectorBoxFinal, Data & data, long int & nbToCompute, long int & nbWithNeighbor )

This method recomposes a group of Boxes into an unique one.

This method selects all the Boxes with the same combination of Facility. Ones creates an unique Box with this combination of Facility and update the bounds of this one w.r.t. the bounds of all Boxes selected.

#### **Parameters**

in,out	vectorBox	: A vector of Box in which one ones adds all the new Boxes
		computed.
in,out	vectorBox-	: A vector of Box containing all the recompose Boxes
	Final	
in	data	: A Data object which contains all the values of the in-
		stance.
in,out	nbTo-	: A long int that takes the number of Boxes with a available
	Compute	decomposition.
in,out	nbWith-	: A long int that takes the number of Boxes overlapped by
	Neighbor	another one.

## 4.1.1.6 weightedSumOneStep ( vector< Box \* > & vectorBox, Data & data )

Compute the weighted sum method to split a Box in two Boxes.

This method adds to the vectorBox the results of dividing each Box in two others. It uses the weighted sum method which splits Box by the first supported point founded.

#### **Parameters**

in	vectorBox	: A vector of Box which contains all the Boxes in which
		one ones computes the weighted sum method.
in	data	: A Data object which contains all the values of the in-
		stance.

#### Returns

A long int which value is the number of solutions of the algorithm proposed.

# 4.2 Methods of Generating

#### **Functions**

long int runLabelSetting (vector < Box \* > &vectorBox, Data &data)

This method runs the Label Setting algorithm.

void filterListSolution (list< Solution > &lsol)

Delete the solutions dominated.

#### 4.2.1 Function Documentation

### 4.2.1.1 filterListSolution ( list< Solution > & Isol )

Delete the solutions dominated.

A method to delete all the dominated solutions into the objective space. This method computes a complete set of solution.

#### **Parameters**

in	Isol	:	Α	vector	of	Solution	which	contains	all	the	_
		So	lu	tions	to	delete.					

# 4.2.1.2 runLabelSetting ( vector< Box \* > & vectorBox, Data & data )

This method runs the Label Setting algorithm.

This method executes a Label Setting algorithm in each Box of the input vector.

#### **Parameters**

in	vectorBox	: A vector of Box which contains all the Boxes in which
		one ones runs the algorithm of Label Setting.
in	data	: A Data object which contains all the values of the in-
		stance.

#### Returns

A long int which value is the number of solutions of the algorithm proposed.

4.3 Others Methods 11

## 4.3 Others Methods

#### **Functions**

double computeCorrelation (Data &data)

This method computes the correlation.

• float time\_ms\_Diff (timeval tvStart, timeval tvEnd)

This method computes the difference in milli-seconds (ms) between two times.

float time\_s\_Diff (timeval tvStart, timeval tvEnd)

This method computes the difference in seconds (ms) between two times.

#### 4.3.1 Function Documentation

#### 4.3.1.1 computeCorrelation ( Data & data )

This method computes the correlation.

This method computes the correlation between the two objectives w.r.t. to the Data.

#### **Parameters**

in	data	: A Data object which contains all the values of the in-
		stance.

#### Returns

A double representing the correlation between the two objectives.

#### 4.3.1.2 time\_ms\_Diff ( timeval tvStart, timeval tvEnd )

This method computes the difference in milli-seconds (ms) between two times.

## Parameters

in	tvStart	: A structure of time represeting the begin time.
in	tvEnd	: A structure of time represeting the end time.

#### Returns

A double representing the difference in ms between the two times.

#### 4.3.1.3 time\_s\_Diff ( timeval tvStart, timeval tvEnd )

This method computes the difference in seconds (ms) between two times.

# **Parameters**

in	tvStart	: A structure of time represeting the begin time.
in	tvEnd	: A structure of time represeting the end time.

# Returns

A double representing the difference in s between the two times.

4.4 Global Variables 13

## 4.4 Global Variables

## **Variables**

· bool modeVerbose

Variable representing the Verbose mode.

bool modeExport

Variable representing the Export mode.

# 4.4.1 Variable Documentation

#### 4.4.1.1 modeExport

Variable representing the Export mode.

This boolean gets the value TRUE if the export mode is on, and FALSE otherwise. If the export mode is on, result files are written in the folder /res.

#### 4.4.1.2 modeVerbose

Variable representing the Verbose mode.

This boolean gets the value TRUE if the verbose mode is on, and FALSE otherwise. If the verbose mode is on, the software prints detailled information while running.

# 4.5 Main

# **Functions**

• int main (int argc, char \*argv[])

This is the main of the software.

# 4.5.1 Function Documentation

4.5.1.1 int main ( int argc, char \* argv[] )

This is the main of the software.

in	argc	: An integer which represents the number of arguments
		passed to the line command.
in	argv	: An array of character which represents all the arguments.

# **Chapter 5**

# **Class Documentation**

# 5.1 Box Class Reference

```
Class to represent a Box. #include "Box.hpp"
```

#### **Public Member Functions**

- Box (Data &data)
  - Default Constructor of the class Box.
- Box (Box \*copy)
  - Constructor of copy of the class Box.
- Box (Data &data, bool \*toOpen)
  - Constructor of the class Box.
- ∼Box ()
  - Destructor of the class Box.
- double getMinZ1 () const
  - Getter for the minimum value w.r.t. objective 1.
- double getMinZ2 () const
  - Getter for the minimum value w.r.t. objective 2.
- double getMaxZ1 () const
  - Getter for the maximum value w.r.t. objective 1.
- double getMaxZ2 () const
  - Getter for the maximum value w.r.t. objective 2.
- double getOriginZ1 () const
  - Getter for the value of the origin w.r.t. objective 1.
- double getOriginZ2 () const
  - Getter for the value of the origin w.r.t. objective 2.
- string getId () const

Getter for the id.

· bool is Assigned (int cust) const

method to know if a Customer is assigned or not.

• bool isOpened (int fac) const

method to know if a Facility is opened or not.

• int getnbCustomerNotAffected () const

Getter for the number of Customers nonaffected.

• int getNbFacilityOpen () const

Getter for the number of Facilities opened.

• bool getHasNeighbor () const

Getter for the neighborhood.

• bool getHasMoreStepWS () const

Getter for the number of Weighted Sum.

• Data & getData () const

Getter for the data.

void setId (string s)

Setter for the id of this Box.

• void setMinZ1 (double v)

Setter for the minimum value w.r.t. objective 1.

void setMinZ2 (double v)

Setter for the minimum value w.r.t. objective 2.

void setMaxZ1 (double v)

Setter for the maximum value w.r.t. objective 1.

void setMaxZ2 (double v)

Setter for the maximum value w.r.t. objective 2.

void setHasNeighbor (bool b)

Setter for the neighborhood.

• void setHasMoreStepWS (bool b)

Setter for the remaining weighted sum method.

• void computeBox ()

A method to expand a box, which means attempting to allocate all Customers to Facilities.

void openFacility (int fac)

A method that opens a Facility in this Box, by adding all the location cost of the two objectives.

• void print ()

A method to print informations about this Box.

#### **Private Attributes**

- string id
- Data & data
- bool \* isAssigned
- bool \* facility\_
- bool hasMoreStepWS
- bool hasNeighbor\_
- int nbCustomerNotAffected\_
- double minZ1
- double minZ2
- double maxZ1
- double maxZ2\_
- double originZ1\_
- double originZ2

#### **Related Functions**

(Note that these are not member functions.)

- bool isDominatedBetweenTwoBoxes (Box \*box1, Box \*box2)
  - Method of comparison between two boxes.
- bool isDominatedBetweenOrigins (Box \*box1, Box \*box2)

Method of comparison between two boxes.

void filterDominatedBoxes (vector< Box \* > &vectBox)

Method to filter a vector of Boxes.

bool isDominatedByItsOrigin (vector < Box \* > &vectBox, Box \*box)

Method of comparison between a Box and a vector of Boxes.

bool isDominatedByItsBox (vector < Box \* > &vectBox, Box \*box)

Method of comparison between a Box and a vector of Boxes.

#### 5.1.1 Detailed Description

Class to represent a Box.

This class represents a Box with all its attributes and methods.

#### 5.1.2 Constructor & Destructor Documentation

### 5.1.2.1 Box::Box ( Data & data )

Default Constructor of the class Box.

The default construtor gives a Box which anyone Facility opened.

#### **Parameters**

in	data	: A Data object which contains all the values of the in-
		stance.

## 5.1.2.2 Box::Box ( Box \* copy )

Constructor of copy of the class Box.

#### **Parameters**

	anny A. A. D do anny	
l ln	copy   : A Box to copy.	

## 5.1.2.3 Box::Box ( Data & data, bool \* toOpen )

Constructor of the class Box.

This construtor gives a Box which a set of Facilities opened.

#### **Parameters**

in	1	data	: A Data object which contains all the values of the in-
			stance.
in	1	toOpen	: A pointer of boolean representing the vector of Facility
			to open in order to construct an object Box.

## 5.1.2.4 Box::∼Box ( )

Destructor of the class Box.

#### 5.1.3 Member Function Documentation

#### 5.1.3.1 void Box::computeBox()

A method to expand a box, which means attempting to allocate all  ${\tt Customers}$  to  ${\tt Facilities}.$ 

## 5.1.3.2 Data & Box::getData ( ) const

Getter for the data.

#### Returns

A reference Data of the Data of this Box.

5.1.3.3 bool Box::getHasMoreStepWS() const

Getter for the number of Weighted Sum.

#### **Returns**

A boolean if this Box gets a remaining iteration of weighhed sum method.

5.1.3.4 bool Box::getHasNeighbor ( ) const

Getter for the neighborhood.

#### Returns

A boolean which value is TRUE if this Box overlaps or is overlapped with an other Box.

5.1.3.5 string Box::getId ( ) const

Getter for the id.

#### Returns

A string as the sequence of 1 and 0 representing the combination of Facility of this Box.

5.1.3.6 double Box::getMaxZ1 ( ) const

Getter for the maximum value w.r.t. objective 1.

Returns

A double as the maximum value w.r.t. objective 1 of this  ${\tt Box}.$ 

5.1.3.7 double Box::getMaxZ2 ( ) const

Getter for the maximum value w.r.t. objective 2.

Returns

A double as the maximum value w.r.t. objective 2 of this Box.

```
5.1.3.8 double Box::getMinZ1 ( ) const
```

Getter for the minimum value w.r.t. objective 1.

Returns

A double as the minimum value w.r.t. objective 1 of this Box.

```
5.1.3.9 double Box::getMinZ2 ( ) const
```

Getter for the minimum value w.r.t. objective 2.

Returns

A double as the minimum value w.r.t. objective 2 of this Box.

```
5.1.3.10 int Box::getnbCustomerNotAffected ( ) const
```

Getter for the number of Customers nonaffected.

Returns

An int as the number of Customers which are not affected to a Facility.

```
5.1.3.11 int Box::getNbFacilityOpen ( ) const
```

Getter for the number of Facilities opened.

Returns

An int as the number of Facilities which are opened (set to 1).

```
5.1.3.12 double Box::getOriginZ1 ( ) const
```

Getter for the value of the origin w.r.t. objective 1.

Returns

A double as the value of the point of origin w.r.t. objective 1 of this Box.

```
5.1.3.13 double Box::getOriginZ2 ( ) const
```

Getter for the value of the origin w.r.t. objective 2.

Returns

A double as the value of the point of origin w.r.t. objective 2 of this Box.

#### 5.1.3.14 bool Box::isAssigned (int cust) const

method to know if a Customer is assigned or not.

#### **Parameters**

_			
	in	cust	: A Customer.

#### **Returns**

A boolean which value is TRUE if the Customer is assigned to any Facility.

#### 5.1.3.15 bool Box::isOpened (int fac) const

method to know if a Facility is opened or not.

#### **Parameters**

in	cust : A Facility.	
----	--------------------	--

#### Returns

A boolean which value is TRUE if the Facility is opened.

#### 5.1.3.16 void Box::openFacility (int fac)

A method that opens a Facility in this Box, by adding all the location cost of the two objectives.

#### **Parameters**

in	fac	: A Facility to open.

# 5.1.3.17 void Box::print()

A method to print informations about this Box.

### 5.1.3.18 void Box::setHasMoreStepWS (bool b)

Setter for the remaining weighted sum method.

in	b	: A boolean which value is TRUE if this $Box$ has a remaining
		iteration of weighted sum method.

## 5.1.3.19 void Box::setHasNeighbor (bool b)

Setter for the neighborhood.

#### **Parameters**

in	b	: A boolean which value is TRUE if this Box has at least
		one neighborhood (a Box which overlaps or is overlapped).

## 5.1.3.20 void Box::setId ( string s )

Setter for the id of this Box.

#### **Parameters**

in	s	:	Α	string	which	represents	the	id	(combination	of	-
		Fε	aci	Llity	) of this	Box.					

# 5.1.3.21 void Box::setMaxZ1 ( double v )

Setter for the maximum value w.r.t. objective 1.

### Parameters

in	V	: A double which represents the maximum value w.r.t.	ob-
		jective 1 of this Box.	

## 5.1.3.22 void Box::setMaxZ2 ( double v )

Setter for the maximum value w.r.t. objective 2.

#### **Parameters**

in	V	: A double which represents the maximum value w.r.t.	ob-
		jective 2 of this Box.	

# 5.1.3.23 void Box::setMinZ1 ( double v )

Setter for the minimum value w.r.t. objective 1.

in	V	: A double which represents the minimum value w.r.t. objec-
		tive 1 of this Box.

#### 5.1.3.24 void Box::setMinZ2 ( double v )

Setter for the minimum value w.r.t. objective 2.

#### **Parameters**

Γ	in	V	: A double which represents the minimum value w.r.t. objec-
			tive 2 of this Box.

#### 5.1.4 Friends And Related Function Documentation

#### **5.1.4.1** void filterDominatedBoxes (vector < Box \* > & vectBox ) [related]

Method to filter a vector of Boxes.

A method to filter and delete Box in a vector of Boxes by comparing each other.

#### **Parameters**

in	vectBox	: A vector of Boxes.
----	---------	----------------------

#### **5.1.4.2** bool isDominatedBetweenOrigins ( Box \* box1, Box \* box2 ) [related]

Method of comparison between two boxes.

A method to compare two Boxes using the bounds minZ1\_, minZ2\_, maxZ1\_, maxZ2\_, originZ1\_ and originZ2\_.

#### **Parameters**

in	box1	: A Box to compare.
in	box2	: A Box to compare.

#### Returns

A boolean which value is TRUE if the origin of the box1 is dominated by the box2.

# 5.1.4.3 boolisDominatedBetweenTwoBoxes ( Box \* box1, Box \* box2 ) [related]

Method of comparison between two boxes.

A method to compare two Boxes using the bounds minZ1\_, minZ2\_, maxZ1\_, maxZ2\_ of each Boxes.

in box2 : A Box to compare.	in	box1	: A Box to compare.
	in	box2	: A Box to compare.

#### Returns

A boolean which value is TRUE if the box1 is dominated by the box2.

# 5.1.4.4 boolisDominatedByltsBox ( vector< Box \* > & vectBox, Box \* box ) [related]

Method of comparison between a Box and a vector of Boxes.

#### **Parameters**

	in	vectBox	: A vector of Boxes to compare.
Г	in	box	: A Box to compare.

#### Returns

A boolean which value is TRUE if one of the Box of the vector vectBox is dominated by the Box box.

5.1.4.5 bool isDominatedByltsOrigin ( vector< Box 
$$* > \& vectBox$$
, Box  $* box$  ) [related]

Method of comparison between a Box and a vector of Boxes.

#### **Parameters**

in	vectBox	: A vector of Boxes to compare.
in	box	: A Box to compare.

# Returns

A boolean which value is TRUE if Box box is dominated by one of the Box of the vector vectBox.

#### 5.1.5 Member Data Documentation

```
5.1.5.1 Data& Box::data_ [private]
```

A reference to the Data of this Box

```
5.1.5.2 bool* Box::facility_ [private]
```

A boolean which represents the vector of Facility opened or not

```
5.1.5.3 bool Box::hasMoreStepWS_ [private]
```

A boolean which represents if this  ${\tt Box}$  has a remaining iteration of weighted sum method

```
5.1.5.4 bool Box::hasNeighbor_ [private]
```

A boolean which represents if this Box has a neighboor or not

```
5.1.5.5 string Box::id_ [private]
```

A string which represents the id of this Box

```
5.1.5.6 bool* Box::isAssigned_ [private]
```

A boolean which represents the vector of Customer assigned or not

```
5.1.5.7 double Box::maxZ1_ [private]
```

A double which represents the maximum value w.r.t. objective 1

```
5.1.5.8 double Box::maxZ2_ [private]
```

A double which represents the maximum value w.r.t. objective 2

```
5.1.5.9 double Box::minZ1_ [private]
```

A double which represents the minimum value w.r.t. objective 1

```
5.1.5.10 double Box::minZ2_ [private]
```

A double which represents the minimum value w.r.t. objective 2

```
5.1.5.11 int Box::nbCustomerNotAffected_ [private]
```

An integer which represents the number of Customer not affected of this Box

```
5.1.5.12 double Box::originZ1_ [private]
```

A double which represents the value of the origin of this Box w.r.t. objective 1

```
5.1.5.13 double Box::originZ2 [private]
```

A double which represents the value of the origin of this Box w.r.t. objective 2

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

- /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Box.hpp
- /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Box.cpp

#### 5.2 Customer Class Reference

```
Class to represent a Customer.
#include "Customer.hpp"
```

#### **Public Member Functions**

• Customer (unsigned short &x, unsigned short &y)

Constructor of the class Customer.

• unsigned short getCoordX () const

Getter for the x coordinate.

• unsigned short getCoordY () const

Getter for the y coordinate.

#### **Private Attributes**

- unsigned short coordX
- unsigned short coordY\_

#### **Related Functions**

(Note that these are not member functions.)

ostream & operator << (ostream &out, const Customer \*cust)</li>
 Operator overloading.

#### 5.2.1 Detailed Description

Class to represent a Customer.

This class represents a Customer with all its attributes. The pair (x,y) representes respectively the first and second coordinates in a bi-dimensional geographical space.

#### 5.2.2 Constructor & Destructor Documentation

#### 5.2.2.1 Customer::Customer ( unsigned short & x, unsigned short & y )

Constructor of the class Customer.

#### **Parameters**

in	X	: An unsigned short which represents the x coordinate of
		the Customer.
in	У	: An unsigned short which represents the y coordinate of
		the Customer.

#### 5.2.3 Member Function Documentation

#### 5.2.3.1 unsigned short Customer::getCoordX ( ) const

Getter for the x coordinate.

#### Returns

An unsigned short as the x coordinate of this Customer.

#### 5.2.3.2 unsigned short Customer::getCoordY() const

Getter for the y coordinate.

#### Returns

An unsigned short as the y coordinate of this Customer.

#### 5.2.4 Friends And Related Function Documentation

5.2.4.1 ostream & operator << ( ostream & out, const Customer \* cust ) [related]

Operator overloading.

Overloading of the standard output stream in order to print a Customer.

#### **Parameters**

out	out	: The standard output stream.
in	cust	: A Customer to print in the standard output stream.

#### 5.2.5 Member Data Documentation

```
5.2.5.1 unsigned short Customer::coordX [private]
```

Unsigned short which represents the value of the x coordinate of this Customer

```
5.2.5.2 unsigned short Customer::coordY [private]
```

Unsigned short which represents the value of the y coordinate of this Customer

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

- /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Customer.hpp
- /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Customer.cpp

#### 5.3 Data Class Reference

```
Class to represent a Data. #include "Data.hpp"
```

#### **Public Types**

• typedef vector< Facility > ListFacilities

The list of all facilities.

typedef vector < Customer > ListCustomers

The list of all customers.

#### **Public Member Functions**

- Data (unsigned int nbFacility, unsigned int nbCustomer, string name)
  - Constructor of the class Data.
- ~Data ()

Destructor of the class Data.

• void addFacility (Facility fac)

Method to add a Facility to the Data.

void addCustomer (Customer cust)

Method to add a Customer to the Data.

• unsigned int getnbFacility () const

Getter for the number of facilities.

• unsigned int getnbCustomer () const

Getter for the number of customers.

double getAllocationObj1Cost (int cust, int fac) const

Getter for the allocation cost w.r.t. objective 1 between a Customer and a - Facility.

• double getAllocationObj2Cost (int cust, int fac) const

Getter for the allocation cost w.r.t. objective 2 between a Customer and a - Facility.

• Facility & getFacility (int fac)

Getter for a Facility.

• string getFileName () const

Getter for the name of the instance.

• void setAllocationObj1Cost (int cust, int fac, double val)

Setter for the allocation cost w.r.t. objective 1 between a Customer and a - Facility.

void setAllocationObj2Cost (int cust, int fac, double val)

Setter for the allocation cost w.r.t. objective 2 between a Customer and a - Facility.

• void setFileName (string name)

Setter for the name of the instance.

#### **Private Attributes**

- ListFacilities facilityList
- ListCustomers customerList\_
- string fileName
- double \*\* allocationObj1Cost\_
- double \*\* allocationObj2Cost\_

#### 5.3.1 Detailed Description

Class to represent a Data.

This class represents a Data with all its attributes, parameters.

#### 5.3.2 Member Typedef Documentation

5.3.2.1 vector < Customer > Data::ListCustomers

The list of all customers.

This is a vector from the STL Containers of object Customer.

5.3.2.2 vector < Facility > Data::ListFacilities

The list of all facilities.

This is a vector from the STL Containers of object Facility.

#### 5.3.3 Constructor & Destructor Documentation

#### 5.3.3.1 Data::Data ( unsigned int nbFacility, unsigned int nbCustomer, string name )

Constructor of the class Data.

#### **Parameters**

ſ	in	nbFacility	: The number of Facility of the instance.
ſ	in	nbCustomer	: The number of Customer of the instance.
Ī	in	name	: The name of the instance.

#### 5.3.3.2 Data::∼Data( )

Destructor of the class Data.

#### 5.3.4 Member Function Documentation

#### 5.3.4.1 void Data::addCustomer ( Customer cust )

Method to add a Customer to the Data.

#### **Parameters**

cust : An object Customer.	

#### 5.3.4.2 void Data::addFacility ( Facility fac )

Method to add a Facility to the Data.

#### **Parameters**

fac : An object Facility.
---------------------------

#### 5.3.4.3 double Data::getAllocationObj1Cost (int cust, int fac) const

Getter for the allocation cost w.r.t. objective 1 between a Customer and a Facility.

#### **Parameters**

in	cust	: The index of the Customer.
in	fac	: The index of the Facility.

#### **Returns**

A double as the value of the allocation cost w.r.t. objective 1 for the Customer cust to the Facility fac.

#### 5.3.4.4 double Data::getAllocationObj2Cost ( int cust, int fac ) const

Getter for the allocation cost w.r.t. objective 2 between a Customer and a Facility.

#### **Parameters**

in	cust	: The index of the Customer.
in	fac	: The index of the Facility.

#### **Returns**

A double as the value of the allocation cost w.r.t. objective 2 for the Customer cust to the Facility fac.

#### 5.3.4.5 Facility & Data::getFacility (int fac)

Getter for a Facility.

#### **Parameters**

in	fac	: The index of the Facility to return.
----	-----	----------------------------------------

#### Returns

A Facility.

#### 5.3.4.6 string Data::getFileName ( ) const

Getter for the name of the instance.

#### **Returns**

A string which represents the name of the instance.

#### 5.3.4.7 unsigned int Data::getnbCustomer ( ) const

Getter for the number of customers.

#### Returns

An unsigned int as the number of customers of the instance.

#### 5.3.4.8 unsigned int Data::getnbFacility ( ) const

Getter for the number of facilities.

#### Returns

An unsigned int as the number of facilities of the instance.

#### 5.3.4.9 void Data::setAllocationObj1Cost (int cust, int fac, double val)

Setter for the allocation cost w.r.t. objective 1 between a Customer and a Facility.

#### **Parameters**

in	cust	: The index of the Customer.
in	fac	: The index of the Facility.
in	val	: The value of the allocation cost of the customer cust to the
		facility fac w.r.t. objective 1.

#### 5.3.4.10 void Data::setAllocationObj2Cost (int cust, int fac, double val)

Setter for the allocation cost w.r.t. objective 2 between a Customer and a Facility.

#### **Parameters**

in	cust	: The index of the Customer.
in	fac	: The index of the Facility.
in	val	: The value of the allocation cost of the customer cust to the
		facility fac w.r.t. objective 2.

#### 5.3.4.11 void Data::setFileName ( string name )

Setter for the name of the instance.

#### Parameters

in	name	: A string which represents the name of the instance.

#### 5.3.5 Member Data Documentation

#### **5.3.5.1** double\*\* Data::allocationObj1Cost\_ [private]

An array of double (2 dimensions) which represents the matrix of allocation cost w.r.t. objective 1

```
5.3.5.2 double** Data::allocationObj2Cost_ [private]
```

An array of double (2 dimensions) which represents the matrix of allocation cost w.r.t. objective 2

```
5.3.5.3 ListCustomers Data::customerList_ [private]5.3.5.4 ListFacilities Data::facilityList_ [private]
```

A string which represents the name of the instance

5.3.5.5 string Data::fileName\_ [private]

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

- /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Data.hpp
- /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Data.cpp

#### 5.4 Facility Class Reference

```
Class to represent a Facility.
#include "Facility.hpp"
```

#### **Public Member Functions**

- Facility (unsigned short &x, unsigned short &y)
   Constructor of the class Facility.
- unsigned short getCoordX () const

Getter for the x coordinate.

• unsigned short getCoordY () const

Getter for the y coordinate.

• double getLocationObj1Cost () const

Getter for the location cost w.r.t. objective 1.

• double getLocationObj2Cost () const

Getter for the location cost w.r.t. objective 2.

void setLocationObj1Cost (double &val)

Setter for the location cost w.r.t. objective 1.

· void setLocationObj2Cost (double &val)

Setter for the location cost w.r.t. objective 2.

#### **Private Attributes**

- unsigned short coordX
- unsigned short coordY\_
- double locationObj1Cost\_
- double locationObj2Cost

#### **Related Functions**

(Note that these are not member functions.)

std::ostream & operator<< (std::ostream &out, const Facility \*fac)</li>
 Operator overloading.

#### 5.4.1 Detailed Description

Class to represent a Facility.

This class represents a Facility with all its attributes. The pair (x,y) representes respectively the first and second coordinates in a bi-dimensional geographical space.

#### 5.4.2 Constructor & Destructor Documentation

#### 5.4.2.1 Facility::Facility ( unsigned short & x, unsigned short & y )

Constructor of the class Facility.

#### **Parameters**

in	Х	: An unsigned integer which represents the x coordinate of
		the Facility.
in	У	: An unsigned integer which represents the y coordinate of
		the Facility.

#### 5.4.3 Member Function Documentation

5.4.3.1 unsigned short Facility::getCoordX ( ) const

Getter for the x coordinate.

#### Returns

An unsigned short as the x coordinate of this Facility.

5.4.3.2 unsigned short Facility::getCoordY ( ) const

Getter for the y coordinate.

#### **Returns**

An unsigned short as the y coordinate of this Facility.

5.4.3.3 double Facility::getLocationObj1Cost ( ) const

Getter for the location cost w.r.t. objective 1.

#### **Returns**

A double as the location cost w.r.t. objective 1 of this Facility.

5.4.3.4 double Facility::getLocationObj2Cost ( ) const

Getter for the location cost w.r.t. objective 2.

#### Returns

A double as the location cost w.r.t. objective 2 of this Facility.

5.4.3.5 void Facility::setLocationObj1Cost ( double & val )

Setter for the location cost w.r.t. objective 1.

#### Parameters

in	val	: A double which represents the value of the location cost of
		this Facility w.r.t. objective 1.

5.4.3.6 void Facility::setLocationObj2Cost ( double & val )

Setter for the location cost w.r.t. objective 2.

#### **Parameters**

in	val	: A double which represents the value of the location cost of
		this Facility w.r.t. objective 2.

#### 5.4.4 Friends And Related Function Documentation

5.4.4.1 std::ostream & operator << ( std::ostream & out, const Facility \* fac ) [related]

Operator overloading.

Overloading of the standard output stream in order to print a Facility.

#### **Parameters**

out	out	: The standard output stream.
in	fac	: A Facility to print in the standard output stream.

#### 5.4.5 Member Data Documentation

**5.4.5.1 unsigned short Facility::coordX** [private]

Unsigned short which represents the value of the x coordinate of this Facility

**5.4.5.2 unsigned short Facility::coordY** [private]

Unsigned short which represents the value of the y coordinate of this Facility

**5.4.5.3 double Facility::locationObj1Cost\_** [private]

Double which represents the value of the location cost of this Facility w.r.t. objective

**5.4.5.4 double Facility::locationObj2Cost** [private]

Double which represents the value of the location cost of this  ${\tt Facility}$  w.r.t. objective 2

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

- /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Facility.hpp
- /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Facility.cpp

### 5.5 LabelSetting Class Reference

Class to represent a LabelSetting.

#include "LabelSetting.hpp"

#### **Public Member Functions**

LabelSetting (Box &box)

Default Constructor of the class LabelSetting.

∼LabelSetting ()

Destructor of the class LabelSetting.

• void compute ()

A method to compute all solutions into a Box.

• void computeNode (int indexNode)

A method to compute all the labels of the algorithm at level indexNode.

void print ()

A method to print informations about the label setting algorithm.

• unsigned int getRank () const

Getter for the rank (number of level).

#### **Public Attributes**

Node \* nodes

#### **Private Attributes**

- unsigned int nbRank\_
- double boundZ1
- double boundZ2
- Data & data\_

#### 5.5.1 Detailed Description

Class to represent a LabelSetting.

This class represents a LabelSetting with all its attributes and methods.

#### 5.5.2 Constructor & Destructor Documentation

#### 5.5.2.1 LabelSetting::LabelSetting ( Box & box )

Default Constructor of the class LabelSetting.

#### **Parameters**

	in	box	: A Box object in which one ones runs	the label setting
1			algorithm.	

```
5.5.2.2 LabelSetting::~LabelSetting()
```

Destructor of the class LabelSetting.

#### 5.5.3 Member Function Documentation

```
5.5.3.1 void LabelSetting::compute ( )
```

A method to compute all solutions into a Box.

#### 5.5.3.2 void LabelSetting::computeNode (int indexNode)

A method to compute all the labels of the algorithm at level indexNode.

#### **Parameters**

indexNode : The level in which one ones executes the algorithm.

#### 5.5.3.3 unsigned int LabelSetting::getRank() const

Getter for the rank (number of level).

#### Returns

A unsigned int which value is the rank (number of level) of the algorithm.

#### 5.5.3.4 void LabelSetting::print()

A method to print informations about the label setting algorithm.

#### 5.5.4 Member Data Documentation

```
5.5.4.1 double LabelSetting::boundZ1_ [private]
```

A unsigned int which represents the maximum value w.r.t. objective 1 of this Box

```
5.5.4.2 double LabelSetting::boundZ2 [private]
```

A unsigned int which represents the maximum value w.r.t. objective 2 of this Box

#### **5.5.4.3 Data& LabelSetting::data\_** [private]

A reference to the Data of this Box

```
5.5.4.4 unsigned int LabelSetting::nbRank [private]
```

A unsigned int which represents the number of level of this algorithm in this Box

```
5.5.4.5 Node* LabelSetting::nodes_
```

A pointer of Node which represents all the solutions in each level

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

- /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Label-Setting.hpp
- /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Label-Setting.cpp

#### 5.6 Node Class Reference

```
Class to represent a Node.
```

```
#include "Node.hpp"
```

#### **Public Member Functions**

• Node ()

Default Constructor of the class Node.

• Node (int size)

Default Constructor of the class Node.

• ∼Node ()

Destructor of the class Node.

void setSize (unsigned int s)

Setter for the size of this Node.

unsigned int getSize () const

Getter for the size.

• double getCostToEnterZ1 (int i) const

Getter for the cost to enter to this Node w.r.t. objective 1.

double getCostToEnterZ2 (int i) const

Getter for the cost to enter to this Node w.r.t. objective 2.

• void clearLabels ()

A method to delete all the labels of this Node.

• void setValues (int index, double z1, double z2)

Setter of the values of the solution ones want to set.

void print ()

A method to print informations about this Node.

#### **Public Attributes**

list < Solution > labels\_

#### **Private Attributes**

- unsigned int size\_
- double \* costToEnterZ1\_
- double \* costToEnterZ2

#### 5.6.1 Detailed Description

Class to represent a Node.

This class represents a Node with all its attributes and methods.

#### 5.6.2 Constructor & Destructor Documentation

```
5.6.2.1 Node::Node()
```

Default Constructor of the class Node.

5.6.2.2 Node::Node ( int size )

Default Constructor of the class Node.

#### **Parameters**

in	size	: An int which represents the size of this node.

```
5.6.2.3 Node::∼Node( )
```

Destructor of the class Node.

#### 5.6.3 Member Function Documentation

#### 5.6.3.1 void Node::clearLabels ( )

A method to delete all the labels of this Node.

#### 5.6.3.2 double Node::getCostToEnterZ1 ( int i ) const

Getter for the cost to enter to this Node w.r.t. objective 1.

#### **Parameters**

*i* : The index.

#### **Returns**

A double as the cost to enter to this Node w.r.t. objective 1 at the index i.

5.6.3.3 double Node::getCostToEnterZ2 ( int i ) const

Getter for the cost to enter to this Node w.r.t. objective 2.

#### **Parameters**

i : The index.

#### Returns

A double as the cost to enter to this Node w.r.t. objective 2 at the index i.

5.6.3.4 unsigned int Node::getSize ( ) const

Getter for the size.

#### Returns

An unsigned int as the size of this Node.

5.6.3.5 void Node::print()

A method to print informations about this Node.

5.6.3.6 void Node::setSize (unsigned int s)

Setter for the size of this Node.

#### **Parameters**

5.6.3.7 void Node::setValues (int index, double z1, double z2)

Setter of the values of the solution ones want to set.

#### **Parameters**

in	index	: The index.
in	z1	: The value of the solution w.r.t. objective 1 ones wants to
		set.
in	z2	: The value of the solution w.r.t. objective 2 ones wants to
		set.

#### 5.6.4 Member Data Documentation

```
5.6.4.1 double* Node::costToEnterZ1_ [private]
```

A pointer of double which represents all the values to enter to this Node w.r.t objective

```
5.6.4.2 double* Node::costToEnterZ2_ [private]
```

A pointer of double which represents all the values to enter to this Node w.r.t objective

5.6.4.3 list<Solution> Node::labels\_

A list of Solution which represents all the Solutions (or labels) in this Node

```
5.6.4.4 unsigned int Node::size_ [private]
```

A unsigned int which represents the size of this Node

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

- /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Node.hpp
- /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Node.cpp

#### 5.7 Parser Class Reference

Class to represent a Parser.

#include "Parser.hpp"

#### **Static Public Member Functions**

• static Data \* Parsing (const char \*filename)

Static method to parse the instance.

#### **Static Private Member Functions**

- static void ignoreLine (ifstream &file)
   Static method to skip a line from the instance.
- static void ignoreChar (ifstream &file)

Static method to skip a character from the instance.

#### 5.7.1 Detailed Description

Class to represent a Parser.

This class represents a Parser with all its methods to read and parse an instance file.

#### 5.7.2 Member Function Documentation

**5.7.2.1 void Parser::ignoreChar (ifstream & file)** [static, private]

Static method to skip a character from the instance.

#### **Parameters**

in	file	: A ifstream (IOstream) which represents the file readed.
----	------	-----------------------------------------------------------

**5.7.2.2 void Parser::ignoreLine (ifstream & file)** [static, private]

Static method to skip a line from the instance.

#### **Parameters**

in	file: A ifstream (IOstream) which represents the file readed.

**5.7.2.3 Data \* Parser::Parsing (const char \* filename )** [static]

Static method to parse the instance.

#### **Parameters**

in	filename	: A const char which represents the instance file.
----	----------	----------------------------------------------------

#### Returns

A Data object which contains all the values of the instance file.

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

- /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Parser.hpp
- /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Parser.cpp

#### 5.8 Solution Class Reference

```
Class to represent a Solution. #include "Solution.hpp"
```

#### **Public Member Functions**

• Solution ()

Default constructor of the class Solution.

Solution (double obj1, double obj2)

Constructor of the class Solution.

•  $\sim$ Solution ()

Destructor of the class Solution.

void setObj1 (double obj)

Setter for the value w.r.t objective 1 of this Solution.

• void setObj2 (double obj)

Setter for the value w.r.t objective 2 of this Solution.

• double getObj1 () const

Getter for the value w.r.t. objective 1 of this Solution.

double getObj2 () const

Getter for the value w.r.t. objective 2 of this Solution.

#### **Private Attributes**

- double obj1\_
- double obj2

#### **Related Functions**

(Note that these are not member functions.)

bool operator< (Solution s1, Solution s2)</li>

Operator overloading.

#### 5.8.1 Detailed Description

Class to represent a Solution.

This class represents a Solution with all its two attributes (obj1\_ and obj2\_).

#### 5.8.2 Constructor & Destructor Documentation

#### 5.8.2.1 Solution::Solution()

Default constructor of the class Solution.

#### 5.8.2.2 Solution::Solution (double obj1, double obj2)

Constructor of the class Solution.

#### **Parameters**

in	obj1	: A double which represents the value of the Solution w.r.t. objective 1.
in	obj2	: A double which represents the value of the Solution
		w.r.t. objective 2.

#### 5.8.2.3 Solution::∼Solution()

Destructor of the class Solution.

#### 5.8.3 Member Function Documentation

#### 5.8.3.1 double Solution::getObj1 ( ) const

Getter for the value w.r.t. objective 1 of this Solution.

#### Returns

A double as the value w.r.t. objective 1 of this Solution.

#### 5.8.3.2 double Solution::getObj2() const

Getter for the value w.r.t. objective 2 of this Solution.

#### **Returns**

A double as the value w.r.t. objective 2 of this Solution.

#### 5.8.3.3 void Solution::setObj1 ( double obj )

Setter for the value w.r.t objective 1 of this Solution.

#### **Parameters**

in	obj	: A double which represents the value of the Solution
		w.r.t objective 1.

#### 5.8.3.4 void Solution::setObj2 ( double obj )

Setter for the value w.r.t objective 2 of this Solution.

#### **Parameters**

in	<i>obj</i> : A double which represents the value of the Solution
	w.r.t objective 2.

#### 5.8.4 Friends And Related Function Documentation

**5.8.4.1** bool operator < (Solution s1, Solution s2) [related]

Operator overloading.

Overloading of the comparison operator < in order to compare two solutions.

#### **Parameters**

ſ	in	s1	: The first Solution to compare.
ſ	in	s2	: The second Solution to compare.

#### Returns

A boolean which gets TRUE if the value w.r.t. objective 1 of the Solution s1 is stricty lower to the value w.r.t. objective 1 of the Solution s2, FALSE otherwise.

#### 5.8.5 Member Data Documentation

**5.8.5.1 double Solution::obj1\_** [private]

Double which represents the value w.r.t. objective 1 of this Solution

5.8.5.2 double Solution::obj2\_ [private]

Double which represents the value w.r.t. objective 2 of this Solution

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

- /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Solution.hpp
- /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Solution.cpp

#### 5.9 ToFile Class Reference

```
Class to represent a ToFile.
#include "ToFile.hpp"
```

#### **Static Public Member Functions**

• static void removeFiles ()

Static method to erase file into folder /res.

• static void saveCorrelation (Data &data)

Static method to save correlation into a file.

- static void saveInitialBoxes (vector< Box \* > &vectorBox, Data &data)
  - Static method to save initial Boxes into a file.
- static void saveFilteringBoxes (vector< Box \* > &vectorBox, Data &data)

Static method to save filtered Boxes into a file.

static void saveReconstructionBoxes (vector< Box \* > &vectorBox, Data &data)

Static method to save recomposed Boxes into a file.

static void saveYN (list < Solution > &lsol, Data &data)

Static method to save solutions into a file.

#### 5.9.1 Detailed Description

Class to represent a ToFile.

This class represents a ToFile with all its attributes and methods.

#### 5.9.2 Member Function Documentation

```
5.9.2.1 void ToFile::removeFiles( ) [static]
```

Static method to erase file into folder /res.

5.9.2.2 void ToFile::saveCorrelation ( Data & data ) [static]

Static method to save correlation into a file.

#### **Parameters**

in	data	: A Data object which contains all the values of the in-
		stance.

### 5.9.2.3 void ToFile::saveFilteringBoxes ( vector< Box \* > & vectorBox, Data & data ) [static]

Static method to save filtered Boxes into a file.

#### **Parameters**

Ī	in	vectorBox	: A vector of filtered Boxes.
	in	data	: A Data object which contains all the values of the in-
			stance.

### 5.9.2.4 void ToFile::saveInitialBoxes ( vector< Box \* > & vectorBox, Data & data ) [static]

Static method to save initial Boxes into a file.

#### **Parameters**

in	vectorBox	: A vector of initial Boxes.
in	data	: A Data object which contains all the values of the in-
		stance.

### 5.9.2.5 void ToFile::saveReconstructionBoxes ( vector< Box \* > & vectorBox, Data & data ) [static]

Static method to save recomposed Boxes into a file.

#### **Parameters**

in	vectorBox	: A vector of recomposed Boxes.
in	data	: A Data object which contains all the values of the in-
		stance.

#### 5.9.2.6 void ToFile::saveYN ( list< Solution > & Isol, Data & data ) [static]

Static method to save solutions into a file.

#### **Parameters**

in	Isol	: A list of Sotuions.
in	data	: A Data object which contains all the values of the in-
		stance.

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

- $\bullet \ \ /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/ToFile.hpp$
- /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/ToFile.cpp

### **Chapter 6**

### **File Documentation**

6.1 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-Box.cpp File Reference

```
#include "Box.hpp"
```

#### **Functions**

- bool isDominatedBetweenTwoBoxes (Box \*box1, Box \*box2)
- bool isDominatedBetweenOrigins (Box \*box1, Box \*box2)
- void filterDominatedBoxes (vector < Box \* > &vectBox)
- bool isDominatedByItsOrigin (vector < Box \* > &vectBox, Box \*box)
- bool isDominatedByItsBox (vector < Box \* > &vectBox, Box \*box)

#### 6.1.1 Function Documentation

- 6.1.1.1 void filterDominatedBoxes ( vector < Box \* > & vectBox )
- **6.1.1.2** bool is Dominated Between Origins (Box \* box1, Box \* box2) [related]
- 6.1.1.3 bool isDominatedBetweenTwoBoxes ( Box \* box1, Box \* box2 ) [related]
- 6.1.1.4 bool isDominatedByItsBox ( vector < Box \* > & vectBox, Box \* box )
- 6.1.1.5 bool isDominatedByltsOrigin (vector < Box \* > & vectBox, Box \* box )

File Documentation

## 6.2 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-Box.hpp File Reference

Class of the Box.

```
#include <iostream> #include <cfloat> #include "Data.-
hpp"
```

#### Classes

52

• class Box

Class to represent a Box.

#### 6.2.1 Detailed Description

Class of the Box.

Author

Salim BOUROUGAA & Alban DERRIEN & Axel GRIMAULT & Xavier GANDIBLE-UX & Anthony PRZYBYLSKI

Date

28 August 2012

Version

1.1

Copyright

GNU General Public License

This class represents an object Box. A Box is a sub-space of dimension 2 defined in the objective space and characterized by at most two feasible solutions. These solutions correspond to the two lexicographic optimal solutions for the two objective functions when a set of Facility opened is considered.

# 6.3 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-Customer.cpp File Reference

```
#include "Customer.hpp"
```

### 6.4 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-Customer.hpp File

Reference 53

**Functions** 

- ostream & operator<< (ostream &out, const Customer \*cust)</li>
- 6.3.1 Function Documentation
- 6.3.1.1 ostream& operator << ( ostream & out, const Customer \* cust )
- 6.4 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Customer.hpp File Reference

```
Class of the Customer.
```

```
#include <iostream>
```

#### **Classes**

class Customer

Class to represent a Customer.

#### 6.4.1 Detailed Description

Class of the Customer.

Author

Salim BOUROUGAA & Alban DERRIEN & Axel GRIMAULT & Xavier GANDIBLE-UX & Anthony PRZYBYLSKI

Date

28 August 2012

Version

1.1

Copyright

This class represents a customer, or client, with its geographical coordinates and its costs of location w.r.t. to the two objectives in a FLP.

## 6.5 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Data.cpp File Reference

```
#include "Data.hpp"
```

# 6.6 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Data.hpp File Reference

```
Class of the Data.
```

```
#include "Facility.hpp" #include "Customer.hpp" #include
<iostream> #include <vector> #include <cfloat>
```

#### Classes

• class Data

Class to represent a Data.

#### 6.6.1 Detailed Description

Class of the Data.

**Author** 

Salim BOUROUGAA & Alban DERRIEN & Axel GRIMAULT & Xavier GANDIBLE-UX & Anthony PRZYBYLSKI

Date

28 August 2012

Version

1.1

Copyright

**GNU General Public License** 

This class will contains all the values, parameters of the instance. Especially, it contains the allocation cost between customers and facilities.

6.7 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-Facility.cpp File

Reference 6.7 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-Facility.cpp File Reference

```
#include "Facility.hpp"
```

#### 6.7.1 Function Documentation

6.7.1.1 ostream& operator << ( ostream & out, const Facility \* fac ) [related]

## 6.8 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-Facility.hpp File Reference

```
Class of the Facility.
#include <iostream>
```

#### Classes

• class Facility

Class to represent a Facility.

#### 6.8.1 Detailed Description

Class of the Facility.

Author

Salim BOUROUGAA & Alban DERRIEN & Axel GRIMAULT & Xavier GANDIBLE-UX & Anthony PRZYBYLSKI

Date

28 August 2012

Version

1.1

Copyright

GNU General Public License

This class represents a facility, or warehouse or service, with its geographical coordinates and its costs of location w.r.t. to the two objectives in a FLP.

### 6.9 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-Functions.cpp File Reference

```
#include "Functions.hpp"
```

#### **Functions**

long int createBox (vector< Box \* > &vectorBox, Data &data)

This method computes all the initial Boxes of our algorithm.

void addChildren (Box \*boxMother, vector< Box \* > &vBox)

This method adds children of a Box into a vector of Boxes.

void filter (vector< Box \* > &vectorBox, long int &nbToCompute, long int &nb-WithNeighbor)

This method filters all the Boxes of a vector of Box.

 void boxFiltering (vector< Box \* > &vectorBox, Data &data, long int &nbTo-Compute, long int &nbWithNeighbor)

This method splits a Box into two Boxes.

 void recomposition (vector< Box \* > &vectorBox, vector< Box \* > &vectorBox-Final, Data &data, long int &nbToCompute, long int &nbWithNeighbor)

This method recomposes a group of Boxes into an unique one.

void weightedSumOneStep (vector< Box \* > &vectorBox, Data &data)

Compute the weighted sum method to split a Box in two Boxes.

long int runLabelSetting (vector< Box \* > &vectorBox, Data &data)

This method runs the Label Setting algorithm.

void filterListSolution (list< Solution > &lsol)

Delete the solutions dominated.

• double computeCorrelation (Data &data)

This method computes the correlation.

float time\_ms\_Diff (timeval tvStart, timeval tvEnd)

This method computes the difference in milli-seconds (ms) between two times.

float time\_s\_Diff (timeval tvStart, timeval tvEnd)

This method computes the difference in seconds (ms) between two times.

### 6.10 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-Functions.hpp File Reference

#### A set of functions usefull for our software.

```
#include <vector> #include <map> #include <list> #include
<iostream> #include <cmath> #include "Data.hpp" #include
"Solution.hpp" #include "Box.hpp" #include "ToFile.hpp" x
#include "LabelSetting.hpp"
```

### 6.10 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-Functions.hpp File

Reference 57 Functions

long int createBox (vector< Box \* > &vectorBox, Data &data)

This method computes all the initial Boxes of our algorithm.

void addChildren (Box \*boxMother, vector< Box \* > &vBox)

This method adds children of a Box into a vector of Boxes.

void filter (vector< Box \* > &vectorBox, long int &nbToCompute, long int &nb-WithNeighbor)

This method filters all the Boxes of a vector of Box.

 void boxFiltering (vector< Box \* > &vectorBox, Data &data, long int &nbTo-Compute, long int &nbWithNeighbor)

This method splits a Box into two Boxes.

 void recomposition (vector< Box \* > &vectorBox, vector< Box \* > &vectorBox-Final, Data &data, long int &nbToCompute, long int &nbWithNeighbor)

This method recomposes a group of Boxes into an unique one.

void weightedSumOneStep (vector< Box \* > &vectorBox, Data &data)

Compute the weighted sum method to split a Box in two Boxes.

long int runLabelSetting (vector < Box \* > &vectorBox, Data &data)

This method runs the Label Setting algorithm.

void filterListSolution (list< Solution > &lsol)

Delete the solutions dominated.

· double computeCorrelation (Data &data)

This method computes the correlation.

• float time\_ms\_Diff (timeval tvStart, timeval tvEnd)

This method computes the difference in milli-seconds (ms) between two times.

float time\_s\_Diff (timeval tvStart, timeval tvEnd)

This method computes the difference in seconds (ms) between two times.

#### **Variables**

bool modeExport

Variable representing the Export mode.

#### 6.10.1 Detailed Description

A set of functions usefull for our software.

#### Author

Salim BOUROUGAA & Alban DERRIEN & Axel GRIMAULT & Xavier GANDIBLE-UX & Anthony PRZYBYLSKI

Date

28 August 2012

Version

1.1

#### Copyright

**GNU General Public License** 

This file groups all the functions for solving our problem which are not methods of Class.

### 6.11 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-LabelSetting.cpp File Reference

```
#include "LabelSetting.hpp"
```

### 6.12 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-LabelSetting.hpp File Reference

Class of the Box.

```
#include <iostream> #include <list> #include "Data.-
hpp" #include "Box.hpp" #include "Node.hpp" #include "-
Solution.hpp" #include "Functions.hpp"
```

#### **Classes**

· class LabelSetting

Class to represent a LabelSetting.

#### **Variables**

• bool modeVerbose

Variable representing the Verbose mode.

#### 6.12.1 Detailed Description

Class of the Box.

### /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Main.cpp File Reference 59

Salim BOUROUGAA & Alban DERRIEN & Axel GRIMAULT & Xavier GANDIBLE-UX & Anthony PRZYBYLSKI

Date

28 August 2012

Version

1.1

#### Copyright

**GNU General Public License** 

This class represents a LabelSetting which is the algorithm used to find all solutions.

## 6.13 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Main.cpp File Reference

#### Main of the software.

```
#include <sys/time.h> #include <stdexcept> #include <iostream> X
#include <string.h> #include <cfloat> #include <cmath>
#include <cstdlib> #include "Parser.hpp" #include "Data.-
hpp" #include "Functions.hpp" #include "ToFile.hpp"
```

#### **Functions**

• int main (int argc, char \*argv[])

This is the main of the software.

#### **Variables**

• bool modeVerbose = false

Variable representing the Verbose mode.

• bool modeExport = false

Variable representing the Export mode.

#### 6.13.1 Detailed Description

Main of the software.

**Author** 

Salim BOUROUGAA & Alban DERRIEN & Axel GRIMAULT & Xavier GANDIBLE-UX & Anthony PRZYBYLSKI

Date

28 August 2012

Version

1.1

Copyright

**GNU General Public License** 

# 6.14 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-Node.cpp File Reference

```
#include "Node.hpp"
```

# 6.15 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-Node.hpp File Reference

Class of the Node.

```
#include <iostream> #include <list> #include "Solution.-
hpp"
```

#### **Classes**

• class Node

Class to represent a Node.

#### **Variables**

· bool modeVerbose

Variable representing the Verbose mode.

/home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Parser.cpp File Reference 61

6.15.1 Detailed Description

Class of the Node.

Author

Salim BOUROUGAA & Alban DERRIEN & Axel GRIMAULT & Xavier GANDIBLE-UX & Anthony PRZYBYLSKI

Date

28 August 2012

Version

1.1

#### Copyright

**GNU General Public License** 

This class represents a Node which represents a level in the Label Setting algorithm.

# 6.16 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-Parser.cpp File Reference

```
#include "Parser.hpp"
```

# 6.17 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/Parser.hpp File Reference

```
Class of the Parser.
```

```
#include <fstream> #include <iostream> #include <math.-
h> #include "Data.hpp"
```

#### Classes

class Parser

Class to represent a Parser.

#### 6.17.1 Detailed Description

Class of the Parser.

**Author** 

Salim BOUROUGAA & Alban DERRIEN & Axel GRIMAULT & Xavier GANDIBLE-UX & Anthony PRZYBYLSKI

Date

28 August 2012

Version

1.1

Copyright

This class represents a parser to read data from an instance file.

# 6.18 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-Solution.cpp File Reference

```
#include "Solution.hpp"
```

#### 6.18.1 Function Documentation

6.18.1.1 bool operator < ( Solution s1, Solution s2 ) [related]

# 6.19 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/-Solution.hpp File Reference

Class of the Solution.

#### Classes

• class Solution

Class to represent a Solution.

### /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/ToFile.cpp File Reference 63

6.19.1 Detailed Description

Class of the Solution.

**Author** 

Salim BOUROUGAA & Alban DERRIEN & Axel GRIMAULT & Xavier GANDIBLE-UX & Anthony PRZYBYLSKI

Date

28 August 2012

Version

1.1

Copyright

This class represents a solution. In FLP, a solution is representes by two values respectively for the objective 1 and the objective 2.

# 6.20 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/ToFile.cpp File Reference

```
#include "ToFile.hpp"
```

## 6.21 /home/axel/Axel/Computing/svn/biuflp2012/trunk/wolseyaward/1.1/src/ToFile.hpp File Reference

Class of the ToFile.

```
#include <list> #include <iomanip> #include <vector>x
#include <stdio.h> #include <fstream> #include <dirent.-
h> #include <sys/types.h> #include <sstream> #include
<string> #include "Box.hpp" #include "Data.hpp" #include
"Solution.hpp"
```

#### Classes

· class ToFile

Class to represent a ToFile.

#### **Variables**

bool modeExport

Variable representing the Export mode.

#### 6.21.1 Detailed Description

Class of the ToFile.

**Author** 

Salim BOUROUGAA & Alban DERRIEN & Axel GRIMAULT & Xavier GANDIBLE-UX & Anthony PRZYBYLSKI

Date

28 August 2012

Version

1.1

Copyright

GNU General Public License

This class exports values, solutions to file in the folder res.