

Blis
trunk

Generated by Doxygen 1.8.1.2

Mon Mar 16 2015 20:20:30

Contents

1	Class Index	1
1.1	Class Hierarchy	1
2	Class Index	2
2.1	Class List	3
3	Class Documentation	4
3.1	BlisBranchObjectBilevel Class Reference	4
3.1.1	Detailed Description	4
3.1.2	Constructor & Destructor Documentation	5
3.1.3	Member Function Documentation	5
3.1.4	Member Data Documentation	6
3.2	BlisBranchObjectInt Class Reference	6
3.2.1	Detailed Description	7
3.2.2	Constructor & Destructor Documentation	7
3.2.3	Member Function Documentation	8
3.3	BlisBranchStrategyBilevel Class Reference	9
3.3.1	Detailed Description	10
3.3.2	Constructor & Destructor Documentation	10
3.3.3	Member Function Documentation	10
3.4	BlisBranchStrategyMaxInf Class Reference	10
3.4.1	Detailed Description	11
3.4.2	Constructor & Destructor Documentation	11
3.4.3	Member Function Documentation	12
3.5	BlisBranchStrategyPseudo Class Reference	12
3.5.1	Detailed Description	13
3.5.2	Constructor & Destructor Documentation	13
3.5.3	Member Function Documentation	13
3.6	BlisBranchStrategyRel Class Reference	14
3.6.1	Detailed Description	14
3.6.2	Constructor & Destructor Documentation	14
3.6.3	Member Function Documentation	15
3.7	BlisBranchStrategyStrong Class Reference	15
3.7.1	Detailed Description	16
3.7.2	Constructor & Destructor Documentation	16
3.7.3	Member Function Documentation	16

3.8	BlisConGenerator Class Reference	17
3.8.1	Detailed Description	19
3.8.2	Constructor & Destructor Documentation	19
3.8.3	Member Function Documentation	19
3.8.4	Member Data Documentation	22
3.9	BlisConstraint Class Reference	24
3.9.1	Detailed Description	25
3.9.2	Constructor & Destructor Documentation	25
3.9.3	Member Function Documentation	25
3.10	BlisHeuristic Class Reference	26
3.10.1	Detailed Description	27
3.10.2	Constructor & Destructor Documentation	28
3.10.3	Member Function Documentation	28
3.10.4	Member Data Documentation	29
3.11	BlisHeurRound Class Reference	30
3.11.1	Detailed Description	31
3.11.2	Constructor & Destructor Documentation	31
3.11.3	Member Function Documentation	31
3.11.4	Member Data Documentation	32
3.12	BlisMessage Class Reference	32
3.12.1	Detailed Description	32
3.13	BlisModel Class Reference	32
3.13.1	Detailed Description	40
3.13.2	Constructor & Destructor Documentation	41
3.13.3	Member Function Documentation	41
3.13.4	Member Data Documentation	51
3.14	BlisNodeDesc Class Reference	56
3.14.1	Detailed Description	57
3.14.2	Constructor & Destructor Documentation	57
3.14.3	Member Function Documentation	57
3.15	BlisObjectInt Class Reference	58
3.15.1	Detailed Description	60
3.15.2	Constructor & Destructor Documentation	60
3.15.3	Member Function Documentation	60
3.15.4	Member Data Documentation	62
3.16	BlisParams Class Reference	62
3.16.1	Detailed Description	63

3.16.2	Member Enumeration Documentation	64
3.16.3	Constructor & Destructor Documentation	65
3.16.4	Member Function Documentation	65
3.17	BlisPresolve Class Reference	66
3.17.1	Detailed Description	66
3.18	BlisPseudocost Class Reference	66
3.18.1	Detailed Description	67
3.18.2	Constructor & Destructor Documentation	67
3.18.3	Member Function Documentation	68
3.19	BlisSolution Class Reference	69
3.19.1	Detailed Description	69
3.19.2	Constructor & Destructor Documentation	69
3.19.3	Member Function Documentation	70
3.20	BlisStrong Struct Reference	70
3.20.1	Detailed Description	70
3.21	BlisTreeNode Class Reference	70
3.21.1	Detailed Description	71
3.21.2	Constructor & Destructor Documentation	72
3.21.3	Member Function Documentation	72
3.22	BlisVariable Class Reference	73
3.22.1	Detailed Description	74
3.22.2	Member Function Documentation	74

1 Class Index

1.1 Class Hierarchy

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

```

std::basic_fstream< char >
std::basic_fstream< wchar_t >
std::basic_ifstream< char >
std::basic_ifstream< wchar_t >
std::basic_ios< char >
std::basic_ios< wchar_t >
std::basic_iostream< char >
std::basic_iostream< wchar_t >
std::basic_istream< char >
std::basic_istream< wchar_t >
std::basic_istreamstream< char >
std::basic_istreamstream< wchar_t >
std::basic_ofstream< char >
std::basic_ofstream< wchar_t >

```

[std::basic_ostream< char >](#)
[std::basic_ostream< wchar_t >](#)
[std::basic_ostringstream< char >](#)
[std::basic_ostringstream< wchar_t >](#)
[std::basic_string< char >](#)
[std::basic_string< wchar_t >](#)
[std::basic_stringstream< char >](#)
[std::basic_stringstream< wchar_t >](#)

BlisBranchObjectBilevel	4
BlisBranchObjectInt	6
BlisBranchStrategyBilevel	9
BlisBranchStrategyMaxInf	10
BlisBranchStrategyPseudo	12
BlisBranchStrategyRel	14
BlisBranchStrategyStrong	15
BlisConGenerator	17
BlisConstraint	24
BlisHeuristic	26
BlisHeurRound	30
BlisMessage	32
BlisModel	32
BlisNodeDesc	56
BlisObjectInt	58
BlisParams	62
BlisPresolve	66
BlisPseudocost	66
BlisSolution	69
BlisStrong	70
BlisTreeNode	70
BlisVariable	73

2 Class Index

2.1 Class List

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

BlisBranchObjectBilevel	4
BlisBranchObjectInt	6
BlisBranchStrategyBilevel This class implements maximum infeasibility branching	9
BlisBranchStrategyMaxInf This class implements maximum infeasibility branching	10
BlisBranchStrategyPseudo Blis branching strategy	12
BlisBranchStrategyRel Blis branching strategy	14
BlisBranchStrategyStrong This class implements strong branching	15
BlisConGenerator Interface between Blis and Cut Generation Library	17
BlisConstraint	24
BlisHeuristic Heuristic base class	26
BlisHeurRound Rounding Heuristic	30
BlisMessage	32
BlisModel	32
BlisNodeDesc	56
BlisObjectInt	58
BlisParams	62
BlisPresolve A interface to Osi/Coin Presolve	66
BlisPseudocost	66
BlisSolution This class contains the solutions generated by the LP solver (either primal or dual)	69
BlisStrong	70
BlisTreeNode This is the class in which we are finally able to concretely define the bounding procedure	70
BlisVariable	73

3 Class Documentation

3.1 BlisBranchObjectBilevel Class Reference

Public Member Functions

- [BlisBranchObjectBilevel](#) ()
Default constructor.
- [BlisBranchObjectBilevel](#) (BcpsModel *model)
Another useful constructor.
- [BlisBranchObjectBilevel](#) (const [BlisBranchObjectBilevel](#) &rhs)
Copy constructor.
- [BlisBranchObjectBilevel](#) & [operator=](#) (const [BlisBranchObjectBilevel](#) &rhs)
Assignment operator.
- virtual BcpsBranchObject * [clone](#) () const
Clone.
- virtual [~BlisBranchObjectBilevel](#) ()
Destructor.
- std::deque< int > * [getBranchingSet](#) () const
Get a pointer to the branching set.
- void [addToBranchingSet](#) (int item)
Get a pointer to the branching set.
- virtual double [branch](#) (bool normalBranch=false)
Set the bounds for the variable according to the current arm of the branch and advances the object state to the next arm.
- virtual void [print](#) (bool normalBranch)
Print something about branch - only if log level high.
- virtual AlpsReturnStatus [encode](#) (AlpsEncoded *encoded) const
Pack to an encoded object.
- virtual AlpsReturnStatus [decode](#) (AlpsEncoded &encoded)
Unpack a branching object from an encoded object.

Protected Member Functions

- AlpsReturnStatus [encodeBlis](#) (AlpsEncoded *encoded) const
Pack Blis portion to an encoded object.
- AlpsReturnStatus [decodeBlis](#) (AlpsEncoded &encoded)
Unpack Blis portion from an encoded object.

Protected Attributes

- std::deque< int > * [branchingSet_](#)
The indices of variables in the branching set.

3.1.1 Detailed Description

Definition at line 38 of file [BlisBranchObjectBilevel.h](#).

3.1.2 Constructor & Destructor Documentation

3.1.2.1 `BlisBranchObjectBilevel::BlisBranchObjectBilevel () [inline]`

Default constructor.

Definition at line 48 of file `BlisBranchObjectBilevel.h`.

3.1.2.2 `BlisBranchObjectBilevel::BlisBranchObjectBilevel (BcpsModel * model) [inline]`

Another useful constructor.

Definition at line 55 of file `BlisBranchObjectBilevel.h`.

3.1.2.3 `BlisBranchObjectBilevel::BlisBranchObjectBilevel (const BlisBranchObjectBilevel & rhs) [inline]`

Copy constructor.

Definition at line 62 of file `BlisBranchObjectBilevel.h`.

3.1.2.4 `virtual BlisBranchObjectBilevel::~BlisBranchObjectBilevel () [inline], [virtual]`

Destructor.

Definition at line 75 of file `BlisBranchObjectBilevel.h`.

3.1.3 Member Function Documentation

3.1.3.1 `BlisBranchObjectBilevel& BlisBranchObjectBilevel::operator= (const BlisBranchObjectBilevel & rhs)`

Assignment operator.

3.1.3.2 `virtual BcpsBranchObject* BlisBranchObjectBilevel::clone () const [inline], [virtual]`

Clone.

Definition at line 70 of file `BlisBranchObjectBilevel.h`.

3.1.3.3 `virtual double BlisBranchObjectBilevel::branch (bool normalBranch = false) [virtual]`

Set the bounds for the variable according to the current arm of the branch and advances the object state to the next arm.

Returns change in guessed objective on next branch.

3.1.3.4 `virtual void BlisBranchObjectBilevel::print (bool normalBranch) [virtual]`

Print something about branch - only if log level high.

3.1.3.5 `AlpsReturnStatus BlisBranchObjectBilevel::encodeBlis (AlpsEncoded * encoded) const [inline], [protected]`

Pack Blis portion to an encoded object.

Definition at line 94 of file `BlisBranchObjectBilevel.h`.

3.1.3.6 `AlpsReturnStatus BlisBranchObjectBilevel::decodeBlis (AlpsEncoded & encoded) [inline], [protected]`

Unpack Blis portion from an encoded object.

Definition at line 101 of file BlisBranchObjectBilevel.h.

3.1.3.7 `virtual AlpsReturnStatus BlisBranchObjectBilevel::encode (AlpsEncoded * encoded) const` `[inline]`,
`[virtual]`

Pack to an encoded object.

Definition at line 109 of file BlisBranchObjectBilevel.h.

3.1.3.8 `virtual AlpsReturnStatus BlisBranchObjectBilevel::decode (AlpsEncoded & encoded)` `[inline]`, `[virtual]`

Unpack a branching object from an encoded object.

Definition at line 119 of file BlisBranchObjectBilevel.h.

3.1.4 Member Data Documentation

3.1.4.1 `std::deque<int>* BlisBranchObjectBilevel::branchingSet_` `[protected]`

The indices of variables in the branching set.

Definition at line 43 of file BlisBranchObjectBilevel.h.

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

- BlisBranchObjectBilevel.h

3.2 BlisBranchObjectInt Class Reference

Public Member Functions

- [BlisBranchObjectInt](#) ()
Default constructor.
- [BlisBranchObjectInt](#) ([BlisModel](#) *model, int varInd, int direction, double value)
Construct a branching object, which branching on variable varInd.
- [BlisBranchObjectInt](#) ([BlisModel](#) *model, int varInd, int intScore, double dblScore, int direction, double value)
Construct a branching object, which branching on variable varInd.
- [BlisBranchObjectInt](#) ([BlisModel](#) *model, int varInd, int direction, double lowerValue, double upperValue)
Create a degenerate branching object.
- [BlisBranchObjectInt](#) (const [BlisBranchObjectInt](#) &)
Copy constructor.
- [BlisBranchObjectInt](#) & `operator=` (const [BlisBranchObjectInt](#) &rhs)
Assignment operator.
- virtual [BcpsBranchObject](#) * `clone` () const
Clone.
- virtual `~BlisBranchObjectInt` ()
Destructor.
- virtual double `branch` (bool normalBranch=false)
Set the bounds for the variable according to the current arm of the branch and advances the object state to the next arm.
- virtual void `print` (bool normalBranch)
Print something about branch - only if log level high.
- const double * `getDown` () const

Get down arm bounds.

- const double * [getUp](#) () const

Get upper arm bounds.

- virtual AlpsReturnStatus [encode](#) (AlpsEncoded *encoded) const

Pack to an encoded object.

- virtual AlpsReturnStatus [decode](#) (AlpsEncoded &encoded)

Unpack a branching object from an encoded object.

Protected Member Functions

- AlpsReturnStatus [encodeBlis](#) (AlpsEncoded *encoded) const

Pack Blis portion to an encoded object.

- AlpsReturnStatus [decodeBlis](#) (AlpsEncoded &encoded)

Unpack Blis portion from an encoded object.

Protected Attributes

- double [down_](#) [2]

Down_[0]: the lower bound of down arm; Down_[1]: the upper bound of down arm;.

- double [up_](#) [2]

Up_[0]: the lower bound of upper arm; Up_[1]: the upper bound of upper arm;.

3.2.1 Detailed Description

Definition at line 38 of file BlisBranchObjectInt.h.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 BlisBranchObjectInt::BlisBranchObjectInt () [inline]

Default constructor.

Definition at line 53 of file BlisBranchObjectInt.h.

3.2.2.2 BlisBranchObjectInt::BlisBranchObjectInt (BlisModel * model, int varInd, int direction, double value) [inline]

Construct a branching object, which branching on variable varInd.

Parameters

<i>varInd</i>	the index of integer variable in object set
<i>direction</i>	the direction of first branching: 1(up), -1(down)
<i>value</i>	the fractional solution value of variable varInd

Definition at line 69 of file BlisBranchObjectInt.h.

3.2.2.3 BlisBranchObjectInt::BlisBranchObjectInt (BlisModel * model, int varInd, int intScore, double dblScore, int direction, double value) [inline]

Construct a branching object, which branching on variable varInd.

Parameters

<i>varInd</i>	the index of integer variable in object set
<i>intScore</i>	the integer score/goodness
<i>dblScore</i>	the double score/goodness
<i>direction</i>	the direction of first branching: 1(up), -1(down)
<i>value</i>	the fractional solution value of variable varInd

Definition at line 91 of file BlisBranchObjectInt.h.

3.2.2.4 `BlisBranchObjectInt::BlisBranchObjectInt (BlisModel * model, int varInd, int direction, double lowerValue, double upperValue) [inline]`

Create a degenerate branching object.

Specifies a 'one-direction branch'. Calling `branch()` for this object will always result in $\text{lowerValue} \leq x \leq \text{upperValue}$. Used to fix a variable when $\text{lowerValue} = \text{upperValue}$.

Definition at line 113 of file BlisBranchObjectInt.h.

3.2.2.5 `BlisBranchObjectInt::BlisBranchObjectInt (const BlisBranchObjectInt &)`

Copy constructor.

3.2.2.6 `virtual BlisBranchObjectInt::~~BlisBranchObjectInt () [inline],[virtual]`

Destructor.

Definition at line 141 of file BlisBranchObjectInt.h.

3.2.3 Member Function Documentation

3.2.3.1 `BlisBranchObjectInt& BlisBranchObjectInt::operator= (const BlisBranchObjectInt & rhs)`

Assignment operator.

3.2.3.2 `virtual BcpsBranchObject* BlisBranchObjectInt::clone () const [inline],[virtual]`

Clone.

Definition at line 136 of file BlisBranchObjectInt.h.

3.2.3.3 `virtual double BlisBranchObjectInt::branch (bool normalBranch = false) [virtual]`

Set the bounds for the variable according to the current arm of the branch and advances the object state to the next arm.

Returns change in guessed objective on next branch.

3.2.3.4 `virtual void BlisBranchObjectInt::print (bool normalBranch) [virtual]`

Print something about branch - only if log level high.

3.2.3.5 `const double* BlisBranchObjectInt::getDown () const [inline]`

Get down arm bounds.

Definition at line 152 of file BlisBranchObjectInt.h.

3.2.3.6 `const double* BlisBranchObjectInt::getUp () const` `[inline]`

Get upper arm bounds.

Definition at line 155 of file `BlisBranchObjectInt.h`.

3.2.3.7 `AlpsReturnStatus BlisBranchObjectInt::encodeBlis (AlpsEncoded * encoded) const` `[inline],[protected]`

Pack Blis portion to an encoded object.

Definition at line 160 of file `BlisBranchObjectInt.h`.

3.2.3.8 `AlpsReturnStatus BlisBranchObjectInt::decodeBlis (AlpsEncoded & encoded)` `[inline],[protected]`

Unpack Blis portion from an encoded object.

Definition at line 176 of file `BlisBranchObjectInt.h`.

3.2.3.9 `virtual AlpsReturnStatus BlisBranchObjectInt::encode (AlpsEncoded * encoded) const` `[inline],[virtual]`

Pack to an encoded object.

Definition at line 193 of file `BlisBranchObjectInt.h`.

3.2.3.10 `virtual AlpsReturnStatus BlisBranchObjectInt::decode (AlpsEncoded & encoded)` `[inline],[virtual]`

Unpack a branching object from an encoded object.

Definition at line 203 of file `BlisBranchObjectInt.h`.

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

- `BlisBranchObjectInt.h`

3.3 BlisBranchStrategyBilevel Class Reference

This class implements maximum infeasibility branching.

```
#include <BlisBranchStrategyBilevel.h>
```

Public Member Functions

- [BlisBranchStrategyBilevel](#) ()
Bilevel Constructor.
- [BlisBranchStrategyBilevel](#) ([BlisModel](#) *model)
Bilevel Constructor.
- virtual [~BlisBranchStrategyBilevel](#) ()
Destructor.
- [BlisBranchStrategyBilevel](#) (const [BlisBranchStrategyBilevel](#) &)
Copy constructor.
- virtual [BcpsBranchStrategy](#) * [clone](#) () const
Clone a branching strategy.
- virtual int [createCandBranchObjects](#) (int numPassesLeft, double ub)
Create a set of candidate branching objects.
- virtual int [betterBranchObject](#) ([BcpsBranchObject](#) *thisOne, [BcpsBranchObject](#) *bestSoFar)
Compare branching object thisOne to bestSoFar.

3.3.1 Detailed Description

This class implements maximum infeasibility branching.

Definition at line 32 of file BlisBranchStrategyBilevel.h.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 BlisBranchStrategyBilevel::BlisBranchStrategyBilevel () [inline]

Bilevel Constructor.

Definition at line 42 of file BlisBranchStrategyBilevel.h.

3.3.2.2 BlisBranchStrategyBilevel::BlisBranchStrategyBilevel (BlisModel * *model*) [inline]

Bilevel Constructor.

Definition at line 47 of file BlisBranchStrategyBilevel.h.

3.3.2.3 virtual BlisBranchStrategyBilevel::~BlisBranchStrategyBilevel () [inline],[virtual]

Destructor.

Definition at line 52 of file BlisBranchStrategyBilevel.h.

3.3.2.4 BlisBranchStrategyBilevel::BlisBranchStrategyBilevel (const BlisBranchStrategyBilevel &)

Copy constructor.

3.3.3 Member Function Documentation

3.3.3.1 virtual BcpsBranchStrategy* BlisBranchStrategyBilevel::clone () const [inline],[virtual]

Clone a branching strategy.

Definition at line 58 of file BlisBranchStrategyBilevel.h.

3.3.3.2 virtual int BlisBranchStrategyBilevel::createCandBranchObjects (int *numPassesLeft*, double *ub*) [virtual]

Create a set of candidate branching objects.

3.3.3.3 virtual int BlisBranchStrategyBilevel::betterBranchObject (BcpsBranchObject * *thisOne*, BcpsBranchObject * *bestSoFar*) [virtual]

Compare branching object *thisOne* to *bestSoFar*.

If *thisOne* is better than *bestObject*, return branching direction(1 or -1), otherwise return 0. If *bestSoFar* is NULL, then always return branching direction(1 or -1).

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

- BlisBranchStrategyBilevel.h

3.4 BlisBranchStrategyMaxInf Class Reference

This class implements maximum infeasibility branching.

```
#include <BlisBranchStrategyMaxInf.h>
```

Public Member Functions

- [BlisBranchStrategyMaxInf](#) ()
MaxInf Constructor.
- [BlisBranchStrategyMaxInf](#) ([BlisModel](#) *model)
MaxInf Constructor.
- virtual [~BlisBranchStrategyMaxInf](#) ()
Destructor.
- [BlisBranchStrategyMaxInf](#) (const [BlisBranchStrategyMaxInf](#) &)
Copy constructor.
- virtual [BcpsBranchStrategy](#) * [clone](#) () const
Clone a branching strategy.
- virtual int [createCandBranchObjects](#) (int numPassesLeft, double ub)
Create a set of candidate branching objects.
- virtual int [betterBranchObject](#) ([BcpsBranchObject](#) *thisOne, [BcpsBranchObject](#) *bestSoFar)
Compare branching object thisOne to bestSoFar.

3.4.1 Detailed Description

This class implements maximum infeasibility branching.

Definition at line 32 of file [BlisBranchStrategyMaxInf.h](#).

3.4.2 Constructor & Destructor Documentation

3.4.2.1 [BlisBranchStrategyMaxInf::BlisBranchStrategyMaxInf](#) () [[inline](#)]

MaxInf Constructor.

Definition at line 42 of file [BlisBranchStrategyMaxInf.h](#).

3.4.2.2 [BlisBranchStrategyMaxInf::BlisBranchStrategyMaxInf](#) ([BlisModel](#) * *model*) [[inline](#)]

MaxInf Constructor.

Definition at line 47 of file [BlisBranchStrategyMaxInf.h](#).

3.4.2.3 virtual [BlisBranchStrategyMaxInf::~~BlisBranchStrategyMaxInf](#) () [[inline](#)], [[virtual](#)]

Destructor.

Definition at line 52 of file [BlisBranchStrategyMaxInf.h](#).

3.4.2.4 [BlisBranchStrategyMaxInf::BlisBranchStrategyMaxInf](#) (const [BlisBranchStrategyMaxInf](#) &)

Copy constructor.

3.4.3 Member Function Documentation

3.4.3.1 `virtual BcpsBranchStrategy* BlisBranchStrategyMaxInf::clone () const [inline],[virtual]`

Clone a branching strategy.

Definition at line 58 of file BlisBranchStrategyMaxInf.h.

3.4.3.2 `virtual int BlisBranchStrategyMaxInf::createCandBranchObjects (int numPassesLeft, double ub) [virtual]`

Create a set of candidate branching objects.

3.4.3.3 `virtual int BlisBranchStrategyMaxInf::betterBranchObject (BcpsBranchObject * thisOne, BcpsBranchObject * bestSoFar) [virtual]`

Compare branching object thisOne to bestSoFar.

If thisOne is better than bestObject, return branching direction(1 or -1), otherwise return 0. If bestSoFar is NULL, then always return branching direction(1 or -1).

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

- BlisBranchStrategyMaxInf.h

3.5 BlisBranchStrategyPseudo Class Reference

Blis branching strategy.

```
#include <BlisBranchStrategyPseudo.h>
```

Public Member Functions

- [BlisBranchStrategyPseudo \(\)](#)
Default Constructor.
- [BlisBranchStrategyPseudo \(BlisModel *model, int rel\)](#)
Useful Constructor.
- `virtual ~BlisBranchStrategyPseudo ()`
Destructor.
- [BlisBranchStrategyPseudo \(const BlisBranchStrategyPseudo &\)](#)
Copy constructor.
- `void setReliability (int rel)`
Set reliability.
- `virtual BcpsBranchStrategy * clone () const`
Clone a branching strategy.
- `virtual int betterBranchObject (BcpsBranchObject *thisOne, BcpsBranchObject *bestSoFar)`
Compare branching object thisOne to bestSoFar.
- `int createCandBranchObjects (int numPassesLeft, double ub)`
Create a set of candidate branching objects.

3.5.1 Detailed Description

Blis branching strategy.

This class implements pseudocost branching.

Definition at line 40 of file BlisBranchStrategyPseudo.h.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 BlisBranchStrategyPseudo::BlisBranchStrategyPseudo () [inline]

Default Constructor.

Definition at line 51 of file BlisBranchStrategyPseudo.h.

3.5.2.2 BlisBranchStrategyPseudo::BlisBranchStrategyPseudo (BlisModel * model, int rel) [inline]

Useful Constructor.

Definition at line 57 of file BlisBranchStrategyPseudo.h.

3.5.2.3 virtual BlisBranchStrategyPseudo::~~BlisBranchStrategyPseudo () [inline],[virtual]

Destructor.

Definition at line 64 of file BlisBranchStrategyPseudo.h.

3.5.2.4 BlisBranchStrategyPseudo::BlisBranchStrategyPseudo (const BlisBranchStrategyPseudo &)

Copy constructor.

3.5.3 Member Function Documentation

3.5.3.1 void BlisBranchStrategyPseudo::setReliability (int rel) [inline]

Set reliability.

Definition at line 70 of file BlisBranchStrategyPseudo.h.

3.5.3.2 virtual BcpsBranchStrategy* BlisBranchStrategyPseudo::clone () const [inline],[virtual]

Clone a branching strategy.

Definition at line 73 of file BlisBranchStrategyPseudo.h.

3.5.3.3 virtual int BlisBranchStrategyPseudo::betterBranchObject (BcpsBranchObject * thisOne, BcpsBranchObject * bestSoFar) [virtual]

Compare branching object thisOne to bestSoFar.

If thisOne is better than bestObject, return branching direction(1 or -1), otherwise return 0. If bestSoFar is NULL, then always return branching direction(1 or -1).

3.5.3.4 int BlisBranchStrategyPseudo::createCandBranchObjects (int numPassesLeft, double ub)

Create a set of candidate branching objects.

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

- [BlisBranchStrategyPseudo.h](#)

3.6 BlisBranchStrategyRel Class Reference

Blis branching strategy.

```
#include <BlisBranchStrategyRel.h>
```

Public Member Functions

- [BlisBranchStrategyRel](#) ()
Default Constructor.
- [BlisBranchStrategyRel](#) ([BlisModel](#) *model, int rel)
Useful Constructor.
- virtual [~BlisBranchStrategyRel](#) ()
Destructor.
- [BlisBranchStrategyRel](#) (const [BlisBranchStrategyRel](#) &)
Copy constructor.
- void [setReliability](#) (int rel)
Set reliability.
- virtual [BcpsBranchStrategy](#) * [clone](#) () const
Clone a branching strategy.
- virtual int [betterBranchObject](#) ([BcpsBranchObject](#) *thisOne, [BcpsBranchObject](#) *bestSoFar)
Compare branching object thisOne to bestSoFar.
- int [createCandBranchObjects](#) (int numPassesLeft, double ub)
Create a set of candidate branching objects.

3.6.1 Detailed Description

Blis branching strategy.

This class implements reliability branching.

Definition at line 40 of file [BlisBranchStrategyRel.h](#).

3.6.2 Constructor & Destructor Documentation

3.6.2.1 [BlisBranchStrategyRel::BlisBranchStrategyRel](#) () [[inline](#)]

Default Constructor.

Definition at line 51 of file [BlisBranchStrategyRel.h](#).

3.6.2.2 [BlisBranchStrategyRel::BlisBranchStrategyRel](#) ([BlisModel](#) * model, int rel) [[inline](#)]

Useful Constructor.

Definition at line 57 of file [BlisBranchStrategyRel.h](#).

3.6.2.3 `virtual BlisBranchStrategyRel::~BlisBranchStrategyRel () [inline],[virtual]`

Destructor.

Definition at line 64 of file `BlisBranchStrategyRel.h`.

3.6.2.4 `BlisBranchStrategyRel::BlisBranchStrategyRel (const BlisBranchStrategyRel &)`

Copy constructor.

3.6.3 Member Function Documentation

3.6.3.1 `void BlisBranchStrategyRel::setReliability (int rel) [inline]`

Set reliability.

Definition at line 70 of file `BlisBranchStrategyRel.h`.

3.6.3.2 `virtual BcpsBranchStrategy* BlisBranchStrategyRel::clone () const [inline],[virtual]`

Clone a branching strategy.

Definition at line 73 of file `BlisBranchStrategyRel.h`.

3.6.3.3 `virtual int BlisBranchStrategyRel::betterBranchObject (BcpsBranchObject * thisOne, BcpsBranchObject * bestSoFar) [virtual]`

Compare branching object `thisOne` to `bestSoFar`.

If `thisOne` is better than `bestObject`, return branching direction(1 or -1), otherwise return 0. If `bestSoFar` is NULL, then always return branching direction(1 or -1).

3.6.3.4 `int BlisBranchStrategyRel::createCandBranchObjects (int numPassesLeft, double ub)`

Create a set of candidate branching objects.

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

- `BlisBranchStrategyRel.h`

3.7 BlisBranchStrategyStrong Class Reference

This class implements strong branching.

```
#include <BlisBranchStrategyStrong.h>
```

Public Member Functions

- [BlisBranchStrategyStrong \(\)](#)
Strong Constructor.
- [BlisBranchStrategyStrong \(BlisModel *model\)](#)
Strong Constructor.
- `virtual ~BlisBranchStrategyStrong ()`
Destructor.
- [BlisBranchStrategyStrong \(const BlisBranchStrategyStrong &\)](#)

Copy constructor.

- virtual BcpsBranchStrategy * **clone** () const

Clone a branching strategy.

- virtual int **createCandBranchObjects** (int numPassesLeft, double ub)

Create a set of candidate branching objects.

- virtual int **betterBranchObject** (BcpsBranchObject *thisOne, BcpsBranchObject *bestSoFar)

Compare branching object thisOne to bestSoFar.

3.7.1 Detailed Description

This class implements strong branching.

Definition at line 57 of file BlisBranchStrategyStrong.h.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 BlisBranchStrategyStrong::BlisBranchStrategyStrong () [inline]

Strong Constructor.

Definition at line 67 of file BlisBranchStrategyStrong.h.

3.7.2.2 BlisBranchStrategyStrong::BlisBranchStrategyStrong (BlisModel * *model*) [inline]

Strong Constructor.

Definition at line 72 of file BlisBranchStrategyStrong.h.

3.7.2.3 virtual BlisBranchStrategyStrong::~~BlisBranchStrategyStrong () [inline],[virtual]

Destructor.

Definition at line 78 of file BlisBranchStrategyStrong.h.

3.7.2.4 BlisBranchStrategyStrong::BlisBranchStrategyStrong (const BlisBranchStrategyStrong &)

Copy constructor.

3.7.3 Member Function Documentation

3.7.3.1 virtual BcpsBranchStrategy* BlisBranchStrategyStrong::clone () const [inline],[virtual]

Clone a branching strategy.

Definition at line 84 of file BlisBranchStrategyStrong.h.

3.7.3.2 virtual int BlisBranchStrategyStrong::createCandBranchObjects (int *numPassesLeft*, double *ub*) [virtual]

Create a set of candidate branching objects.

3.7.3.3 virtual int BlisBranchStrategyStrong::betterBranchObject (BcpsBranchObject * *thisOne*, BcpsBranchObject * *bestSoFar*) [virtual]

Compare branching object thisOne to bestSoFar.

If thisOne is better than bestObject, return branching direction(1 or -1), otherwise return 0. If bestSorFar is NULL, then always return branching direction(1 or -1).

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

- BlisBranchStrategyStrong.h

3.8 BlisConGenerator Class Reference

Interface between Blis and Cut Generation Library.

```
#include <BlisConGenerator.h>
```

Public Member Functions

Constructors and destructors

- [BlisConGenerator](#) ()
Default constructor.
- [BlisConGenerator](#) ([BlisModel](#) *model, [CglCutGenerator](#) *generator, const char *name=NULL, [BlisCutStrategy](#) strategy=[BlisCutStrategyAuto](#), int cutGenerationFrequency_=1, bool normal=true, bool atSolution=false, bool infeasible=false)
Useful constructor.
- [BlisConGenerator](#) (const [BlisConGenerator](#) &)
Copy constructor.
- [BlisConGenerator](#) & operator= (const [BlisConGenerator](#) &rhs)
Assignment operator.
- virtual ~[BlisConGenerator](#) ()
Destructor.

Generate Constraints

- virtual bool [generateConstraints](#) ([BcpsConstraintPool](#) &conPool)
Generate cons for the client model.

Gets and sets

- [BlisModel](#) * [getModel](#) ()
Set the client model.
- void [setModel](#) ([BlisModel](#) *m)
Set the model.
- void [refreshModel](#) ([BlisModel](#) *model)
Refresh the model.
- void [setName](#) (const char *str)
return name of generator.
- std::string [name](#) () const
return name of generator.
- void [setStrategy](#) ([BlisCutStrategy](#) value)
Set the con generation strategy.
- [BlisCutStrategy](#) [strategy](#) () const
Get the con generation interval.
- void [setCutGenerationFreq](#) (int freq)
Set the con generation strategy.
- int [cutGenerationFreq](#) () const

- *Get the con generation interval.*
- bool `normal` () const
- *Get whether the con generator should be called in the normal place.*
- void `setNormal` (bool value)
- *Set whether the con generator should be called in the normal place.*
- bool `atSolution` () const
- *Get whether the con generator should be called when a solution is found.*
- void `setAtSolution` (bool value)
- *Set whether the con generator should be called when a solution is found.*
- bool `whenInfeasible` () const
- *Get whether the con generator should be called when the subproblem is found to be infeasible.*
- void `setWhenInfeasible` (bool value)
- *Set whether the con generator should be called when the subproblem is found to be infeasible.*
- CglCutGenerator * `generator` () const
- *Get the CglCutGenerator bound to this BlisConGenerator.*
- int `numConsGenerated` ()
- *Get number of generated cons.*
- void `addNumConsGenerated` (int n)
- *Increase the number of generated cons.*
- int `numConsUsed` ()
- *Get number of used cons.*
- void `addNumConsUsed` (int n)
- *Increase the number of generated cons.*
- double `time` () const
- *Cpu time used.*
- void `addTime` (double t)
- *Increase Cpu time used.*
- int `calls` () const
- *Number called.*
- void `addCalls` (int n=1)
- *Increase the number of called.*
- int `noConsCalls` () const
- *Number called and no cons found.*
- void `addNoConsCalls` (int n=1)
- *Increase the number of no cons called.*

Protected Attributes

- BlisModel * `model_`
- *The client model.*
- CglCutGenerator * `generator_`
- *The CglCutGenerator object.*
- BlisCutStrategy `strategy_`
- *When to call CglCutGenerator::generateCuts routine.*
- int `cutGenerationFrequency_`
- *The frequency of calls to the cut generator.*
- std::string `name_`
- *Name of generator.*
- bool `normal_`
- *Whether to call the generator in the normal place.*
- bool `atSolution_`
- *Whether to call the generator when a new solution is found.*

- bool `whenInfeasible_`
Whether to call generator when a subproblem is found to be infeasible.
- int `numConsGenerated_`
Number of cons generated.
- int `numConsUsed_`
Number of cons used.
- double `time_`
Used CPU/User time.
- int `calls_`
The times of calling this generator.
- int `noConsCalls_`
The times of calling this generator and no cons found.

3.8.1 Detailed Description

Interface between Blis and Cut Generation Library.

`BlisConGenerator` is intended to provide an intelligent interface between Blis and the cutting plane algorithms in the CGL. A `BlisConGenerator` is bound to a `CglCutGenerator` and to an `BlisModel`. It contains parameters which control when and how the `generateCuts` method of the `CglCutGenerator` will be called.

The builtin decision criteria available to use when deciding whether to generate cons are: at root, automatic, every X nodes, when a solution is found, and when a subproblem is found to be infeasible.

Definition at line 58 of file `BlisConGenerator.h`.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 `BlisConGenerator::BlisConGenerator () [inline]`

Default constructor.

Definition at line 119 of file `BlisConGenerator.h`.

3.8.2.2 `BlisConGenerator::BlisConGenerator (BlisModel * model, CglCutGenerator * generator, const char * name = NULL, BlisCutStrategy strategy = BlisCutStrategyAuto, int cutGenerationFrequency_ = 1, bool normal = true, bool atSolution = false, bool infeasible = false)`

Useful constructor.

3.8.2.3 `BlisConGenerator::BlisConGenerator (const BlisConGenerator &)`

Copy constructor.

3.8.2.4 `virtual BlisConGenerator::~~BlisConGenerator () [inline],[virtual]`

Destructor.

Definition at line 152 of file `BlisConGenerator.h`.

3.8.3 Member Function Documentation

3.8.3.1 `BlisConGenerator& BlisConGenerator::operator= (const BlisConGenerator & rhs)`

Assignment operator.

3.8.3.2 `virtual bool BlisConGenerator::generateConstraints (BcpsConstraintPool & conPool)` `[virtual]`

Generate cons for the client model.

Evaluate the state of the client model and decide whether to generate cons. The generated cons are inserted into and returned in the collection of cons `cs`.

The routine returns true if reoptimisation is needed (because the state of the solver interface has been modified).

3.8.3.3 `BlisModel* BlisConGenerator::getModel ()` `[inline]`

Set the client model.

In addition to setting the client model, `refreshModel` also calls the `refreshSolver` method of the `CglCutGenerator` object. Get a pointer to the model

Definition at line 182 of file `BlisConGenerator.h`.

3.8.3.4 `void BlisConGenerator::setName (const char * str)` `[inline]`

return name of generator.

Definition at line 191 of file `BlisConGenerator.h`.

3.8.3.5 `std::string BlisConGenerator::name () const` `[inline]`

return name of generator.

Definition at line 194 of file `BlisConGenerator.h`.

3.8.3.6 `void BlisConGenerator::setStrategy (BlisCutStrategy value)` `[inline]`

Set the con generation strategy.

Definition at line 197 of file `BlisConGenerator.h`.

3.8.3.7 `BlisCutStrategy BlisConGenerator::strategy () const` `[inline]`

Get the con generation interval.

Definition at line 200 of file `BlisConGenerator.h`.

3.8.3.8 `void BlisConGenerator::setCutGenerationFreq (int freq)` `[inline]`

Set the con generation strategy.

Definition at line 203 of file `BlisConGenerator.h`.

3.8.3.9 `int BlisConGenerator::cutGenerationFreq () const` `[inline]`

Get the con generation interval.

Definition at line 206 of file `BlisConGenerator.h`.

3.8.3.10 `bool BlisConGenerator::normal () const` `[inline]`

Get whether the con generator should be called in the normal place.

Definition at line 209 of file `BlisConGenerator.h`.

3.8.3.11 `void BlisConGenerator::setNormal (bool value) [inline]`

Set whether the con generator should be called in the normal place.

Definition at line 212 of file `BlisConGenerator.h`.

3.8.3.12 `bool BlisConGenerator::atSolution () const [inline]`

Get whether the con generator should be called when a solution is found.

Definition at line 216 of file `BlisConGenerator.h`.

3.8.3.13 `void BlisConGenerator::setAtSolution (bool value) [inline]`

Set whether the con generator should be called when a solution is found.

Definition at line 220 of file `BlisConGenerator.h`.

3.8.3.14 `bool BlisConGenerator::whenInfeasible () const [inline]`

Get whether the con generator should be called when the subproblem is found to be infeasible.

Definition at line 224 of file `BlisConGenerator.h`.

3.8.3.15 `void BlisConGenerator::setWhenInfeasible (bool value) [inline]`

Set whether the con generator should be called when the subproblem is found to be infeasible.

Definition at line 228 of file `BlisConGenerator.h`.

3.8.3.16 `CglCutGenerator* BlisConGenerator::generator () const [inline]`

Get the `CglCutGenerator` bound to this `BlisConGenerator`.

Definition at line 231 of file `BlisConGenerator.h`.

3.8.3.17 `int BlisConGenerator::numConsGenerated () [inline]`

Get number of generated cons.

Definition at line 234 of file `BlisConGenerator.h`.

3.8.3.18 `void BlisConGenerator::addNumConsGenerated (int n) [inline]`

Increase the number of generated cons.

Definition at line 237 of file `BlisConGenerator.h`.

3.8.3.19 `int BlisConGenerator::numConsUsed () [inline]`

Get number of used cons.

Definition at line 240 of file `BlisConGenerator.h`.

3.8.3.20 `void BlisConGenerator::addNumConsUsed (int n) [inline]`

Increase the number of generated cons.

Definition at line 243 of file `BlisConGenerator.h`.

3.8.3.21 `double BlisConGenerator::time () const [inline]`

Cpu time used.

Definition at line 246 of file BlisConGenerator.h.

3.8.3.22 `void BlisConGenerator::addTime (double t) [inline]`

Increase Cpu time used.

Definition at line 249 of file BlisConGenerator.h.

3.8.3.23 `int BlisConGenerator::calls () const [inline]`

Number called.

Definition at line 252 of file BlisConGenerator.h.

3.8.3.24 `void BlisConGenerator::addCalls (int n = 1) [inline]`

Increase the number of called.

Definition at line 255 of file BlisConGenerator.h.

3.8.3.25 `int BlisConGenerator::noConsCalls () const [inline]`

Number called and no cons found.

Definition at line 258 of file BlisConGenerator.h.

3.8.3.26 `void BlisConGenerator::addNoConsCalls (int n = 1) [inline]`

Increase the number of no cons called.

Definition at line 261 of file BlisConGenerator.h.

3.8.4 Member Data Documentation

3.8.4.1 `BlisModel* BlisConGenerator::model_ [protected]`

The client model.

Definition at line 62 of file BlisConGenerator.h.

3.8.4.2 `CglCutGenerator* BlisConGenerator::generator_ [protected]`

The CglCutGenerator object.

Definition at line 65 of file BlisConGenerator.h.

3.8.4.3 `BlisCutStrategy BlisConGenerator::strategy_ [protected]`

When to call CglCutGenerator::generateCuts routine.

BlisCutStrategyNone: disable BlisCutStrategyRoot: just root BlisCutStrategyAuto: automatically decided by BLIS BlisCutStrategyPeriodic: Generate every 't' nodes

Definition at line 77 of file BlisConGenerator.h.

3.8.4.4 int BlisConGenerator::cutGenerationFrequency_ [protected]

The frequency of calls to the cut generator.

Definition at line 80 of file BlisConGenerator.h.

3.8.4.5 std::string BlisConGenerator::name_ [protected]

Name of generator.

Definition at line 83 of file BlisConGenerator.h.

3.8.4.6 bool BlisConGenerator::normal_ [protected]

Whether to call the generator in the normal place.

Definition at line 86 of file BlisConGenerator.h.

3.8.4.7 bool BlisConGenerator::atSolution_ [protected]

Whether to call the generator when a new solution is found.

Definition at line 89 of file BlisConGenerator.h.

3.8.4.8 bool BlisConGenerator::whenInfeasible_ [protected]

Whether to call generator when a subproblem is found to be infeasible.

Definition at line 93 of file BlisConGenerator.h.

3.8.4.9 int BlisConGenerator::numConsGenerated_ [protected]

Number of cons generated.

Definition at line 100 of file BlisConGenerator.h.

3.8.4.10 int BlisConGenerator::numConsUsed_ [protected]

Number of cons used.

Definition at line 103 of file BlisConGenerator.h.

3.8.4.11 double BlisConGenerator::time_ [protected]

Used CPU/User time.

Definition at line 106 of file BlisConGenerator.h.

3.8.4.12 int BlisConGenerator::calls_ [protected]

The times of calling this generator.

Definition at line 109 of file BlisConGenerator.h.

3.8.4.13 int BlisConGenerator::noConsCalls_ [protected]

The times of calling this generator and no cons found.

Definition at line 112 of file BlisConGenerator.h.

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

- BlisConGenerator.h

3.9 BlisConstraint Class Reference

Public Member Functions

- [BlisConstraint](#) ()
Default constructor.
- [BlisConstraint](#) (int s, const int *ind, const double *val)
Useful constructor.
- [BlisConstraint](#) (double lbh, double ubh, double lbs, double ubs)
Useful constructor.
- [BlisConstraint](#) (double lbh, double ubh, double lbs, double ubs, int size, const int *ind, const double *val)
Useful constructor.
- virtual [~BlisConstraint](#) ()
Destructor.
- [BlisConstraint](#) (const [BlisConstraint](#) &rhs)
Copy constructor.
- OsiRowCut * [createOsiRowCut](#) ()
Create a OsiRowCut based on this constraint.
- virtual void [hashing](#) (BcpsModel *model=NULL)
Compute a hash key.
- double [violation](#) (const double *lpSolution)
Check if violates a given lp solution.
- virtual AlpsReturnStatus [encode](#) (AlpsEncoded *encoded)
Pack into a encode object.
- virtual AlpsKnowledge * [decode](#) (AlpsEncoded &encoded) const
Decode a constraint from an encoded object.
- int [getSize](#) () const
Return data.
- void [setData](#) (int s, const int *ind, const double *val)
Set data.

Protected Member Functions

- AlpsReturnStatus [encodeBlis](#) (AlpsEncoded *encoded)
Pack Blis part into an encoded object.
- AlpsReturnStatus [decodeBlis](#) (AlpsEncoded &encoded)
Unpack Blis part from a encode object.

Protected Attributes

- int [size_](#)
Number of nonzero coefficients.
- int * [indices_](#)
Variable indices.
- double * [values_](#)
Value of nonzero coefficients.

3.9.1 Detailed Description

Definition at line 33 of file BlisConstraint.h.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 `BlisConstraint::BlisConstraint ()`

Default constructor.

3.9.2.2 `BlisConstraint::BlisConstraint (int s, const int * ind, const double * val)`

Useful constructor.

3.9.2.3 `BlisConstraint::BlisConstraint (double lbh, double ubh, double lbs, double ubs)`

Useful constructor.

3.9.2.4 `BlisConstraint::BlisConstraint (double lbh, double ubh, double lbs, double ubs, int size, const int * ind, const double * val)`

Useful constructor.

3.9.2.5 `virtual BlisConstraint::~~BlisConstraint ()` [virtual]

Destructor.

3.9.2.6 `BlisConstraint::BlisConstraint (const BlisConstraint & rhs)`

Copy constructor.

3.9.3 Member Function Documentation

3.9.3.1 `AlpsReturnStatus BlisConstraint::encodeBlis (AlpsEncoded * encoded)` [protected]

Pack Blis part into an encoded object.

3.9.3.2 `AlpsReturnStatus BlisConstraint::decodeBlis (AlpsEncoded & encoded)` [protected]

Unpack Blis part from a encode object.

3.9.3.3 `OsiRowCut* BlisConstraint::createOsiRowCut ()`

Create a OsiRowCut based on this constraint.

3.9.3.4 `virtual void BlisConstraint::hashing (BcpsModel * model = NULL)` [virtual]

Compute a hash key.

3.9.3.5 `double BlisConstraint::violation (const double * lpSolution)`

Check if violates a given lp solution.

3.9.3.6 `virtual AlpsReturnStatus BlisConstraint::encode (AlpsEncoded * encoded) [virtual]`

Pack into a encode object.

3.9.3.7 `virtual AlpsKnowledge* BlisConstraint::decode (AlpsEncoded & encoded) const [virtual]`

Decode a constraint from an encoded object.

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

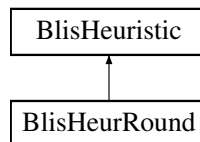
- BlisConstraint.h

3.10 BlisHeuristic Class Reference

Heuristic base class.

```
#include <BlisHeuristic.h>
```

Inheritance diagram for BlisHeuristic:



Public Member Functions

- [BlisHeuristic](#) ()
Default Constructor.
- [BlisHeuristic](#) (BlisModel *model, const char *name, BlisHeurStrategy strategy, int heurCallFrequency)
Useful constructor.
- virtual [~BlisHeuristic](#) ()
Distructor.
- [BlisHeuristic](#) (const [BlisHeuristic](#) &rhs)
Copy constructor.
- virtual void [setModel](#) (BlisModel *model)
update model (This is needed if cliques update matrix etc).
- virtual void [setStrategy](#) (BlisHeurStrategy strategy)
Get/set strategy.
- virtual void [setHeurCallFrequency](#) (int freq)
Get/set call frequency.
- virtual [BlisHeuristic](#) * [clone](#) () const
Clone a heuristic.
- virtual bool [searchSolution](#) (double &objectiveValue, double *newSolution)=0
returns 0 if no solution, 1 if valid solution with better objective value than one passed in Sets solution values if good, sets objective value This is called after cuts have been added - so can not add cuts
- virtual bool [searchSolution](#) (double &objectiveValue, double *newSolution, OsiCuts &cs)

returns 0 if no solution, 1 if valid solution, -1 if just returning an estimate of best possible solution with better objective value than one passed in Sets solution values if good, sets objective value (only if nonzero code) This is called at same time as cut generators - so can add cuts Default is do nothing

- `const char * name () const`
return name of generator.
- `void addNumSolutions (int num=1)`
Record number of solutions found.
- `int numSolutions () const`
Number of solutions found.
- `void addTime (double t=0.0)`
Record Cpu time used.
- `double time () const`
Cpu time used.
- `void addCalls (int c=1)`
Record number of times called.
- `int calls () const`
Number of times called.
- `int noSolCalls () const`
Number called and no cons found.
- `void addNoSolCalls (int n=1)`
Increase the number of no cons called.

Protected Attributes

- `BlisModel * model_`
Pointer to the model.
- `char * name_`
Heuristics name.
- `BlisHeurStrategy strategy_`
When to call findSolution() routine.
- `int heurCallFrequency_`
The frequency with which to call the heuristic.
- `int numSolutions_`
Number of solutions found.
- `double time_`
Used CPU/User time.
- `int calls_`
The times of calling this heuristic.
- `int noSolsCalls_`
The times of calling this heuristic and no solutions found.

3.10.1 Detailed Description

Heuristic base class.

Definition at line 48 of file BlisHeuristic.h.

3.10.2 Constructor & Destructor Documentation

3.10.2.1 `BlisHeuristic::BlisHeuristic ()` `[inline]`

Default Constructor.

Definition at line 90 of file `BlisHeuristic.h`.

3.10.2.2 `BlisHeuristic::BlisHeuristic (BlisModel * model, const char * name, BlisHeurStrategy strategy, int heurCallFrequency)` `[inline]`

Useful constructor.

Definition at line 102 of file `BlisHeuristic.h`.

3.10.2.3 `virtual BlisHeuristic::~~BlisHeuristic ()` `[inline]`, `[virtual]`

Distructor.

Definition at line 120 of file `BlisHeuristic.h`.

3.10.2.4 `BlisHeuristic::BlisHeuristic (const BlisHeuristic & rhs)` `[inline]`

Copy constructor.

Definition at line 123 of file `BlisHeuristic.h`.

3.10.3 Member Function Documentation

3.10.3.1 `virtual void BlisHeuristic::setModel (BlisModel * model)` `[inline]`, `[virtual]`

update model (This is needed if cliques update matrix etc).

Reimplemented in [BlisHeurRound](#).

Definition at line 135 of file `BlisHeuristic.h`.

3.10.3.2 `virtual void BlisHeuristic::setStrategy (BlisHeurStrategy strategy)` `[inline]`, `[virtual]`

Get/set strategy.

Definition at line 139 of file `BlisHeuristic.h`.

3.10.3.3 `virtual void BlisHeuristic::setHeurCallFrequency (int freq)` `[inline]`, `[virtual]`

Get/set call frequency.

Definition at line 145 of file `BlisHeuristic.h`.

3.10.3.4 `virtual BlisHeuristic* BlisHeuristic::clone () const` `[inline]`, `[virtual]`

Clone a heuristic.

Reimplemented in [BlisHeurRound](#).

Definition at line 150 of file `BlisHeuristic.h`.

3.10.3.5 `const char* BlisHeuristic::name () const` `[inline]`

return name of generator.

Definition at line 177 of file BlisHeuristic.h.

3.10.3.6 `void BlisHeuristic::addNumSolutions (int num = 1) [inline]`

Record number of solutions found.

Definition at line 180 of file BlisHeuristic.h.

3.10.3.7 `int BlisHeuristic::numSolutions () const [inline]`

Number of solutions found.

Definition at line 183 of file BlisHeuristic.h.

3.10.3.8 `void BlisHeuristic::addTime (double t = 0.0) [inline]`

Record Cpu time used.

Definition at line 186 of file BlisHeuristic.h.

3.10.3.9 `double BlisHeuristic::time () const [inline]`

Cpu time used.

Definition at line 189 of file BlisHeuristic.h.

3.10.3.10 `void BlisHeuristic::addCalls (int c = 1) [inline]`

Record number of times called.

Definition at line 192 of file BlisHeuristic.h.

3.10.3.11 `int BlisHeuristic::calls () const [inline]`

Number of times called.

Definition at line 195 of file BlisHeuristic.h.

3.10.3.12 `int BlisHeuristic::noSolCalls () const [inline]`

Number called and no cons found.

Definition at line 198 of file BlisHeuristic.h.

3.10.3.13 `void BlisHeuristic::addNoSolCalls (int n = 1) [inline]`

Increase the number of no cons called.

Definition at line 201 of file BlisHeuristic.h.

3.10.4 Member Data Documentation

3.10.4.1 `BlisHeurStrategy BlisHeuristic::strategy_ [protected]`

When to call findSolution() routine.

BlisHeurStrategyNone: disable BlisHeurStrategyRoot: just root BlisHeurStrategyAuto: automatically decided by BLIS
BlisHeurStrategyPeriodic: every 't' nodes BlisHeurStrategyBeforeRoot: before solving first LP

Definition at line 70 of file BlisHeuristic.h.

3.10.4.2 int BlisHeuristic::numSolutions_ [protected]

Number of solutions found.

Definition at line 76 of file BlisHeuristic.h.

3.10.4.3 double BlisHeuristic::time_ [protected]

Used CPU/User time.

Definition at line 79 of file BlisHeuristic.h.

3.10.4.4 int BlisHeuristic::calls_ [protected]

The times of calling this heuristic.

Definition at line 82 of file BlisHeuristic.h.

3.10.4.5 int BlisHeuristic::noSolsCalls_ [protected]

The times of calling this heuristic and no solutions found.

Definition at line 85 of file BlisHeuristic.h.

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

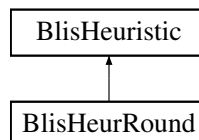
- BlisHeuristic.h

3.11 BlisHeurRound Class Reference

Rounding Heuristic.

```
#include <BlisHeurRound.h>
```

Inheritance diagram for BlisHeurRound:



Public Member Functions

- [BlisHeurRound](#) ()
Default Constructor.
- [BlisHeurRound](#) ([BlisModel](#) *model, const char *name, BlisHeurStrategy strategy, int freq)
Constructor with model - assumed before cuts.
- [~BlisHeurRound](#) ()
Destructor.
- [BlisHeurRound](#) (const [BlisHeurRound](#) &)
Copy constructor.
- virtual [BlisHeuristic](#) * [clone](#) () const
Clone a rounding heuristic.
- virtual void [setModel](#) ([BlisModel](#) *model)

update model (This is needed if cliques update matrix etc).

- virtual bool [searchSolution](#) (double &objectiveValue, double *newSolution)
returns 0 if no solution, 1 if valid solution with better objective value than one passed in Sets solution values if good, sets objective value (only if good) This is called after cuts have been added - so can not add cuts
- void [setSeed](#) (int value)
Set seed.

Protected Attributes

- CoinPackedMatrix [matrix_](#)
Column majored matrix.
- CoinPackedMatrix [matrixByRow_](#)
Row majored matrix.
- int [seed_](#)
Seed for random stuff.

3.11.1 Detailed Description

Rounding Heuristic.

Definition at line 44 of file BlisHeurRound.h.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 `BlisHeurRound::BlisHeurRound () [inline]`

Default Constructor.

Definition at line 61 of file BlisHeurRound.h.

3.11.2.2 `BlisHeurRound::BlisHeurRound (BlisModel * model, const char * name, BlisHeurStrategy strategy, int freq) [inline]`

Constructor with model - assumed before cuts.

Definition at line 64 of file BlisHeurRound.h.

3.11.2.3 `BlisHeurRound::~~BlisHeurRound () [inline]`

Destructor.

Definition at line 73 of file BlisHeurRound.h.

3.11.2.4 `BlisHeurRound::BlisHeurRound (const BlisHeurRound &)`

Copy constructor.

3.11.3 Member Function Documentation

3.11.3.1 `virtual void BlisHeurRound::setModel (BlisModel * model) [virtual]`

update model (This is needed if cliques update matrix etc).

Reimplemented from [BlisHeuristic](#).

3.11.4 Member Data Documentation

3.11.4.1 CoinPackedMatrix BlisHeurRound::matrix_ [protected]

Column majored matrix.

Definition at line 51 of file BlisHeurRound.h.

3.11.4.2 CoinPackedMatrix BlisHeurRound::matrixByRow_ [protected]

Row majored matrix.

Definition at line 54 of file BlisHeurRound.h.

3.11.4.3 int BlisHeurRound::seed_ [protected]

Seed for random stuff.

Definition at line 57 of file BlisHeurRound.h.

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

- BlisHeurRound.h

3.12 BlisMessage Class Reference

Public Member Functions

Constructors etc

- [BlisMessage](#) (Language language=us_en)
Constructor.

3.12.1 Detailed Description

Definition at line 58 of file BlisMessage.h.

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

- BlisMessage.h

3.13 BlisModel Class Reference

Public Member Functions

- [BlisModel](#) ()
Default construtor.
- virtual [~BlisModel](#) ()
Destructor.
- void [gutsOfDestructor](#) ()
Actual destructor.
- void [setColMatrix](#) (CoinPackedMatrix *mat)
Pass a matrix in.
- void [setNumCons](#) (int num)

- Pass column upper bounds.*
 - void [setNumVars](#) (int num)
- Pass column upper bounds.*
 - void [setNumElems](#) (int num)
- Pass column upper bounds.*
 - void [setConLb](#) (double *cl)
- Pass column upper bounds.*
 - void [setConUb](#) (double *cu)
- Pass column lower bounds.*
 - void [setVarLb](#) (double *lb)
- Pass variable upper bounds.*
 - void [setVarUb](#) (double *ub)
- Pass variable lower bounds.*
 - void [setColType](#) (char *colType)
- Pass variable types.*
 - void [setObjCoef](#) (double *obj)
- Pass objective coefficients.*
 - virtual void [readInstance](#) (const char *dataFile)
- For parallel code, only the master calls this function.*
 - virtual void [importModel](#) (std::vector< [BlisVariable](#) * > vars, std::vector< [BlisConstraint](#) * > cons)
- For parallel code, only the master calls this function.*
 - virtual void [readParameters](#) (const int argnum, const char *const *arglist)
- Read in Alps, Blis parameters.*
 - virtual void [writeParameters](#) (std::ostream &outstream) const
- Write out parameters.*
 - virtual AlpsTreeNode * [createRoot](#) ()
- For parallel code, only the master calls this function.*
 - virtual bool [setupSelf](#) ()
- All processes call this function.*
 - virtual void [preprocess](#) ()
- Preprocessing the model.*
 - virtual void [postprocess](#) ()
- Postprocessing the searching results.*
 - virtual void [setSolver](#) (OsiSolverInterface *si)
- Set lp solver.*
 - virtual OsiSolverInterface * [getSolver](#) ()
- Get lp solver.*
 - virtual OsiSolverInterface * [solver](#) ()
- Get lp solver.*
 - bool [resolve](#) ()
- Resolving a lp.*
 - void [setActiveNode](#) (AlpsTreeNode *node)
- Set active node.*
 - void [setSolEstimate](#) (double est)
- Set the solution estimate of the active node.*
 - int [getNumStrong](#) ()
- Get number of strong branchings.*

- void `addNumStrong` (int num=1)
Add num to number of strong branchings.
- int `getNumBranchResolve` ()
Get the maximum number of resolve during branching.
- void `setNumBranchResolve` (int num)
Set the maximum number of resolve during branching.
- double * `getObjCoef` () const
Get objective coefficients.
- const double * `getColLower` ()
Get column lower bound.
- const double * `getColUpper` ()
Get column upper bound.
- int `getNumCols` ()
Get number of columns.
- int `getNumRows` ()
Get number of rows.
- double * `varLB` ()
Get variable bounds array.
- double * `conLB` ()
Get original constraint bounds array.
- double * `startVarLB` ()
The starting variable bounds array of a subproblem (internal use).
- double * `startConLB` ()
The starting constraint bounds array of a subproblem (internal use).
- int * `tempVarLBPos` ()
Temporary storage.
- double `getLpObjValue` () const
Get current objective function value.
- const double * `getLpSolution` () const
Get active lp solution.
- int `getNumSolutions` () const
Get number of solutions.
- int `getNumHeurSolutions` () const
Get number of heuristic solutions.
- double * `incumbent` ()
Return best ip solution found so far.
- int `storeSolution` (BlisSolutionType how, `BlisSolution` *sol)
Record a new incumbent solution and update objectiveValue.
- double `getCutoff` () const
Get cut off value.
- void `setCutoff` (double co)
Set cut off value.
- `BlisSolution` * `feasibleSolutionHeur` (const double *solution)
Test if a solution found by heuristic is feasible.
- virtual `BlisSolution` * `feasibleSolution` (int &numIntegerInfs, int &numObjectInfs)
Test the current LP solution for feasibility.
- virtual `BlisSolution` * `userFeasibleSolution` (const double *solution, bool &feasible)

- User's criteria for a feasible solution.*

 - void [createIntegerObjects](#) (bool startAgain)
- Identify integer variable.*

 - int * [getIntObjIndices](#) () const
- Get integers' object indices.*

 - int [getNumIntObjects](#) () const
- Get number of integers.*

 - int * [getIntColIndices](#) () const
- Get integers' column indices.*

 - bool [checkInteger](#) (double value) const
- Check if a value is integer.*

 - void [addHeuristic](#) (BlisHeuristic *heur)
- Add a heuristic.*

 - BlisHeuristic * [heuristics](#) (int i) const
- Get a specific heuristic.*

 - int [numHeuristics](#) () const
- Get the number of heuristics.*

 - void [addCutGenerator](#) (BlisConGenerator *generator)
- Add a Blis cut generator.*

 - void [addCutGenerator](#) (CglCutGenerator *generator, const char *name=NULL, BlisCutStrategy strategy=BlisCutStrategyAuto, int cutGenerationFrequency=1, bool normal=true, bool atSolution=false, bool whenInfeasible=false)
- Add a Cgl cut generator.*

 - BlisConGenerator * [cutGenerators](#) (int i) const
- Get a specific cut generator.*

 - int [numCutGenerators](#) () const
- Get the number of cut generators.*

 - int [getMaxNumCons](#) () const
- Get the max number of cuts can be generated.*

 - void [setMaxNumCons](#) (int m)
- Set the max number of cuts can be generated.*

 - BcpsConstraintPool * [constraintPool](#) ()
- Access constraint pool.*

 - BcpsConstraintPool * [constraintPoolReceive](#) ()
- Access receive constraint pool.*

 - BcpsConstraintPool * [constraintPoolSend](#) ()
- Access send constraint pool.*

 - BlisCutStrategy [getCutStrategy](#) () const
- Query constraint generation strategy.*

 - void [setCutStrategy](#) (BlisCutStrategy u)
- Set constraint generation strategy.*

 - int [getCutGenerationFrequency](#) () const
- Query constraint generation frequency.*

 - void [setCutStrategy](#) (int f)
- Set constraint generation frequency.*

 - int [getDenseConCutoff](#) () const
- Get the threshold to be considered as a dense constraint.*

- void `setDenseConCutoff` (int cutoff)
Set the threshold to be considered as a dense constraint.
- double * `getConRandoms` () const
Get randoms for check parallel constraints.
- void `passInPriorities` (const int *priorities, bool ifNotSimpleIntegers, int defaultValue=1000)
Pass in branching priorities.
- const int * `priority` () const
Priorities.
- int `priority` (int sequence) const
Returns priority level for an object (or 1000 if no priorities exist)
- virtual void `modelLog` ()
Log of specific models.
- int `getNumNodes` () const
Get how many Nodes it took to solve the problem.
- int `getNumIterations` () const
Get how many iterations it took to solve the problem.
- int `getAveIterations` () const
Get the average iterations it took to solve a lp.
- void `addNumNodes` (int newNodes=1)
Increment node count.
- void `addNumIterations` (int newIter)
Increment Iteration count.
- CoinMessageHandler * `blisMessageHandler` () const
Get the message handler.
- CoinMessages `blisMessages` ()
Return messages.
- virtual void `nodeLog` (AlpsTreeNode *node, bool force)
Node log.
- virtual bool `fathomAllNodes` ()
Return true, if all nodes can be fathomed.
- virtual void `registerKnowledge` ()
Register knowledge.
- virtual AlpsEncoded * `encode` () const
The method that encodes the model into an encoded object.
- virtual void `decodeToSelf` (AlpsEncoded &)
The method that decodes the model from an encoded object.
- virtual AlpsEncoded * `packSharedKnowledge` ()
Pack knowledge to be shared with others into an encoded object.
- virtual void `unpackSharedKnowledge` (AlpsEncoded &)
Unpack and store shared knowledge from an encoded object.

Branching Strategies

See the `BcpsBranchStrategy` class for additional information.

- BcpsBranchStrategy * `branchStrategy` () const
Get the current branching strategy.
- void `setBranchingMethod` (BcpsBranchStrategy *method)
Set the branching strategy.

- void [setBranchingMethod](#) (BcpsBranchStrategy &method)
Set the branching strategy.
- BcpsBranchStrategy * [rampUpBranchStrategy](#) () const

Object manipulation routines

- int [numObjects](#) () const
Get the number of objects.
- void [setNumObjects](#) (int num)
Set the number of objects.
- BcpsObject ** [objects](#) ()
Get the array of objects.
- BcpsObject * [objects](#) (int which)
Get the specified object.
- void [setSharedObjectMark](#) (int i)
Mark object to be shared.
- void [clearSharedObjectMark](#) ()
Clear all the share mark.
- void [deleteObjects](#) ()
Delete all object information.
- void [addObjects](#) (int [numObjects](#), BcpsObject **[objects](#))
Add in object information.
- int [getNumOldConstraints](#) () const
Get number of old constraints.
- void [setNumOldConstraints](#) (int num)
Set number of old constraints.
- int [getOldConstraintsSize](#) () const
Get max number of old constraints.
- void [setOldConstraintsSize](#) (int num)
Set max number of old constraints.
- [BlisConstraint](#) ** [oldConstraints](#) ()
Access old constraints.
- void [setOldConstraints](#) ([BlisConstraint](#) **old)
set old constraints.
- void [delOldConstraints](#) ()
Set max number of old constraints.
- [BlisParams](#) * [BlisPar](#) ()
Access parameters.

Public Attributes

- bool [isRoot_](#)
If root node.
- int [boundingPass_](#)
The number of passes during bounding procedure.
- double [integerTol_](#)
Integer tolerance.
- double [optimalRelGap_](#)

- *Input relative optimal gap.*
- double [optimalAbsGap_](#)
- *Input absolute optimal gap.*
- double [currRelGap_](#)
- *Current relative optimal gap.*
- double [currAbsGap_](#)
- *Current absolute optimal gap.*
- BlisHeurStrategy [heurStrategy_](#)
- *If use heuristics.*
- int [heurCallFrequency_](#)
- *Frequency of using heuristics.*
- OsiCuts [newCutPool_](#)
- *Store new cuts in each pass.*
- std::vector< AlpsTreeNode * > [leafToRootPath](#)
- *Record the path from leaf to root.*

Protected Member Functions

- void [init](#) ()
- *Intialize member data.*
- void [createObjects](#) ()
- *Create variables and constraints.*
- AlpsReturnStatus [encodeBlis](#) (AlpsEncoded *encoded) const
- *Pack Blis portion of the model into an encoded object.*
- AlpsReturnStatus [decodeBlis](#) (AlpsEncoded &encoded)
- *Unpack Blis portion of the model from an encoded object.*
- void [packSharedPseudocost](#) (AlpsEncoded *encoded, int numToShare)
- *Retrieve and pack shared pseudocost.*
- void [unpackSharedPseudocost](#) (AlpsEncoded &encoded)
- *Unpack and store shared pseduocost.*
- void [packSharedConstraints](#) (AlpsEncoded *encoded)
- *Retrieve and pack shared constraints.*
- void [unpackSharedConstraints](#) (AlpsEncoded &encoded)
- *Unpack and store shared constraints.*
- void [packSharedVariables](#) (AlpsEncoded *encoded)
- *Retrieve and pack shared variables.*
- void [unpackSharedVariables](#) (AlpsEncoded &encoded)
- *Unpack and store shared variables.*

Protected Attributes

- OsiSolverInterface * [origLpSolver_](#)
- *Input by user.*
- OsiSolverInterface * [presolvedLpSolver_](#)
- *Presolved.*
- OsiSolverInterface * [lpSolver_](#)
- *Actually used.*

- CoinPackedMatrix * [colMatrix_](#)
Column majored matrix.
- double [incObjValue_](#)
Incumbent objective value.
- double * [incumbent_](#)
Incumbent.
- double [cutoff_](#)
Cutoff in Ip solver.
- double [cutoffInc_](#)
Cutoff increment.
- BcpsBranchStrategy * [branchStrategy_](#)
Variable selection function.
- int [numObjects_](#)
Number of objects.
- BcpsObject ** [objects_](#)
The set of objects.
- char * [sharedObjectMark_](#)
The objects that can be shared.
- int * [priority_](#)
Priorities of integer object.
- AlpsTreeNode * [activeNode_](#)
Active node.
- int [numStrong_](#)
Number of strong branching.
- int [numBranchResolve_](#)
Maximum number of resolve during branching.
- int [numHeuristics_](#)
Number of heuristics.
- BlisHeuristic ** [heuristics_](#)
The list of heuristics.
- BlisCutStrategy [cutStrategy_](#)
If use cut generators.
- int [cutGenerationFrequency_](#)
Frequency of cut generation.
- int [numCutGenerators_](#)
Number of cut generators used.
- int [maxNumCons_](#)
Number of cuts can be generators.
- BlisConGenerator ** [generators_](#)
The list of cut generators used.
- BcpsConstraintPool * [constraintPool_](#)
Store all the cuts.
- BlisConstraint ** [oldConstraints_](#)
Temporary store old cuts at a node when installing a node.
- int [oldConstraintsSize_](#)
The memory size allocated for oldConstraints_.
- int [numOldConstraints_](#)

- Number of old constraints.*

 - double * [conRandoms_](#)
- Random keys.*

 - int [denseConCutoff_](#)
- Dense constraint cutoff.*

 - [BlisParams](#) * [BlisPar_](#)
- Blis parameters.*

 - CoinMessageHandler * [blisMessageHandler_](#)
- Message handler.*

 - CoinMessages [blisMessages_](#)
- Blis messages.*

 - int [numNodes_](#)
- Number of processed nodes.*

 - int [numIterations_](#)
- Number of lp(Simplex) iterations.*

 - int [avelterations_](#)
- Average number of lp iterations to solve a subproblem.*

 - BcpsConstraintPool * [constraintPoolSend_](#)
- Constraints that can be sent/broadcasted to other processes.*

 - BcpsConstraintPool * [constraintPoolReceive_](#)
- Constraints that are received from other processses.*
- double * [varLB_](#)

Variable and constraint bounds.
- int [numCols_](#)

Number of columns/rows/elements.
- double [objSense_](#)

Objective function.
- int [numIntObjects_](#)

Column types.
- std::vector< BcpsVariable * > [inputVar_](#)

User's input objects.
- double * [startVarLB_](#)

Starting var/con bounds for processing each node.
- int * [tempVarLBPos_](#)

Tempory storage for var/con indices.

3.13.1 Detailed Description

Definition at line 69 of file BlisModel.h.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 `BlisModel::BlisModel ()` `[inline]`

Default construtor.

Definition at line 339 of file `BlisModel.h`.

3.13.2.2 `virtual BlisModel::~~BlisModel ()` `[virtual]`

Destructor.

3.13.3 Member Function Documentation

3.13.3.1 `void BlisModel::createObjects ()` `[protected]`

Create variables and constraints.

3.13.3.2 `void BlisModel::gutsOfDestructor ()`

Actual destructor.

3.13.3.3 `virtual void BlisModel::readInstance (const char * dataFile)` `[virtual]`

For parallel code, only the master calls this function.

1) Read in the instance data 2) Set `colMatrix_`, `varLB_`, `varUB_`, `conLB_`, `conUB` `numCols_`, `numRows_` 3) Set `objCoef_` and `objSense_` 4) Set `colType_` ('C', 'I', or 'B') 5) Create variables and constraints 6) Set `numCoreVariables_` and `numCoreConstraints_`

3.13.3.4 `virtual void BlisModel::importModel (std::vector< BlisVariable * > vars, std::vector< BlisConstraint * > cons)` `[virtual]`

For parallel code, only the master calls this function.

Import model from `vars` and `cons`. 1) Set `colMatrix_`, `varLB_`, `varUB_`, `conLB_`, `conUB` `numCols_`, `numRows_` 2) Set `objCoef_` (Assume minimization) 3) Set `colType_` ('C', 'I', or 'B') 4) Set `variables_` and `constraints_` 5) Set `numCoreVariables_` and `numCoreConstraints_` NOTE: Blis takes over the memory ownership of `vars` and `cons`, which means users must NOT free `vars` or `cons`.

3.13.3.5 `virtual void BlisModel::readParameters (const int argnum, const char *const * arglist)` `[virtual]`

Read in Alps, Blis parameters.

3.13.3.6 `virtual void BlisModel::writeParameters (std::ostream & outstream) const` `[virtual]`

Write out parameters.

3.13.3.7 `virtual AlpsTreeNode* BlisModel::createRoot ()` `[virtual]`

For parallel code, only the master calls this function.

Create the root node based on model.

3.13.3.8 `virtual bool BlisModel::setUpSelf ()` `[virtual]`

All processes call this function.

Do necessary work to make model usable. Return success or not. 1) Set numIntObjects_, intColIndices_, intObjectIndices_ 2) Load problem to LP solver. 3) Create integer objects (must after load to lp since using lp info) 4) Set branch strategy 5) Add heuristics 6) Add Cgl cut generators

3.13.3.9 `virtual void BlisModel::preprocess () [virtual]`

Preprocessing the model.

3.13.3.10 `virtual void BlisModel::postprocess () [virtual]`

Postprocessing the searching results.

3.13.3.11 `virtual void BlisModel::setSolver (OsiSolverInterface * si) [inline],[virtual]`

Set lp solver.

Definition at line 440 of file BlisModel.h.

3.13.3.12 `virtual OsiSolverInterface* BlisModel::getSolver () [inline],[virtual]`

Get lp solver.

Definition at line 443 of file BlisModel.h.

3.13.3.13 `virtual OsiSolverInterface* BlisModel::solver () [inline],[virtual]`

Get lp solver.

Definition at line 446 of file BlisModel.h.

3.13.3.14 `bool BlisModel::resolve ()`

Resolving a lp.

3.13.3.15 `void BlisModel::setActiveNode (AlpsTreeNode * node) [inline]`

Set active node.

Definition at line 452 of file BlisModel.h.

3.13.3.16 `void BlisModel::setSolEstimate (double est) [inline]`

Set the solution estimate of the active node.

Definition at line 455 of file BlisModel.h.

3.13.3.17 `int BlisModel::getNumStrong () [inline]`

Get number of strong branchings.

Definition at line 458 of file BlisModel.h.

3.13.3.18 `void BlisModel::addNumStrong (int num = 1) [inline]`

Add num to number of strong branchings.

Definition at line 461 of file BlisModel.h.

3.13.3.19 `int BlisModel::getNumBranchResolve () [inline]`

Get the maximum number of resolve during branching.

Definition at line 464 of file BlisModel.h.

3.13.3.20 `void BlisModel::setNumBranchResolve (int num) [inline]`

Set the maximum number of resolve during branching.

Definition at line 467 of file BlisModel.h.

3.13.3.21 `double* BlisModel::getObjCoef () const [inline]`

Get objective coefficients.

Definition at line 474 of file BlisModel.h.

3.13.3.22 `const double* BlisModel::getColLower () [inline]`

Get column lower bound.

Definition at line 477 of file BlisModel.h.

3.13.3.23 `const double* BlisModel::getColUpper () [inline]`

Get column upper bound.

Definition at line 480 of file BlisModel.h.

3.13.3.24 `int BlisModel::getNumCols () [inline]`

Get number of columns.

Definition at line 483 of file BlisModel.h.

3.13.3.25 `int BlisModel::getNumRows () [inline]`

Get number of rows.

Definition at line 486 of file BlisModel.h.

3.13.3.26 `double* BlisModel::varLB () [inline]`

Get variable bounds array.

Definition at line 489 of file BlisModel.h.

3.13.3.27 `double* BlisModel::conLB () [inline]`

Get original constraint bounds array.

Definition at line 493 of file BlisModel.h.

3.13.3.28 `double* BlisModel::startVarLB () [inline]`

The starting variable bounds array of a subproblem (internal use).

Definition at line 497 of file BlisModel.h.

3.13.3.29 `double* BlisModel::startConLB () [inline]`

The starting constraint bounds array of a subproblem (internal use).

Definition at line 501 of file BlisModel.h.

3.13.3.30 `int* BlisModel::tempVarLBPos () [inline]`

Temporary storage.

Definition at line 505 of file BlisModel.h.

3.13.3.31 `double BlisModel::getLpObjValue () const [inline]`

Get current objective function value.

Definition at line 515 of file BlisModel.h.

3.13.3.32 `const double* BlisModel::getLpSolution () const [inline]`

Get active lp solution.

Definition at line 518 of file BlisModel.h.

3.13.3.33 `int BlisModel::getNumSolutions () const [inline]`

Get number of solutions.

Definition at line 525 of file BlisModel.h.

3.13.3.34 `int BlisModel::getNumHeurSolutions () const [inline]`

Get number of heuristic solutions.

Definition at line 528 of file BlisModel.h.

3.13.3.35 `double* BlisModel::incumbent () [inline]`

Return best ip solution found so far.

Definition at line 531 of file BlisModel.h.

3.13.3.36 `int BlisModel::storeSolution (BlisSolutionType how, BlisSolution * sol)`

Record a new incumbent solution and update objectiveValue.

3.13.3.37 `double BlisModel::getCutoff () const [inline]`

Get cut off value.

Definition at line 537 of file BlisModel.h.

3.13.3.38 `void BlisModel::setCutoff (double co) [inline]`

Set cut off value.

Definition at line 540 of file BlisModel.h.

3.13.3.39 `BlisSolution* BlisModel::feasibleSolutionHeur (const double * solution)`

Test if a solution found by heuristic is feasible.

3.13.3.40 `virtual BlisSolution* BlisModel::feasibleSolution (int & numIntegerInfs, int & numObjectInfs) [virtual]`

Test the current LP solution for feasibility.

Scan all objects for indications of infeasibility. This is broken down into simple integer infeasibility (*numIntegerInfs*) and all other reports of infeasibility (*numObjectInfs*).

3.13.3.41 `virtual BlisSolution* BlisModel::userFeasibleSolution (const double * solution, bool & feasible) [inline],
[virtual]`

User's criteria for a feasible solution.

If user think the given solution is feasible then need 1) set userFeasible to true, and 2) return a non-null solution. If user think the solution is infeasible then need 1) set userFeasible to false, and 2) return a null.

Definition at line 571 of file BlisModel.h.

3.13.3.42 `BcpsBranchStrategy* BlisModel::branchStrategy () const [inline]`

Get the current branching strategy.

Definition at line 587 of file BlisModel.h.

3.13.3.43 `void BlisModel::setBranchingMethod (BcpsBranchStrategy * method) [inline]`

Set the branching strategy.

Definition at line 591 of file BlisModel.h.

3.13.3.44 `void BlisModel::setBranchingMethod (BcpsBranchStrategy & method) [inline]`

Set the branching strategy.

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 597 of file BlisModel.h.

3.13.3.45 `int BlisModel::numObjects () const [inline]`

Get the number of objects.

Definition at line 609 of file BlisModel.h.

3.13.3.46 `void BlisModel::setNumObjects (int num) [inline]`

Set the number of objects.

Definition at line 612 of file BlisModel.h.

3.13.3.47 `BcpsObject** BlisModel::objects () [inline]`

Get the array of objects.

Definition at line 615 of file BlisModel.h.

3.13.3.48 `BcpsObject* BlisModel::objects (int which) [inline]`

Get the specified object.

Definition at line 618 of file BlisModel.h.

3.13.3.49 `void BlisModel::setSharedObjectMark (int i) [inline]`

Mark object to be shared.

Definition at line 621 of file BlisModel.h.

3.13.3.50 void **BlisModel::clearSharedObjectMark** () [inline]

Clear all the share mark.

Definition at line 624 of file `BlisModel.h`.

3.13.3.51 void **BlisModel::deleteObjects** ()

Delete all object information.

3.13.3.52 void **BlisModel::addObjects** (int *numObjects*, **BcpsObject** ** *objects*)

Add in object information.

Objects are cloned; the owner can delete the originals.

3.13.3.53 void **BlisModel::createIntgerObjects** (bool *startAgain*)

Identify integer variable.

3.13.3.54 int* **BlisModel::getIntObjIndices** () const [inline]

Get integers' object indices.

Definition at line 642 of file `BlisModel.h`.

3.13.3.55 int **BlisModel::getNumIntObjects** () const [inline]

Get number of integers.

Definition at line 645 of file `BlisModel.h`.

3.13.3.56 int* **BlisModel::getIntColIndices** () const [inline]

Get integers' column indices.

Definition at line 648 of file `BlisModel.h`.

3.13.3.57 bool **BlisModel::checkInteger** (double *value*) const [inline]

Check if a value is integer.

Definition at line 651 of file `BlisModel.h`.

3.13.3.58 void **BlisModel::addHeuristic** (**BlisHeuristic** * *heur*)

Add a heuristic.

3.13.3.59 **BlisHeuristic*** **BlisModel::heuristics** (int *i*) const [inline]

Get a specific heuristic.

Definition at line 672 of file `BlisModel.h`.

3.13.3.60 int **BlisModel::numHeuristics** () const [inline]

Get the number of heuristics.

Definition at line 675 of file `BlisModel.h`.

3.13.3.61 void BlisModel::addCutGenerator (BlisConGenerator * *generator*)

Add a Blis cut generator.

3.13.3.62 void BlisModel::addCutGenerator (CglCutGenerator * *generator*, const char * *name* = NULL, BlisCutStrategy *strategy* = BlisCutStrategyAuto, int *cutGenerationFrequency* = 1, bool *normal* = true, bool *atSolution* = false, bool *whenInfeasible* = false)

Add a Cgl cut generator.

3.13.3.63 BlisConGenerator* BlisModel::cutGenerators (int *i*) const [inline]

Get a specific cut generator.

Definition at line 694 of file BlisModel.h.

3.13.3.64 int BlisModel::numCutGenerators () const [inline]

Get the number of cut generators.

Definition at line 697 of file BlisModel.h.

3.13.3.65 int BlisModel::getMaxNumCons () const [inline]

Get the max number of cuts can be generated.

Definition at line 700 of file BlisModel.h.

3.13.3.66 void BlisModel::setMaxNumCons (int *m*) [inline]

Set the max number of cuts can be generated.

Definition at line 703 of file BlisModel.h.

3.13.3.67 BcpsConstraintPool* BlisModel::constraintPool () [inline]

Access constraint pool.

Definition at line 706 of file BlisModel.h.

3.13.3.68 BcpsConstraintPool* BlisModel::constraintPoolReceive () [inline]

Access receive constraint pool.

Definition at line 709 of file BlisModel.h.

3.13.3.69 BcpsConstraintPool* BlisModel::constraintPoolSend () [inline]

Access send constraint pool.

Definition at line 713 of file BlisModel.h.

3.13.3.70 int BlisModel::getNumOldConstraints () const [inline]

Get number of old constraints.

Definition at line 717 of file BlisModel.h.

3.13.3.71 void BlisModel::setNumOldConstraints (int *num*) [inline]

Set number of old constraints.

Definition at line 720 of file BlisModel.h.

3.13.3.72 `int BlisModel::getOldConstraintsSize () const [inline]`

Get max number of old constraints.

Definition at line 723 of file BlisModel.h.

3.13.3.73 `void BlisModel::setOldConstraintsSize (int num) [inline]`

Set max number of old constraints.

Definition at line 726 of file BlisModel.h.

3.13.3.74 `BlisConstraint** BlisModel::oldConstraints () [inline]`

Access old constraints.

Definition at line 729 of file BlisModel.h.

3.13.3.75 `void BlisModel::setOldConstraints (BlisConstraint ** old) [inline]`

set old constraints.

Definition at line 732 of file BlisModel.h.

3.13.3.76 `void BlisModel::delOldConstraints () [inline]`

Set max number of old constraints.

Definition at line 735 of file BlisModel.h.

3.13.3.77 `BlisCutStrategy BlisModel::getCutStrategy () const [inline]`

Query constraint generation strategy.

Definition at line 742 of file BlisModel.h.

3.13.3.78 `void BlisModel::setCutStrategy (BlisCutStrategy u) [inline]`

Set constraint generation strategy.

Definition at line 747 of file BlisModel.h.

3.13.3.79 `int BlisModel::getCutGenerationFrequency () const [inline]`

Query constraint generation frequency.

Definition at line 750 of file BlisModel.h.

3.13.3.80 `void BlisModel::setCutStrategy (int f) [inline]`

Set constraint generation frequency.

Definition at line 753 of file BlisModel.h.

3.13.3.81 `int BlisModel::getDenseConCutoff () const [inline]`

Get the threshold to be considered as a dense constraint.

Definition at line 756 of file BlisModel.h.

3.13.3.82 `void BlisModel::setDenseConCutoff (int cutoff) [inline]`

Set the threshold to be considered as a dense constraint.

Definition at line 759 of file BlisModel.h.

3.13.3.83 `double* BlisModel::getConRandoms () const [inline]`

Get randoms for check parallel constraints.

Definition at line 762 of file BlisModel.h.

3.13.3.84 `void BlisModel::passInPriorities (const int * priorities, bool ifNotSimpleIntegers, int defaultValue = 1000)`

Pass in branching priorities.

If ifClique then priorities are on cliques otherwise priorities are on integer variables. Other type (if exists set to default) 1 is highest priority. (well actually -INT_MAX is but that's ugly) If hotstart > 0 then branches are created to force the variable to the value given by best solution. This enables a sort of hot start. The node choice should be greatest depth and hotstart should normally be switched off after a solution.

If ifNotSimpleIntegers true then appended to normal integers

3.13.3.85 `const int* BlisModel::priority () const [inline]`

Priorities.

Definition at line 787 of file BlisModel.h.

3.13.3.86 `virtual void BlisModel::modelLog () [virtual]`

Log of specific models.

3.13.3.87 `int BlisModel::getNumNodes () const [inline]`

Get how many Nodes it took to solve the problem.

Definition at line 808 of file BlisModel.h.

3.13.3.88 `int BlisModel::getNumIterations () const [inline]`

Get how many iterations it took to solve the problem.

Definition at line 811 of file BlisModel.h.

3.13.3.89 `int BlisModel::getAveIterations () const [inline]`

Get the average iterations it took to solve a lp.

Definition at line 814 of file BlisModel.h.

3.13.3.90 `void BlisModel::addNumNodes (int newNodes = 1) [inline]`

Increment node count.

Definition at line 817 of file BlisModel.h.

3.13.3.91 `void BlisModel::addNumIterations (int newIter) [inline]`

Increment Iteration count.

Definition at line 820 of file BlisModel.h.

3.13.3.92 `CoinMessageHandler* BlisModel::blisMessageHandler () const` `[inline]`

Get the message handler.

Definition at line 826 of file `BlisModel.h`.

3.13.3.93 `CoinMessages BlisModel::blisMessages ()` `[inline]`

Return messages.

Definition at line 830 of file `BlisModel.h`.

3.13.3.94 `BlisParams* BlisModel::BlisPar ()` `[inline]`

Access parameters.

Definition at line 834 of file `BlisModel.h`.

3.13.3.95 `virtual void BlisModel::nodeLog (AlpsTreeNode * node, bool force)` `[virtual]`

Node log.

3.13.3.96 `virtual bool BlisModel::fathomAllNodes ()` `[virtual]`

Return true, if all nodes can be fathomed.

3.13.3.97 `AlpsReturnStatus BlisModel::encodeBlis (AlpsEncoded * encoded) const` `[protected]`

Pack Blis portion of the model into an encoded object.

3.13.3.98 `AlpsReturnStatus BlisModel::decodeBlis (AlpsEncoded & encoded)` `[protected]`

Unpack Blis portion of the model from an encoded object.

3.13.3.99 `void BlisModel::packSharedPseudocost (AlpsEncoded * encoded, int numToShare)` `[protected]`

Retrieve and pack shared pseudocost.

3.13.3.100 `void BlisModel::packSharedConstraints (AlpsEncoded * encoded)` `[protected]`

Retrieve and pack shared constraints.

3.13.3.101 `void BlisModel::unpackSharedConstraints (AlpsEncoded & encoded)` `[protected]`

Unpack and store shared constraints.

3.13.3.102 `void BlisModel::packSharedVariables (AlpsEncoded * encoded)` `[protected]`

Retrieve and pack shared variables.

3.13.3.103 `void BlisModel::unpackSharedVariables (AlpsEncoded & encoded)` `[protected]`

Unpack and store shared variables.

3.13.3.104 `virtual void BlisModel::registerKnowledge ()` `[virtual]`

Register knowledge.

3.13.3.105 `virtual AlpsEncoded* BlisModel::encode () const` [virtual]

The method that encodes the model into an encoded object.

3.13.3.106 `virtual void BlisModel::decodeToSelf (AlpsEncoded &)` [virtual]

The method that decodes the model from an encoded object.

3.13.3.107 `virtual AlpsEncoded* BlisModel::packSharedKnowledge ()` [virtual]

Pack knowledge to be shared with others into an encoded object.

Return NULL means that no knowledge can be shared.

3.13.3.108 `virtual void BlisModel::unpackSharedKnowledge (AlpsEncoded &)` [virtual]

Unpack and store shared knowledge from an encoded object.

3.13.4 Member Data Documentation

3.13.4.1 `OsiSolverInterface* BlisModel::origLpSolver_` [protected]

Input by user.

Definition at line 78 of file BlisModel.h.

3.13.4.2 `OsiSolverInterface* BlisModel::presolvedLpSolver_` [protected]

Presolved.

Definition at line 80 of file BlisModel.h.

3.13.4.3 `OsiSolverInterface* BlisModel::lpSolver_` [protected]

Actually used.

If using presolve, then it is presolved; otherwise it is the original.

Definition at line 83 of file BlisModel.h.

3.13.4.4 `CoinPackedMatrix* BlisModel::colMatrix_` [protected]

Column majored matrix.

(For MPS file, etc.)

Definition at line 90 of file BlisModel.h.

3.13.4.5 `double* BlisModel::varLB_` [protected]

Variable and constraint bounds.

Definition at line 94 of file BlisModel.h.

3.13.4.6 `double BlisModel::objSense_` [protected]

Objective function.

Definition at line 109 of file BlisModel.h.

3.13.4.7 `int BlisModel::numIntObjects_` `[protected]`

Column types.

Definition at line 115 of file `BlisModel.h`.

3.13.4.8 `std::vector<BcpsVariable*> BlisModel::inputVar_` `[protected]`

User's input objects.

Definition at line 121 of file `BlisModel.h`.

3.13.4.9 `double BlisModel::incObjValue_` `[protected]`

Incumbent objective value.

Definition at line 143 of file `BlisModel.h`.

3.13.4.10 `double BlisModel::cutoff_` `[protected]`

Cutoff in lp solver.

Definition at line 149 of file `BlisModel.h`.

3.13.4.11 `double BlisModel::cutoffInc_` `[protected]`

Cutoff increment.

Definition at line 152 of file `BlisModel.h`.

3.13.4.12 `BcpsBranchStrategy* BlisModel::branchStrategy_` `[protected]`

Variable selection function.

Definition at line 170 of file `BlisModel.h`.

3.13.4.13 `int BlisModel::numObjects_` `[protected]`

Number of objects.

Definition at line 179 of file `BlisModel.h`.

3.13.4.14 `BcpsObject** BlisModel::objects_` `[protected]`

The set of objects.

Definition at line 182 of file `BlisModel.h`.

3.13.4.15 `char* BlisModel::sharedObjectMark_` `[protected]`

The objects that can be shared.

Definition at line 185 of file `BlisModel.h`.

3.13.4.16 `int* BlisModel::priority_` `[protected]`

Priorities of integer object.

Definition at line 188 of file `BlisModel.h`.

3.13.4.17 `AlpsTreeNode* BlisModel::activeNode_` [protected]

Active node.

Definition at line 191 of file `BlisModel.h`.

3.13.4.18 `int BlisModel::numStrong_` [protected]

Number of strong branching.

Definition at line 194 of file `BlisModel.h`.

3.13.4.19 `int BlisModel::numBranchResolve_` [protected]

Maximum number of resolve during branching.

Definition at line 200 of file `BlisModel.h`.

3.13.4.20 `int BlisModel::numHeuristics_` [protected]

Number of heuristics.

Definition at line 207 of file `BlisModel.h`.

3.13.4.21 `BlisHeuristic** BlisModel::heuristics_` [protected]

The list of heuristics.

Definition at line 210 of file `BlisModel.h`.

3.13.4.22 `BlisCutStrategy BlisModel::cutStrategy_` [protected]

If use cut generators.

Definition at line 217 of file `BlisModel.h`.

3.13.4.23 `int BlisModel::numCutGenerators_` [protected]

Number of cut generators used.

Definition at line 223 of file `BlisModel.h`.

3.13.4.24 `int BlisModel::maxNumCons_` [protected]

Number of cuts can be generators.

Definition at line 226 of file `BlisModel.h`.

3.13.4.25 `BlisConGenerator** BlisModel::generators_` [protected]

The list of cut generators used.

Definition at line 229 of file `BlisModel.h`.

3.13.4.26 `BcpsConstraintPool* BlisModel::constraintPool_` [protected]

Store all the cuts.

Definition at line 232 of file `BlisModel.h`.

3.13.4.27 BlisConstraint BlisModel::oldConstraints_** [protected]

Temporary store old cuts at a node when installing a node.

Definition at line 235 of file BlisModel.h.

3.13.4.28 int BlisModel::oldConstraintsSize_ [protected]

The memory size allocated for oldConstraints_.

Definition at line 238 of file BlisModel.h.

3.13.4.29 int BlisModel::numOldConstraints_ [protected]

Number of old constraints.

Definition at line 241 of file BlisModel.h.

3.13.4.30 double* BlisModel::conRandoms_ [protected]

Random keys.

Definition at line 244 of file BlisModel.h.

3.13.4.31 BlisParams* BlisModel::BlisPar_ [protected]

Blis parameters.

Definition at line 254 of file BlisModel.h.

3.13.4.32 CoinMessageHandler* BlisModel::blisMessageHandler_ [protected]

Message handler.

Definition at line 257 of file BlisModel.h.

3.13.4.33 CoinMessages BlisModel::blisMessages_ [protected]

Blis messages.

Definition at line 260 of file BlisModel.h.

3.13.4.34 int BlisModel::numNodes_ [protected]

Number of processed nodes.

Definition at line 263 of file BlisModel.h.

3.13.4.35 int BlisModel::numIterations_ [protected]

Number of Ip(Simplex) iterations.

Definition at line 266 of file BlisModel.h.

3.13.4.36 int BlisModel::aveIterations_ [protected]

Average number of Ip iterations to solve a subproblem.

Definition at line 269 of file BlisModel.h.

3.13.4.37 `int* BlisModel::tempVarLBPos_` `[protected]`

Tempory storage for var/con indices.

Definition at line 277 of file BlisModel.h.

3.13.4.38 `BcpsConstraintPool* BlisModel::constraintPoolSend_` `[protected]`

Constraints that can be sent/broadcasted to other processes.

Definition at line 288 of file BlisModel.h.

3.13.4.39 `BcpsConstraintPool* BlisModel::constraintPoolReceive_` `[protected]`

Constraints that are received from other processses.

Definition at line 291 of file BlisModel.h.

3.13.4.40 `bool BlisModel::isRoot_`

If root node.

Definition at line 296 of file BlisModel.h.

3.13.4.41 `int BlisModel::boundingPass_`

The number of passes during bounding procedure.

Definition at line 299 of file BlisModel.h.

3.13.4.42 `double BlisModel::integerTol_`

Integer tolerance.

Definition at line 302 of file BlisModel.h.

3.13.4.43 `double BlisModel::optimalRelGap_`

Input relative optimal gap.

Definition at line 305 of file BlisModel.h.

3.13.4.44 `double BlisModel::optimalAbsGap_`

Input absolute optimal gap.

Definition at line 308 of file BlisModel.h.

3.13.4.45 `double BlisModel::currRelGap_`

Current relative optimal gap.

Definition at line 311 of file BlisModel.h.

3.13.4.46 `double BlisModel::currAbsGap_`

Current absolute optimal gap.

Definition at line 314 of file BlisModel.h.

3.13.4.47 `BlisHeurStrategy` `BlisModel::heurStrategy_`

If use heuristics.

Definition at line 317 of file `BlisModel.h`.

3.13.4.48 `int` `BlisModel::heurCallFrequency_`

Frequency of using heuristics.

Definition at line 320 of file `BlisModel.h`.

3.13.4.49 `OsiCuts` `BlisModel::newCutPool_`

Store new cuts in each pass.

Definition at line 323 of file `BlisModel.h`.

3.13.4.50 `std::vector<AlpsTreeNode*>` `BlisModel::leafToRootPath`

Record the path from leaf to root.

Definition at line 326 of file `BlisModel.h`.

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

- `BlisModel.h`

3.14 BlisNodeDesc Class Reference

Public Member Functions

- `BlisNodeDesc` ()
Default constructor.
- `BlisNodeDesc` (`BlisModel` *m)
Useful constructor.
- virtual `~BlisNodeDesc` ()
Destructor.
- void `setBasis` (`CoinWarmStartBasis` *&ws)
Set basis.
- `CoinWarmStartBasis` * `getBasis` () const
Get warm start basis.
- void `setBranchedDir` (int d)
Set branching direction.
- int `getBranchedDir` () const
Get branching direction.
- void `setBranchedInd` (int d)
Set branching object index.
- int `getBranchedInd` () const
Get branching object index.
- void `setBranchedVal` (double d)
Set branching value.
- double `getBranchedVal` () const
Get branching direction.

- virtual AlpsReturnStatus [encode](#) (AlpsEncoded *encoded) const
Pack node description into an encoded.
- virtual AlpsReturnStatus [decode](#) (AlpsEncoded &encoded)
Unpack a node description from an encoded.

Protected Member Functions

- AlpsReturnStatus [encodeBlis](#) (AlpsEncoded *encoded) const
Pack blis portion of node description into an encoded.
- AlpsReturnStatus [decodeBlis](#) (AlpsEncoded &encoded)
Unpack blis portion of node description from an encoded.

3.14.1 Detailed Description

Definition at line 40 of file BlisNodeDesc.h.

3.14.2 Constructor & Destructor Documentation

3.14.2.1 BlisNodeDesc::BlisNodeDesc () [\[inline\]](#)

Default constructor.

Definition at line 59 of file BlisNodeDesc.h.

3.14.2.2 BlisNodeDesc::BlisNodeDesc (BlisModel * m) [\[inline\]](#)

Useful constructor.

Definition at line 68 of file BlisNodeDesc.h.

3.14.2.3 virtual BlisNodeDesc::~BlisNodeDesc () [\[inline\]](#), [\[virtual\]](#)

Destructor.

Definition at line 78 of file BlisNodeDesc.h.

3.14.3 Member Function Documentation

3.14.3.1 void BlisNodeDesc::setBasis (CoinWarmStartBasis *& ws) [\[inline\]](#)

Set basis.

Definition at line 81 of file BlisNodeDesc.h.

3.14.3.2 CoinWarmStartBasis* BlisNodeDesc::getBasis () const [\[inline\]](#)

Get warm start basis.

Definition at line 88 of file BlisNodeDesc.h.

3.14.3.3 void BlisNodeDesc::setBranchedDir (int d) [\[inline\]](#)

Set branching direction.

Definition at line 91 of file BlisNodeDesc.h.

3.14.3.4 `int BlisNodeDesc::getBranchedDir () const [inline]`

Get branching direction.

Definition at line 94 of file `BlisNodeDesc.h`.

3.14.3.5 `void BlisNodeDesc::setBranchedInd (int d) [inline]`

Set branching object index.

Definition at line 97 of file `BlisNodeDesc.h`.

3.14.3.6 `int BlisNodeDesc::getBranchedInd () const [inline]`

Get branching object index.

Definition at line 100 of file `BlisNodeDesc.h`.

3.14.3.7 `void BlisNodeDesc::setBranchedVal (double d) [inline]`

Set branching value.

Definition at line 103 of file `BlisNodeDesc.h`.

3.14.3.8 `double BlisNodeDesc::getBranchedVal () const [inline]`

Get branching direction.

Definition at line 106 of file `BlisNodeDesc.h`.

3.14.3.9 `AlpsReturnStatus BlisNodeDesc::encodeBlis (AlpsEncoded * encoded) const [inline],[protected]`

Pack blis portion of node description into an encoded.

Definition at line 111 of file `BlisNodeDesc.h`.

3.14.3.10 `AlpsReturnStatus BlisNodeDesc::decodeBlis (AlpsEncoded & encoded) [inline],[protected]`

Unpack blis portion of node description from an encoded.

Definition at line 133 of file `BlisNodeDesc.h`.

3.14.3.11 `virtual AlpsReturnStatus BlisNodeDesc::encode (AlpsEncoded * encoded) const [inline],[virtual]`

Pack node description into an encoded.

Definition at line 157 of file `BlisNodeDesc.h`.

3.14.3.12 `virtual AlpsReturnStatus BlisNodeDesc::decode (AlpsEncoded & encoded) [inline],[virtual]`

Unpack a node description from an encoded.

Fill member data.

Definition at line 167 of file `BlisNodeDesc.h`.

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

- `BlisNodeDesc.h`

3.15 BlisObjectInt Class Reference

Public Member Functions

- [BlisObjectInt](#) ()
Default Constructor.
- [BlisObjectInt](#) (int objectIndex, int iColumn, double lb, double ub, double [breakEven](#)=0.5)
Useful constructor - passed integer index and model index.
- virtual [~BlisObjectInt](#) ()
Destructor.
- [BlisObjectInt](#) (const [BlisObjectInt](#) &)
Copy constructor.
- virtual [BcpsObject](#) * [clone](#) () const
Clone an object.
- [BlisObjectInt](#) & [operator=](#) (const [BlisObjectInt](#) &rhs)
Assignment operator.
- virtual double [infeasibility](#) ([BcpsModel](#) *m, int &preferredWay) const
Infeasibility.
- virtual void [feasibleRegion](#) ([BcpsModel](#) *m)
Set bounds to contain the current solution.
- virtual [BcpsBranchObject](#) * [createBranchObject](#) ([BcpsModel](#) *m, int direction) const
Creates a branching object.
- virtual [BcpsBranchObject](#) * [preferredNewFeasible](#) ([BcpsModel](#) *m) const
Given a valid solution (with reduced costs, etc.), return a branching object which would give a new feasible point in the good direction.
- virtual [BcpsBranchObject](#) * [notPreferredNewFeasible](#) ([BcpsModel](#) *m) const
Given a valid solution (with reduced costs, etc.), return a branching object which would give a new feasible point in a bad direction.
- virtual void [resetBounds](#) ([BcpsModel](#) *m)
Reset original upper and lower bound values from the solver.
- virtual int [columnIndex](#) () const
Column number if single column object, otherwise.
- double [breakEven](#) () const
Breakeven e.g 0.7 -> >= 0.7 go up first.
- void [setBreakEven](#) (double value)
Set breakeven e.g 0.7 -> >= 0.7 go up first.
- [BlisPseudocost](#) & [pseudocost](#) ()
Access pseudocost.

Get or set Original bounds.

- double [originalLowerBound](#) () const
- void [setOriginalLowerBound](#) (double value)
- double [originalUpperBound](#) () const
- void [setOriginalUpperBound](#) (double value)

Protected Attributes

- int `columnIndex_`
Column index in the lp model.
- double `originalLower_`
Original lower bound.
- double `originalUpper_`
Original upper bound.
- double `breakEven_`
Breakeven i.e.
- `BlisPseudocost` `pseudocost_`
Pseudo cost.

3.15.1 Detailed Description

Definition at line 36 of file `BlisObjectInt.h`.

3.15.2 Constructor & Destructor Documentation

3.15.2.1 `BlisObjectInt::BlisObjectInt ()`

Default Constructor.

3.15.2.2 `BlisObjectInt::BlisObjectInt (int objectIndex, int iColumn, double lb, double ub, double breakEven = 0.5)`

Useful constructor - passed integer index and model index.

3.15.2.3 `virtual BlisObjectInt::~~BlisObjectInt () [inline], [virtual]`

Destructor.

Definition at line 68 of file `BlisObjectInt.h`.

3.15.2.4 `BlisObjectInt::BlisObjectInt (const BlisObjectInt &)`

Copy constructor.

3.15.3 Member Function Documentation

3.15.3.1 `virtual BcpsObject* BlisObjectInt::clone () const [inline], [virtual]`

Clone an object.

Definition at line 74 of file `BlisObjectInt.h`.

3.15.3.2 `BlisObjectInt& BlisObjectInt::operator= (const BlisObjectInt & rhs)`

Assignment operator.

3.15.3.3 `virtual double BlisObjectInt::infeasibility (BcpsModel * m, int & preferredWay) const [virtual]`

Infeasibility.

Range is [0.0, 0.5].

Parameters

<i>PreferredWay</i>	the direction close to an integer.
---------------------	------------------------------------

3.15.3.4 `virtual void BlisObjectInt::feasibleRegion (BcpsModel * m) [virtual]`

Set bounds to contain the current solution.

More precisely, for the variable associated with this object, take the value given in the current solution, force it within the current bounds if required, then set the bounds to fix the variable at the integer nearest the solution value.

3.15.3.5 `virtual BcpsBranchObject* BlisObjectInt::createBranchObject (BcpsModel * m, int direction) const [virtual]`

Creates a branching object.

3.15.3.6 `virtual BcpsBranchObject* BlisObjectInt::preferredNewFeasible (BcpsModel * m) const [virtual]`

Given a valid solution (with reduced costs, etc.), return a branching object which would give a new feasible point in the good direction.

The preferred branching object will force the variable to be +/-1 from its current value, depending on the reduced cost and objective sense. If movement in the direction which improves the objective is impossible due to bounds on the variable, the branching object will move in the other direction. If no movement is possible, the method returns NULL.

Only the bounds on this variable are considered when determining if the new point is feasible.

3.15.3.7 `virtual BcpsBranchObject* BlisObjectInt::notPreferredNewFeasible (BcpsModel * m) const [virtual]`

Given a valid solution (with reduced costs, etc.), return a branching object which would give a new feasible point in a bad direction.

As for [preferredNewFeasible\(\)](#), but the preferred branching object will force movement in a direction that degrades the objective.

3.15.3.8 `virtual void BlisObjectInt::resetBounds (BcpsModel * m) [virtual]`

Reset original upper and lower bound values from the solver.

Handy for updating bounds held in this object after bounds held in the solver have been tightened.

3.15.3.9 `virtual int BlisObjectInt::columnIndex () const [inline],[virtual]`

Column number if single column object, otherwise.

Definition at line 129 of file BlisObjectInt.h.

3.15.3.10 `double BlisObjectInt::breakEven () const [inline]`

Breakeven e.g 0.7 -> >= 0.7 go up first.

Definition at line 140 of file BlisObjectInt.h.

3.15.3.11 `void BlisObjectInt::setBreakEven (double value) [inline]`

Set breakeven e.g 0.7 -> >= 0.7 go up first.

Definition at line 143 of file BlisObjectInt.h.

3.15.3.12 `BlisPseudocost& BlisObjectInt::pseudocost () [inline]`

Access pseudocost.

Definition at line 146 of file BlisObjectInt.h.

3.15.4 Member Data Documentation

3.15.4.1 `int BlisObjectInt::columnIndex_` [protected]

Column index in the lp model.

Definition at line 41 of file BlisObjectInt.h.

3.15.4.2 `double BlisObjectInt::originalLower_` [protected]

Original lower bound.

Definition at line 44 of file BlisObjectInt.h.

3.15.4.3 `double BlisObjectInt::originalUpper_` [protected]

Original upper bound.

Definition at line 47 of file BlisObjectInt.h.

3.15.4.4 `double BlisObjectInt::breakEven_` [protected]

Breakeven i.e.

\geq this preferred is up.

Definition at line 50 of file BlisObjectInt.h.

3.15.4.5 `BlisPseudocost BlisObjectInt::pseudocost_` [protected]

Pseudo cost.

Definition at line 53 of file BlisObjectInt.h.

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

- BlisObjectInt.h

3.16 BlisParams Class Reference

Public Types

- enum `chrParams` {
`cutRampUp`, `presolve`, `shareConstraints`, `shareVariables`,
`sharePseudocostRampUp`, `sharePseudocostSearch` }
Character parameters.
- enum `intParams` {
`branchStrategy`, `cutStrategy`, `cutGenerationFrequency`, `quickCutPass`,
`cutCliqueStrategy`, `difference`, `heurStrategy`, `heurCallFrequency`,
`lookAhead`, `pseudoReliability`, `sharePcostDepth`, `sharePcostFrequency`,
`strongCandSize` }
Integer paramters.
- enum `dblParams` {
`cutFactor`, `cutoffInc`, `denseConFactor`,
`integerTol`, `objSense`, `optimalRelGap`, `optimalAbsGap`,

`pseudoWeight, scaleConFactor, tailOff }`

Double parameters.

- enum `strParams`

String parameters.

- enum `strArrayParams`

There are no string array parameters.

Public Member Functions

- virtual void `createKeywordList ()`

Method for creating the list of keyword looked for in the parameter file.

- virtual void `setDefaultEntries ()`

Method for setting the default values for the parameters.

- void `setEntry (const chrParams key, const char *val)`

char is true(1) or false(0), not used*

- void `setEntry (const chrParams key, const char val)`

char is true(1) or false(0), not used

- void `setEntry (const chrParams key, const bool val)`

This method is the one that ever been used.

Constructors.

- `BlisParams ()`

The default constructor creates a parameter set with from the template argument structure.

Query methods

For user application: Following code are do NOT need to change.

The reason can not put following functions in base class `AlpsParameterSet` is that `chrParams` and `end-OfChrParams` etc., are NOT the same as those declared in base class.

The members of the parameter set can be queried for using the overloaded `entry()` method. Using the example in the class documentation the user can get a parameter with the "`<code>param.entry(USER_par::parameter_name)</code>`" expression.

- bool `entry (const chrParams key) const`
- int `entry (const intParams key) const`
- double `entry (const dblParams key) const`
- const std::string & `entry (const strParams key) const`
- const std::vector< std::string > & `entry (const strArrayParams key) const`

Packing/unpacking methods

- void `pack (AlpsEncoded &buf)`
Pack the parameter set into the buffer (AlpsEncoded is used as buffer Here).
- void `unpack (AlpsEncoded &buf)`
Unpack the parameter set from the buffer.

3.16.1 Detailed Description

Definition at line 35 of file `BlisParams.h`.

3.16.2 Member Enumeration Documentation

3.16.2.1 enum `BlisParams::chrParams`

Character parameters.

All of these variable are used as booleans (ture = 1, false = 0).

Enumerator:

- cutRampUp*** Generate cuts during rampup. Default: true
- presolve*** Presolve or not.
- shareConstraints*** Share constraints Default: false.
- shareVariables*** Share constraints Default: false.
- sharePseudocostRampUp*** Share pseudocost during ramp up. Default: true
- sharePseudocostSearch*** Share pseudocost during search Default: false.

Definition at line 39 of file `BlisParams.h`.

3.16.2.2 enum `BlisParams::intParams`

Integer paramters.

Enumerator:

- branchStrategy*** Branching strategy. 0: max infeasibility, 1: pseudocost, 2: relibility, 3: strong branching. 4: bilevel branching
- cutStrategy*** Cut generators control. -2: root, -1: auto, 0: disable, any positive frequency
- cutGenerationFrequency*** All constraint generators.
- quickCutPass*** The pass to generate cuts.
- cutCliquesStrategy*** The pass to generate cuts for quick branching.
- difference*** -1 auto, 0, no, any integer frequency
- heurStrategy*** Heuristics control. `BlisHeurStrategyRoot`: root, `BlisHeurStrategyAuto`: auto, `BlisHuerStrategyNone`: disable, `BlisHeurStrategyPeriodic`: every 't' nodes
- heurCallFrequency*** All heuristics.
- lookAhead*** The look ahead of pseudocost.
- pseudoReliability*** The relibility of pseudocost.
- sharePcostDepth*** Maximum tree depth of sharing pseudocost.
- sharePcostFrequency*** Frequency of sharing pseudocost.
- strongCandSize*** The number of candidate used in strong branching. Default: 10.

Definition at line 62 of file `BlisParams.h`.

3.16.2.3 enum `BlisParams::dblParams`

Double parameters.

Enumerator:

- cutFactor*** Limit the max number cuts applied at a node. $\text{maxNumCons} = (\text{CutFactor} - 1) * \text{numCoreConstraints}$.
- cutoff*** Cutoff any nodes whose objective value is higher than it.

cutoffInc The value added to relaxation value when deciding fathom. Default:1.0e-6

denseConFactor Dense constraint factor.

integerTol Tolerance to treat as an integer. Default: 1.0e-5

objSense Objective sense: min = 1.0, max = -1.0.

optimalRelGap If the relative gap between best feasible and best relaxed fall into this gap, search stops. Default: 1.0e-6

optimalAbsGap If the absolute gap between best feasible and best relaxed fall into this gap, search stops. Default: 1.0e-4

pseudoWeight Weight used to calculate pseudocost.

scaleConFactor Scaling indicator of a constraint.

tailOff Tail off.

Definition at line 133 of file BlisParams.h.

3.16.2.4 enum BlisParams::strParams

String parameters.

Definition at line 174 of file BlisParams.h.

3.16.2.5 enum BlisParams::strArrayParams

There are no string array parameters.

Definition at line 181 of file BlisParams.h.

3.16.3 Constructor & Destructor Documentation

3.16.3.1 BlisParams::BlisParams () [inline]

The default constructor creates a parameter set with from the template argument structure.

The keyword list is created and the defaults are set.

Definition at line 193 of file BlisParams.h.

3.16.4 Member Function Documentation

3.16.4.1 virtual void BlisParams::createKeywordList () [virtual]

Method for creating the list of keyword looked for in the parameter file.

3.16.4.2 virtual void BlisParams::setDefaultEntries () [virtual]

Method for setting the default values for the parameters.

3.16.4.3 void BlisParams::pack (AlpsEncoded & buf) [inline]

Pack the parameter set into the buffer (AlpsEncoded is used as buffer Here).

Definition at line 284 of file BlisParams.h.

3.16.4.4 void BlisParams::unpack (AlpsEncoded & buf) [inline]

Unpack the parameter set from the buffer.

Definition at line 297 of file BlisParams.h.

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

- BlisParams.h

3.17 BlisPresolve Class Reference

A interface to Osi/Coin Presolve.

```
#include <BlisPresolve.h>
```

Public Member Functions

- [BlisPresolve](#) ()
Default constructor (empty object)
- virtual [~BlisPresolve](#) ()
Virtual destructor.
- virtual OsiSolverInterface * [preprocess](#) (OsiSolverInterface &origModel, double feasibilityTolerance=0.0, bool keepIntegers=true, int numberPasses=5, const char *prohibited=NULL)
Presolve.
- virtual void [postprocess](#) (bool updateStatus=true)
Postsolve.

3.17.1 Detailed Description

A interface to Osi/Coin Presolve.

Definition at line 37 of file BlisPresolve.h.

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

- BlisPresolve.h

3.18 BlisPseudocost Class Reference

Public Member Functions

- [BlisPseudocost](#) ()
Default constructor.
- [BlisPseudocost](#) (double uc, int un, double dc, int dn, double s)
Useful constructor.
- [BlisPseudocost](#) (const [BlisPseudocost](#) &cost)
Copy constructor.
- [BlisPseudocost](#) & [operator=](#) (const [BlisPseudocost](#) &cost)
Overload operator =.
- void [setWeight](#) (double w)

- *Set weigth.*
- void **update** (const int dir, const double parentObjValue, const double objValue, const double solValue)
- *Update pseudocost.*
- void **update** (const int dir, const double objDiff, const double solValue)
- *Update pseudocost.*
- void **update** (double upCost, int upCount, double downCost, int downCount)
- *Update pseudocost.*
- int **getUpCount** ()
- *Get up branching count.*
- double **getUpCost** ()
- *Get up branching cost.*
- int **getDownCount** ()
- *Get down branching count.*
- double **getDownCost** ()
- *Get down branching cost.*
- double **getScore** ()
- *Get importance.*
- void **setScore** (double s)
- *Set importance.*
- AlpsReturnStatus **encodeTo** (AlpsEncoded *encoded) const
- *Pack pseudocost to the given object.*
- AlpsReturnStatus **decodeFrom** (AlpsEncoded &encoded)
- *Unpack pseudocost from the given encode object.*
- virtual AlpsEncoded * **encode** () const
- *Encode this node for message passing.*
- virtual AlpsKnowledge * **decode** (AlpsEncoded &) const
- *Decode a node from an encoded object.*

3.18.1 Detailed Description

Definition at line 32 of file BlisPseudo.h.

3.18.2 Constructor & Destructor Documentation

3.18.2.1 BlisPseudocost::BlisPseudocost () [inline]

Default constructor.

Definition at line 58 of file BlisPseudo.h.

3.18.2.2 BlisPseudocost::BlisPseudocost (double uc, int un, double dc, int dn, double s) [inline]

Useful constructor.

Definition at line 68 of file BlisPseudo.h.

3.18.3 Member Function Documentation

3.18.3.1 void BlisPseudocost::setWeight (double *w*) [inline]

Set weight.

Definition at line 104 of file BlisPseudo.h.

3.18.3.2 void BlisPseudocost::update (const int *dir*, const double *parentObjValue*, const double *objValue*, const double *solValue*)

Update pseudocost.

3.18.3.3 void BlisPseudocost::update (const int *dir*, const double *objDiff*, const double *solValue*)

Update pseudocost.

3.18.3.4 void BlisPseudocost::update (double *upCost*, int *upCount*, double *downCost*, int *downCount*)

Update pseudocost.

3.18.3.5 int BlisPseudocost::getUpCount () [inline]

Get up branching count.

Definition at line 130 of file BlisPseudo.h.

3.18.3.6 double BlisPseudocost::getUpCost () [inline]

Get up branching cost.

Definition at line 133 of file BlisPseudo.h.

3.18.3.7 int BlisPseudocost::getDownCount () [inline]

Get down branching count.

Definition at line 136 of file BlisPseudo.h.

3.18.3.8 double BlisPseudocost::getDownCost () [inline]

Get down branching cost.

Definition at line 139 of file BlisPseudo.h.

3.18.3.9 double BlisPseudocost::getScore () [inline]

Get importance.

Definition at line 142 of file BlisPseudo.h.

3.18.3.10 void BlisPseudocost::setScore (double *s*) [inline]

Set importance.

Definition at line 145 of file BlisPseudo.h.

3.18.3.11 AlpsReturnStatus BlisPseudocost::encodeTo (AlpsEncoded * *encoded*) const

Pack pseudocost to the given object.

3.18.3.12 AlpsReturnStatus BlisPseudocost::decodeFrom (AlpsEncoded & encoded)

Unpack pseudocost from the given encode object.

3.18.3.13 virtual AlpsEncoded* BlisPseudocost::encode () const [virtual]

Encode this node for message passing.

3.18.3.14 virtual AlpsKnowledge* BlisPseudocost::decode (AlpsEncoded &) const [virtual]

Decode a node from an encoded object.

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

- BlisPseudo.h

3.19 BlisSolution Class Reference

This class contains the solutions generated by the LP solver (either primal or dual.

```
#include <BlisSolution.h>
```

Public Member Functions

- [BlisSolution](#) ()
Default constructor.
- [BlisSolution](#) (int s, const double *values, double objValue)
Useful constructor.
- virtual [~BlisSolution](#) ()
Destructor.
- virtual void [print](#) (std::ostream &os) const
Print out the solution.
- virtual AlpsEncoded * [encode](#) () const
The method that encodes the solution into a encoded object.
- virtual AlpsKnowledge * [decode](#) (AlpsEncoded &encoded) const
The method that decodes the solution from a encoded object.

3.19.1 Detailed Description

This class contains the solutions generated by the LP solver (either primal or dual.

The class exists primarily to pass solutions to the object generator(s).

Definition at line 36 of file BlisSolution.h.

3.19.2 Constructor & Destructor Documentation

3.19.2.1 BlisSolution::BlisSolution () [inline]

Default constructor.

Definition at line 43 of file BlisSolution.h.

3.19.2.2 `BlisSolution::BlisSolution (int s, const double * values, double objValue) [inline]`

Useful constructor.

Definition at line 49 of file `BlisSolution.h`.

3.19.2.3 `virtual BlisSolution::~~BlisSolution () [inline],[virtual]`

Destructor.

Definition at line 55 of file `BlisSolution.h`.

3.19.3 Member Function Documentation

3.19.3.1 `virtual void BlisSolution::print (std::ostream & os) const [inline],[virtual]`

Print out the solution.

Print the solution.

Definition at line 59 of file `BlisSolution.h`.

3.19.3.2 `virtual AlpsEncoded* BlisSolution::encode () const [inline],[virtual]`

The method that encodes the solution into a encoded object.

Definition at line 80 of file `BlisSolution.h`.

3.19.3.3 `virtual AlpsKnowledge* BlisSolution::decode (AlpsEncoded & encoded) const [inline],[virtual]`

The method that decodes the solution from a encoded object.

Definition at line 88 of file `BlisSolution.h`.

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

- `BlisSolution.h`

3.20 BlisStrong Struct Reference

3.20.1 Detailed Description

Definition at line 41 of file `BlisBranchStrategyStrong.h`.

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

- `BlisBranchStrategyStrong.h`

3.21 BlisTreeNode Class Reference

This is the class in which we are finally able to concretely define the bounding procedure.

```
#include <BlisSubTree.h>
```

Public Member Functions

- [BlisTreeNode](#) ()

- *Default constructor.*
- `BlisTreeNode (BlisModel *m)`
- *Useful constructor.*
- `BlisTreeNode (AlpsNodeDesc *&desc)`
- *Useful constructor.*
- `virtual ~BlisTreeNode ()`
- *Destructor.*
- `void init ()`
- *Initilize member data when constructing a node.*
- `AlpsTreeNode * createNewTreeNode (AlpsNodeDesc *&desc) const`
- *Create a new node based on given desc.*
- `virtual int installSubProblem (BcpsModel *mode)`
- *intall subproblem*
- `virtual int process (bool isRoot=false, bool rampUp=false)`
- *Performing the bounding operation.*
- `virtual int bound (BcpsModel *model)`
- *Bounding procedure.*
- `virtual std::vector`
`< CoinTriple< AlpsNodeDesc`
`*, AlpsNodeStatus, double > > branch ()`
- *Takes the explicit description of the current active node and creates the children's descriptions, which contain information about how the branching is to be done.*
- `int selectBranchObject (BlisModel *model, bool &foundSol, int numPassesLeft)`
- *Select a branching object based on give branching strategy.*
- `virtual int chooseBranchingObject (BcpsModel *)`
- *To be defined.*
- `int generateConstraints (BlisModel *model, BcpsConstraintPool &conPool)`
- *Generate constraints.*
- `int callHeuristics (BlisModel *model, bool onlyBeforeRoot=false)`
- *Call heuristic to search solutions.*
- `void getViolatedConstraints (BlisModel *model, const double *currLpSolution, BcpsConstraintPool &conPool)`
- *Get violated constraints.*
- `BlisReturnStatus applyConstraints (BlisModel *model, const double *solution, BcpsConstraintPool &conPool)`
- *Select and apply constraints.*
- `BlisReturnStatus reducedCostFix (BlisModel *model)`
- *Fix and tighten varaibles based optimality conditions.*
- `virtual AlpsEncoded * encode () const`
- *Encode this node for message passing.*
- `virtual AlpsKnowledge * decode (AlpsEncoded &) const`
- *Decode a node from an encoded object.*
- `virtual void convertToExplicit ()`
- *Convert explicit description to difference, and vise-versa.*

3.21.1 Detailed Description

This is the class in which we are finally able to concretely define the bounding procedure.

Here we can assume that we have an LP solver and that the objects are cuts and variables, etc.

Definition at line 33 of file BlisSubTree.h.

3.21.2 Constructor & Destructor Documentation

3.21.2.1 `BlisTreeNode::BlisTreeNode () [inline]`

Default constructor.

Definition at line 79 of file `BlisTreeNode.h`.

3.21.2.2 `BlisTreeNode::BlisTreeNode (BlisModel * m) [inline]`

Useful constructor.

Definition at line 85 of file `BlisTreeNode.h`.

3.21.2.3 `BlisTreeNode::BlisTreeNode (AlpsNodeDesc *& desc) [inline]`

Useful constructor.

Definition at line 91 of file `BlisTreeNode.h`.

3.21.2.4 `virtual BlisTreeNode::~~BlisTreeNode () [inline], [virtual]`

Destructor.

Definition at line 98 of file `BlisTreeNode.h`.

3.21.3 Member Function Documentation

3.21.3.1 `void BlisTreeNode::init () [inline]`

Initilize member data when constructing a node.

Definition at line 103 of file `BlisTreeNode.h`.

3.21.3.2 `AlpsTreeNode* BlisTreeNode::createNewTreeNode (AlpsNodeDesc *& desc) const`

Create a new node based on given desc.

3.21.3.3 `virtual int BlisTreeNode::process (bool isRoot = false, bool rampUp = false) [virtual]`

Performing the bounding operation.

3.21.3.4 `virtual std::vector< CoinTriple<AlpsNodeDesc*, AlpsNodeStatus, double> > BlisTreeNode::branch () [virtual]`

Takes the explicit description of the current active node and creates the children's descriptions, which contain information about how the branching is to be done.

The stati of the children are `AlpsNodeStatusCandidate`.

3.21.3.5 `int BlisTreeNode::selectBranchObject (BlisModel * model, bool & foundSol, int numPassesLeft)`

Select a branching object based on give branching strategy.

3.21.3.6 `virtual int BlisTreeNode::chooseBranchingObject (BcpsModel *) [inline], [virtual]`

To be defined.

Definition at line 139 of file `BlisTreeNode.h`.

3.21.3.7 `int BlisTreeNode::generateConstraints (BlisModel * model, BcpsConstraintPool & conPool)`

Generate constraints.

3.21.3.8 `int BlisTreeNode::callHeuristics (BlisModel * model, bool onlyBeforeRoot = false)`

Call heuristic to search solutions.

0: no solution; 1: found solutions; 2: fathom this node. onlyBeforeRoot is for heuristics like feasibility pump.

3.21.3.9 `void BlisTreeNode::getViolatedConstraints (BlisModel * model, const double * currLpSolution, BcpsConstraintPool & conPool)`

Get violated constraints.

3.21.3.10 `BlisReturnStatus BlisTreeNode::applyConstraints (BlisModel * model, const double * solution, BcpsConstraintPool & conPool)`

Select and apply constraints.

3.21.3.11 `BlisReturnStatus BlisTreeNode::reducedCostFix (BlisModel * model)`

Fix and tighten variables based optimality conditions.

3.21.3.12 `virtual AlpsEncoded* BlisTreeNode::encode () const` [virtual]

Encode this node for message passing.

3.21.3.13 `virtual AlpsKnowledge* BlisTreeNode::decode (AlpsEncoded &) const` [virtual]

Decode a node from an encoded object.

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

- BlisSubTree.h
- BlisTreeNode.h

3.22 BlisVariable Class Reference

Public Member Functions

- virtual AlpsReturnStatus [encode](#) (AlpsEncoded *encoded)
Pack to a encode object.
- virtual AlpsKnowledge * [decode](#) (AlpsEncoded &encoded) const
Decode a variable from an encoded object.
- double [getObjCoef](#) ()
Return data.
- void [setData](#) (int s, const int *ind, const double *val)
Set data.

Protected Member Functions

- AlpsReturnStatus [encodeBlis](#) (AlpsEncoded *encoded)

Pack Blis part into an encoded object.

- `AlpsReturnStatus decodeBlis` (`AlpsEncoded &encoded`)

Unpack Blis part from a encode object.

3.22.1 Detailed Description

Definition at line 31 of file `BlisVariable.h`.

3.22.2 Member Function Documentation

3.22.2.1 `AlpsReturnStatus BlisVariable::encodeBlis (AlpsEncoded * encoded)` `[inline]`, `[protected]`

Pack Blis part into an encoded object.

Definition at line 106 of file `BlisVariable.h`.

3.22.2.2 `AlpsReturnStatus BlisVariable::decodeBlis (AlpsEncoded & encoded)` `[inline]`, `[protected]`

Unpack Blis part from a encode object.

Definition at line 119 of file `BlisVariable.h`.

3.22.2.3 `virtual AlpsReturnStatus BlisVariable::encode (AlpsEncoded * encoded)` `[inline]`, `[virtual]`

Pack to a encode object.

Definition at line 135 of file `BlisVariable.h`.

3.22.2.4 `virtual AlpsKnowledge* BlisVariable::decode (AlpsEncoded & encoded) const` `[inline]`, `[virtual]`

Decode a variable from an encoded object.

Definition at line 145 of file `BlisVariable.h`.

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

- `BlisVariable.h`

Index

- ~BlisBranchObjectBilevel
 - BlisBranchObjectBilevel, [4](#)
- ~BlisBranchObjectInt
 - BlisBranchObjectInt, [7](#)
- ~BlisBranchStrategyBilevel
 - BlisBranchStrategyBilevel, [9](#)
- ~BlisBranchStrategyMaxInf
 - BlisBranchStrategyMaxInf, [11](#)
- ~BlisBranchStrategyPseudo
 - BlisBranchStrategyPseudo, [12](#)
- ~BlisBranchStrategyRel
 - BlisBranchStrategyRel, [14](#)
- ~BlisBranchStrategyStrong
 - BlisBranchStrategyStrong, [15](#)
- ~BlisConGenerator
 - BlisConGenerator, [18](#)
- ~BlisConstraint
 - BlisConstraint, [24](#)
- ~BlisHeurRound
 - BlisHeurRound, [30](#)
- ~BlisHeuristic
 - BlisHeuristic, [27](#)
- ~BlisModel
 - BlisModel, [40](#)
- ~BlisNodeDesc
 - BlisNodeDesc, [56](#)
- ~BlisObjectInt
 - BlisObjectInt, [59](#)
- ~BlisSolution
 - BlisSolution, [69](#)
- ~BlisTreeNode
 - BlisTreeNode, [71](#)
- activeNode_
 - BlisModel, [52](#)
- addCalls
 - BlisConGenerator, [21](#)
 - BlisHeuristic, [28](#)
- addCutGenerator
 - BlisModel, [46](#)
- addHeuristic
 - BlisModel, [45](#)
- addNoConsCalls
 - BlisConGenerator, [21](#)
- addNoSolCalls
 - BlisHeuristic, [28](#)
- addNumConsGenerated
 - BlisConGenerator, [20](#)
- addNumConsUsed
 - BlisConGenerator, [21](#)
- addNumIterations
 - BlisModel, [48](#)
- addNumNodes
 - BlisModel, [48](#)
- addNumSolutions
 - BlisHeuristic, [28](#)
- addNumStrong
 - BlisModel, [41](#)
- addObjects
 - BlisModel, [45](#)
- addTime
 - BlisConGenerator, [21](#)
 - BlisHeuristic, [28](#)
- applyConstraints
 - BlisTreeNode, [72](#)
- atSolution
 - BlisConGenerator, [20](#)
- atSolution_
 - BlisConGenerator, [22](#)
- avelterations_
 - BlisModel, [53](#)
- betterBranchObject
 - BlisBranchStrategyBilevel, [10](#)
 - BlisBranchStrategyMaxInf, [11](#)
 - BlisBranchStrategyPseudo, [13](#)
 - BlisBranchStrategyRel, [14](#)
 - BlisBranchStrategyStrong, [16](#)
- BlisParams
 - branchStrategy, [63](#)
 - cutCliqueStrategy, [63](#)
 - cutFactor, [63](#)
 - cutGenerationFrequency, [63](#)
 - cutRampUp, [63](#)
 - cutStrategy, [63](#)
 - cutoff, [63](#)
 - cutoffInc, [63](#)
 - denseConFactor, [64](#)
 - difference, [63](#)
 - heurCallFrequency, [63](#)
 - heurStrategy, [63](#)
 - integerTol, [64](#)
 - lookAhead, [63](#)
 - objSense, [64](#)
 - optimalAbsGap, [64](#)
 - optimalRelGap, [64](#)
 - presolve, [63](#)
 - pseudoReliability, [63](#)
 - pseudoWeight, [64](#)
 - quickCutPass, [63](#)
 - scaleConFactor, [64](#)
 - shareConstraints, [63](#)

- sharePcostDepth, 63
- sharePcostFrequency, 63
- sharePseudocostRampUp, 63
- sharePseudocostSearch, 63
- shareVariables, 63
- strongCandSize, 63
- tailOff, 64
- BlisBranchObjectBilevel, 3
 - ~BlisBranchObjectBilevel, 4
 - BlisBranchObjectBilevel, 4
 - BlisBranchObjectBilevel, 4
 - branch, 5
 - branchingSet_, 5
 - clone, 4
 - decode, 5
 - decodeBlis, 5
 - encode, 5
 - encodeBlis, 5
 - operator=, 4
 - print, 5
- BlisBranchObjectInt, 5
 - ~BlisBranchObjectInt, 7
 - BlisBranchObjectInt, 7
 - BlisBranchObjectInt, 7
 - branch, 8
 - clone, 8
 - decode, 8
 - decodeBlis, 8
 - encode, 8
 - encodeBlis, 8
 - getDown, 8
 - getUp, 8
 - operator=, 7
 - print, 8
- BlisBranchStrategyBilevel, 9
 - ~BlisBranchStrategyBilevel, 9
 - betterBranchObject, 10
 - BlisBranchStrategyBilevel, 9
 - BlisBranchStrategyBilevel, 9
 - clone, 10
 - createCandBranchObjects, 10
- BlisBranchStrategyMaxInf, 10
 - ~BlisBranchStrategyMaxInf, 11
 - betterBranchObject, 11
 - BlisBranchStrategyMaxInf, 11
 - BlisBranchStrategyMaxInf, 11
 - clone, 11
 - createCandBranchObjects, 11
- BlisBranchStrategyPseudo, 11
 - ~BlisBranchStrategyPseudo, 12
 - betterBranchObject, 13
 - BlisBranchStrategyPseudo, 12
 - BlisBranchStrategyPseudo, 12
 - clone, 12
 - createCandBranchObjects, 13
 - setReliability, 12
- BlisBranchStrategyRel, 13
 - ~BlisBranchStrategyRel, 14
 - betterBranchObject, 14
 - BlisBranchStrategyRel, 14
 - BlisBranchStrategyRel, 14
 - clone, 14
 - createCandBranchObjects, 14
 - setReliability, 14
- BlisBranchStrategyStrong, 14
 - ~BlisBranchStrategyStrong, 15
 - betterBranchObject, 16
 - BlisBranchStrategyStrong, 15
 - BlisBranchStrategyStrong, 15
 - clone, 15
 - createCandBranchObjects, 15
- BlisConGenerator, 16
 - ~BlisConGenerator, 18
 - addCalls, 21
 - addNoConsCalls, 21
 - addNumConsGenerated, 20
 - addNumConsUsed, 21
 - addTime, 21
 - atSolution, 20
 - atSolution_, 22
 - BlisConGenerator, 18
 - BlisConGenerator, 18
 - calls, 21
 - calls_, 22
 - cutGenerationFreq, 19
 - cutGenerationFrequency_, 22
 - generateConstraints, 19
 - generator, 20
 - generator_, 21
 - getModel, 19
 - model_, 21
 - name, 19
 - name_, 22
 - noConsCalls, 21
 - noConsCalls_, 22
 - normal, 20
 - normal_, 22
 - numConsGenerated, 20
 - numConsGenerated_, 22
 - numConsUsed, 20
 - numConsUsed_, 22
 - operator=, 19
 - setAtSolution, 20
 - setCutGenerationFreq, 19
 - setName, 19
 - setNormal, 20
 - setStrategy, 19
 - setWhenInfeasible, 20

- strategy, 19
- strategy_, 21
- time, 21
- time_, 22
- whenInfeasible, 20
- whenInfeasible_, 22
- BlisConstraint, 23
 - ~BlisConstraint, 24
 - BlisConstraint, 24
 - BlisConstraint, 24
 - createOsiRowCut, 24
 - decode, 25
 - decodeBlis, 24
 - encode, 25
 - encodeBlis, 24
 - hashing, 24
 - violation, 25
- BlisHeurRound, 29
 - ~BlisHeurRound, 30
 - BlisHeurRound, 30
 - BlisHeurRound, 30
 - matrix_, 31
 - matrixByRow_, 31
 - seed_, 31
 - setModel, 31
- BlisHeuristic, 25
 - ~BlisHeuristic, 27
 - addCalls, 28
 - addNoSolCalls, 28
 - addNumSolutions, 28
 - addTime, 28
 - BlisHeuristic, 27
 - BlisHeuristic, 27
 - calls, 28
 - calls_, 29
 - clone, 27
 - name, 28
 - noSolCalls, 28
 - noSolsCalls_, 29
 - numSolutions, 28
 - numSolutions_, 29
 - setHeurCallFrequency, 27
 - setModel, 27
 - setStrategy, 27
 - strategy_, 28
 - time, 28
 - time_, 29
- BlisMessage, 31
- blisMessageHandler
 - BlisModel, 49
- blisMessageHandler_
 - BlisModel, 53
- blisMessages
 - BlisModel, 49
- blisMessages_
 - BlisModel, 53
- BlisModel, 31
 - ~BlisModel, 40
 - activeNode_, 52
 - addCutGenerator, 46
 - addHeuristic, 45
 - addNumIterations, 48
 - addNumNodes, 48
 - addNumStrong, 41
 - addObjects, 45
 - avelterations_, 53
 - blisMessageHandler, 49
 - blisMessageHandler_, 53
 - blisMessages, 49
 - blisMessages_, 53
 - BlisModel, 40
 - BlisPar, 49
 - BlisPar_, 53
 - BlisModel, 40
 - boundingPass_, 54
 - branchStrategy, 44
 - branchStrategy_, 51
 - checkInteger, 45
 - clearSharedObjectMark, 45
 - colMatrix_, 50
 - conLB, 42
 - conRandoms_, 53
 - constraintPool, 46
 - constraintPool_, 52
 - constraintPoolReceive, 46
 - constraintPoolReceive_, 54
 - constraintPoolSend, 46
 - constraintPoolSend_, 54
 - createIntegerObjects, 45
 - createObjects, 40
 - createRoot, 40
 - currAbsGap_, 54
 - currRelGap_, 54
 - cutGenerators, 46
 - cutStrategy_, 52
 - cutoff_, 51
 - cutoffInc_, 51
 - decodeBlis, 49
 - decodeToSelf, 50
 - delOldConstraints, 47
 - deleteObjects, 45
 - encode, 50
 - encodeBlis, 49
 - fathomAllNodes, 49
 - feasibleSolution, 43
 - feasibleSolutionHeur, 43
 - generators_, 52
 - getAvelterations, 48

getColLower, 42
getColUpper, 42
getConRandoms, 48
getCutGenerationFrequency, 47
getCutStrategy, 47
getCutoff, 43
getDenseConCutoff, 47
getIntColIndices, 45
getIntObjIndices, 45
getLpObjValue, 43
getLpSolution, 43
getMaxNumCons, 46
getNumBranchResolve, 41
getNumCols, 42
getNumHeurSolutions, 43
getNumIntObjects, 45
getNumIterations, 48
getNumNodes, 48
getNumOldConstraints, 46
getNumRows, 42
getNumSolutions, 43
getNumStrong, 41
getObjCoef, 42
getOldConstraintsSize, 47
getSolver, 41
gutsOfDestructor, 40
heurCallFrequency_, 55
heurStrategy_, 55
heuristics, 45
heuristics_, 52
importModel, 40
incObjValue_, 51
incumbent, 43
inputVar_, 51
integerTol_, 54
isRoot_, 54
leafToRootPath, 55
lpSolver_, 50
maxNumCons_, 52
modelLog, 48
newCutPool_, 55
nodeLog, 49
numBranchResolve_, 52
numCutGenerators, 46
numCutGenerators_, 52
numHeuristics, 45
numHeuristics_, 52
numIntObjects_, 51
numIterations_, 53
numNodes_, 53
numObjects, 44
numObjects_, 51
numOldConstraints_, 53
numStrong_, 52
objSense_, 50
objects, 44
objects_, 51
oldConstraints, 47
oldConstraints_, 53
oldConstraintsSize_, 53
optimalAbsGap_, 54
optimalRelGap_, 54
origLpSolver_, 50
packSharedConstraints, 49
packSharedKnowledge, 50
packSharedPseudocost, 49
packSharedVariables, 49
passInPriorities, 48
postprocess, 41
preprocess, 41
presolvedLpSolver_, 50
priority, 48
priority_, 51
readInstance, 40
readParameters, 40
registerKnowledge, 49
resolve, 41
setActiveNode, 41
setBranchingMethod, 44
setCutStrategy, 47
setCutoff, 43
setDenseConCutoff, 48
setMaxNumCons, 46
setNumBranchResolve, 42
setNumObjects, 44
setNumOldConstraints, 47
setOldConstraints, 47
setOldConstraintsSize, 47
setSharedObjectMark, 44
setSolEstimate, 41
setSolver, 41
setupSelf, 40
sharedObjectMark_, 51
solver, 41
startConLB, 42
startVarLB, 42
storeSolution, 43
tempVarLBPos, 43
tempVarLBPos_, 54
unpackSharedConstraints, 49
unpackSharedKnowledge, 50
unpackSharedVariables, 49
userFeasibleSolution, 44
varLB, 42
varLB_, 50
writeParameters, 40
BlisNodeDesc, 55
~BlisNodeDesc, 56

- BlisNodeDesc, 56
- BlisNodeDesc, 56
- decode, 57
- decodeBlis, 57
- encode, 57
- encodeBlis, 57
- getBasis, 56
- getBranchedDir, 57
- getBranchedInd, 57
- getBranchedVal, 57
- setBasis, 56
- setBranchedDir, 56
- setBranchedInd, 57
- setBranchedVal, 57
- BlisObjectInt, 58
 - ~BlisObjectInt, 59
 - BlisObjectInt, 59
 - BlisObjectInt, 59
 - breakEven, 60
 - breakEven_, 61
 - clone, 59
 - columnIndex, 60
 - columnIndex_, 61
 - createBranchObject, 60
 - feasibleRegion, 60
 - infeasibility, 59
 - notPreferredNewFeasible, 60
 - operator=, 59
 - originalLower_, 61
 - originalUpper_, 61
 - preferredNewFeasible, 60
 - pseudocost, 60
 - pseudocost_, 61
 - resetBounds, 60
 - setBreakEven, 60
- BlisPar
 - BlisModel, 49
- BlisPar_
 - BlisModel, 53
- BlisParams, 61
 - BlisParams, 64
 - BlisParams, 64
 - chrParams, 63
 - createKeywordList, 64
 - dblParams, 63
 - intParams, 63
 - pack, 64
 - setDefaultEntries, 64
 - strArrayParams, 64
 - strParams, 64
 - unpack, 64
- BlisPresolve, 65
- BlisPseudocost, 65
 - BlisPseudocost, 66
 - BlisPseudocost, 66
 - decode, 68
 - decodeFrom, 67
 - encode, 68
 - encodeTo, 67
 - getDownCost, 67
 - getDownCount, 67
 - getScore, 67
 - getUpCost, 67
 - getUpCount, 67
 - setScore, 67
 - setWeight, 67
 - update, 67
- BlisSolution, 68
 - ~BlisSolution, 69
 - BlisSolution, 68
 - BlisSolution, 68
 - decode, 69
 - encode, 69
 - print, 69
- BlisStrong, 69
- BlisTreeNode, 69
 - ~BlisTreeNode, 71
 - applyConstraints, 72
 - BlisTreeNode, 71
 - BlisTreeNode, 71
 - branch, 71
 - callHeuristics, 72
 - chooseBranchingObject, 71
 - createNewTreeNode, 71
 - decode, 72
 - encode, 72
 - generateConstraints, 71
 - getViolatedConstraints, 72
 - init, 71
 - process, 71
 - reducedCostFix, 72
 - selectBranchObject, 71
- BlisVariable, 72
 - decode, 73
 - decodeBlis, 73
 - encode, 73
 - encodeBlis, 73
- boundingPass_
 - BlisModel, 54
- branch
 - BlisBranchObjectBilevel, 5
 - BlisBranchObjectInt, 8
 - BlisTreeNode, 71
- branchStrategy
 - BlisParams, 63
- branchStrategy
 - BlisModel, 44
- branchStrategy_

- BlisModel, 51
- branchingSet_
 - BlisBranchObjectBilevel, 5
- breakEven
 - BlisObjectInt, 60
- breakEven_
 - BlisObjectInt, 61
- callHeuristics
 - BlisTreeNode, 72
- calls
 - BlisConGenerator, 21
 - BlisHeuristic, 28
- calls_
 - BlisConGenerator, 22
 - BlisHeuristic, 29
- checkInteger
 - BlisModel, 45
- chooseBranchingObject
 - BlisTreeNode, 71
- chrParams
 - BlisParams, 63
- clearSharedObjectMark
 - BlisModel, 45
- clone
 - BlisBranchObjectBilevel, 4
 - BlisBranchObjectInt, 8
 - BlisBranchStrategyBilevel, 10
 - BlisBranchStrategyMaxInf, 11
 - BlisBranchStrategyPseudo, 12
 - BlisBranchStrategyRel, 14
 - BlisBranchStrategyStrong, 15
 - BlisHeuristic, 27
 - BlisObjectInt, 59
- colMatrix_
 - BlisModel, 50
- columnIndex
 - BlisObjectInt, 60
- columnIndex_
 - BlisObjectInt, 61
- conLB
 - BlisModel, 42
- conRandoms_
 - BlisModel, 53
- constraintPool
 - BlisModel, 46
- constraintPool_
 - BlisModel, 52
- constraintPoolReceive
 - BlisModel, 46
- constraintPoolReceive_
 - BlisModel, 54
- constraintPoolSend
 - BlisModel, 46
- constraintPoolSend_
 - BlisModel, 54
- createBranchObject
 - BlisObjectInt, 60
- createCandBranchObjects
 - BlisBranchStrategyBilevel, 10
 - BlisBranchStrategyMaxInf, 11
 - BlisBranchStrategyPseudo, 13
 - BlisBranchStrategyRel, 14
 - BlisBranchStrategyStrong, 15
- createIntgerObjects
 - BlisModel, 45
- createKeywordList
 - BlisParams, 64
- createNewTreeNode
 - BlisTreeNode, 71
- createObjects
 - BlisModel, 40
- createOsiRowCut
 - BlisConstraint, 24
- createRoot
 - BlisModel, 40
- currAbsGap_
 - BlisModel, 54
- currRelGap_
 - BlisModel, 54
- cutCliqueStrategy
 - BlisParams, 63
- cutFactor
 - BlisParams, 63
- cutGenerationFrequency
 - BlisParams, 63
- cutRampUp
 - BlisParams, 63
- cutStrategy
 - BlisParams, 63
- cutGenerationFreq
 - BlisConGenerator, 19
- cutGenerationFrequency_
 - BlisConGenerator, 22
- cutGenerators
 - BlisModel, 46
- cutStrategy_
 - BlisModel, 52
- cutoff
 - BlisParams, 63
- cutoffInc
 - BlisParams, 63
- cutoff_
 - BlisModel, 51
- cutoffInc_
 - BlisModel, 51
- dblParams

- BlisParams, 63
- decode
 - BlisBranchObjectBilevel, 5
 - BlisBranchObjectInt, 8
 - BlisConstraint, 25
 - BlisNodeDesc, 57
 - BlisPseudocost, 68
 - BlisSolution, 69
 - BlisTreeNode, 72
 - BlisVariable, 73
- decodeBlis
 - BlisBranchObjectBilevel, 5
 - BlisBranchObjectInt, 8
 - BlisConstraint, 24
 - BlisModel, 49
 - BlisNodeDesc, 57
 - BlisVariable, 73
- decodeFrom
 - BlisPseudocost, 67
- decodeToSelf
 - BlisModel, 50
- delOldConstraints
 - BlisModel, 47
- deleteObjects
 - BlisModel, 45
- denseConFactor
 - BlisParams, 64
- difference
 - BlisParams, 63
- encode
 - BlisBranchObjectBilevel, 5
 - BlisBranchObjectInt, 8
 - BlisConstraint, 25
 - BlisModel, 50
 - BlisNodeDesc, 57
 - BlisPseudocost, 68
 - BlisSolution, 69
 - BlisTreeNode, 72
 - BlisVariable, 73
- encodeBlis
 - BlisBranchObjectBilevel, 5
 - BlisBranchObjectInt, 8
 - BlisConstraint, 24
 - BlisModel, 49
 - BlisNodeDesc, 57
 - BlisVariable, 73
- encodeTo
 - BlisPseudocost, 67
- fathomAllNodes
 - BlisModel, 49
- feasibleRegion
 - BlisObjectInt, 60
- feasibleSolution
 - BlisModel, 43
- feasibleSolutionHeur
 - BlisModel, 43
- generateConstraints
 - BlisConGenerator, 19
 - BlisTreeNode, 71
- generator
 - BlisConGenerator, 20
- generator_
 - BlisConGenerator, 21
- generators_
 - BlisModel, 52
- getAvelterations
 - BlisModel, 48
- getBasis
 - BlisNodeDesc, 56
- getBranchedDir
 - BlisNodeDesc, 57
- getBranchedInd
 - BlisNodeDesc, 57
- getBranchedVal
 - BlisNodeDesc, 57
- getColLower
 - BlisModel, 42
- getColUpper
 - BlisModel, 42
- getConRandoms
 - BlisModel, 48
- getCutGenerationFrequency
 - BlisModel, 47
- getCutStrategy
 - BlisModel, 47
- getCutoff
 - BlisModel, 43
- getDenseConCutoff
 - BlisModel, 47
- getDown
 - BlisBranchObjectInt, 8
- getDownCost
 - BlisPseudocost, 67
- getDownCount
 - BlisPseudocost, 67
- getIntColIndices
 - BlisModel, 45
- getIntObjIndices
 - BlisModel, 45
- getLpObjValue
 - BlisModel, 43
- getLpSolution
 - BlisModel, 43
- getMaxNumCons
 - BlisModel, 46
- getModel

- BlisConGenerator, 19
- getNumBranchResolve
 - BlisModel, 41
- getNumCols
 - BlisModel, 42
- getNumHeurSolutions
 - BlisModel, 43
- getNumIntObjects
 - BlisModel, 45
- getNumIterations
 - BlisModel, 48
- getNumNodes
 - BlisModel, 48
- getNumOldConstraints
 - BlisModel, 46
- getNumRows
 - BlisModel, 42
- getNumSolutions
 - BlisModel, 43
- getNumStrong
 - BlisModel, 41
- getObjCoef
 - BlisModel, 42
- getOldConstraintsSize
 - BlisModel, 47
- getScore
 - BlisPseudocost, 67
- getSolver
 - BlisModel, 41
- getUp
 - BlisBranchObjectInt, 8
- getUpCost
 - BlisPseudocost, 67
- getUpCount
 - BlisPseudocost, 67
- getViolatedConstraints
 - BlisTreeNode, 72
- gutsOfDestructor
 - BlisModel, 40
- hashing
 - BlisConstraint, 24
- heurCallFrequency
 - BlisParams, 63
- heurStrategy
 - BlisParams, 63
- heurCallFrequency_
 - BlisModel, 55
- heurStrategy_
 - BlisModel, 55
- heuristics
 - BlisModel, 45
- heuristics_
 - BlisModel, 52
- importModel
 - BlisModel, 40
- incObjValue_
 - BlisModel, 51
- incumbent
 - BlisModel, 43
- infeasibility
 - BlisObjectInt, 59
- init
 - BlisTreeNode, 71
- inputVar_
 - BlisModel, 51
- intParams
 - BlisParams, 63
- integerTol
 - BlisParams, 64
- integerTol_
 - BlisModel, 54
- isRoot_
 - BlisModel, 54
- leafToRootPath
 - BlisModel, 55
- lookAhead
 - BlisParams, 63
- lpSolver_
 - BlisModel, 50
- matrix_
 - BlisHeurRound, 31
- matrixByRow_
 - BlisHeurRound, 31
- maxNumCons_
 - BlisModel, 52
- model_
 - BlisConGenerator, 21
- modelLog
 - BlisModel, 48
- name
 - BlisConGenerator, 19
 - BlisHeuristic, 28
- name_
 - BlisConGenerator, 22
- newCutPool_
 - BlisModel, 55
- noConsCalls
 - BlisConGenerator, 21
- noConsCalls_
 - BlisConGenerator, 22
- noSolCalls
 - BlisHeuristic, 28
- noSolsCalls_
 - BlisHeuristic, 29
- nodeLog

- BlisModel, [49](#)
- normal
 - BlisConGenerator, [20](#)
- normal_
 - BlisConGenerator, [22](#)
- notPreferredNewFeasible
 - BlisObjectInt, [60](#)
- numBranchResolve_
 - BlisModel, [52](#)
- numConsGenerated
 - BlisConGenerator, [20](#)
- numConsGenerated_
 - BlisConGenerator, [22](#)
- numConsUsed
 - BlisConGenerator, [20](#)
- numConsUsed_
 - BlisConGenerator, [22](#)
- numCutGenerators
 - BlisModel, [46](#)
- numCutGenerators_
 - BlisModel, [52](#)
- numHeuristics
 - BlisModel, [45](#)
- numHeuristics_
 - BlisModel, [52](#)
- numIntObjects_
 - BlisModel, [51](#)
- numIterations_
 - BlisModel, [53](#)
- numNodes_
 - BlisModel, [53](#)
- numObjects
 - BlisModel, [44](#)
- numObjects_
 - BlisModel, [51](#)
- numOldConstraints_
 - BlisModel, [53](#)
- numSolutions
 - BlisHeuristic, [28](#)
- numSolutions_
 - BlisHeuristic, [29](#)
- numStrong_
 - BlisModel, [52](#)
- objSense
 - BlisParams, [64](#)
- objSense_
 - BlisModel, [50](#)
- objects
 - BlisModel, [44](#)
- objects_
 - BlisModel, [51](#)
- oldConstraints
 - BlisModel, [47](#)
- oldConstraints_
 - BlisModel, [53](#)
- oldConstraintsSize_
 - BlisModel, [53](#)
- operator=
 - BlisBranchObjectBilevel, [4](#)
 - BlisBranchObjectInt, [7](#)
 - BlisConGenerator, [19](#)
 - BlisObjectInt, [59](#)
- optimalAbsGap
 - BlisParams, [64](#)
- optimalRelGap
 - BlisParams, [64](#)
- optimalAbsGap_
 - BlisModel, [54](#)
- optimalRelGap_
 - BlisModel, [54](#)
- origLpSolver_
 - BlisModel, [50](#)
- originalLower_
 - BlisObjectInt, [61](#)
- originalUpper_
 - BlisObjectInt, [61](#)
- pack
 - BlisParams, [64](#)
- packSharedConstraints
 - BlisModel, [49](#)
- packSharedKnowledge
 - BlisModel, [50](#)
- packSharedPseudocost
 - BlisModel, [49](#)
- packSharedVariables
 - BlisModel, [49](#)
- passInPriorities
 - BlisModel, [48](#)
- postprocess
 - BlisModel, [41](#)
- preferredNewFeasible
 - BlisObjectInt, [60](#)
- preprocess
 - BlisModel, [41](#)
- presolve
 - BlisParams, [63](#)
- presolvedLpSolver_
 - BlisModel, [50](#)
- print
 - BlisBranchObjectBilevel, [5](#)
 - BlisBranchObjectInt, [8](#)
 - BlisSolution, [69](#)
- priority
 - BlisModel, [48](#)
- priority_
 - BlisModel, [51](#)

- process
 - BlisTreeNode, 71
- pseudoReliability
 - BlisParams, 63
- pseudoWeight
 - BlisParams, 64
- pseudocost
 - BlisObjectInt, 60
- pseudocost_
 - BlisObjectInt, 61
- quickCutPass
 - BlisParams, 63
- readInstance
 - BlisModel, 40
- readParameters
 - BlisModel, 40
- reducedCostFix
 - BlisTreeNode, 72
- registerKnowledge
 - BlisModel, 49
- resetBounds
 - BlisObjectInt, 60
- resolve
 - BlisModel, 41
- scaleConFactor
 - BlisParams, 64
- seed_
 - BlisHeurRound, 31
- selectBranchObject
 - BlisTreeNode, 71
- setActiveNode
 - BlisModel, 41
- setAtSolution
 - BlisConGenerator, 20
- setBasis
 - BlisNodeDesc, 56
- setBranchedDir
 - BlisNodeDesc, 56
- setBranchedInd
 - BlisNodeDesc, 57
- setBranchedVal
 - BlisNodeDesc, 57
- setBranchingMethod
 - BlisModel, 44
- setBreakEven
 - BlisObjectInt, 60
- setCutGenerationFreq
 - BlisConGenerator, 19
- setCutStrategy
 - BlisModel, 47
- setCutoff
 - BlisModel, 43
- setDefaultEntries
 - BlisParams, 64
- setDenseConCutoff
 - BlisModel, 48
- setHeurCallFrequency
 - BlisHeuristic, 27
- setMaxNumCons
 - BlisModel, 46
- setModel
 - BlisHeuristic, 27
 - BlisHeurRound, 31
- setName
 - BlisConGenerator, 19
- setNormal
 - BlisConGenerator, 20
- setNumBranchResolve
 - BlisModel, 42
- setNumObjects
 - BlisModel, 44
- setNumOldConstraints
 - BlisModel, 47
- setOldConstraints
 - BlisModel, 47
- setOldConstraintsSize
 - BlisModel, 47
- setReliability
 - BlisBranchStrategyPseudo, 12
 - BlisBranchStrategyRel, 14
- setScore
 - BlisPseudocost, 67
- setSharedObjectMark
 - BlisModel, 44
- setSolEstimate
 - BlisModel, 41
- setSolver
 - BlisModel, 41
- setStrategy
 - BlisConGenerator, 19
 - BlisHeuristic, 27
- setWeight
 - BlisPseudocost, 67
- setWhenInfeasible
 - BlisConGenerator, 20
- setUpSelf
 - BlisModel, 40
- shareConstraints
 - BlisParams, 63
- sharePcostDepth
 - BlisParams, 63
- sharePcostFrequency
 - BlisParams, 63
- sharePseudocostRampUp
 - BlisParams, 63
- sharePseudocostSearch

- BlisParams, 63
- shareVariables
 - BlisParams, 63
- sharedObjectMark_
 - BlisModel, 51
- solver
 - BlisModel, 41
- startConLB
 - BlisModel, 42
- startVarLB
 - BlisModel, 42
- storeSolution
 - BlisModel, 43
- strArrayParams
 - BlisParams, 64
- strParams
 - BlisParams, 64
- strategy
 - BlisConGenerator, 19
- strategy_
 - BlisConGenerator, 21
 - BlisHeuristic, 28
- strongCandSize
 - BlisParams, 63
- tailOff
 - BlisParams, 64
- tempVarLBPos
 - BlisModel, 43
- tempVarLBPos_
 - BlisModel, 54
- time
 - BlisConGenerator, 21
 - BlisHeuristic, 28
- time_
 - BlisConGenerator, 22
 - BlisHeuristic, 29
- unpack
 - BlisParams, 64
- unpackSharedConstraints
 - BlisModel, 49
- unpackSharedKnowledge
 - BlisModel, 50
- unpackSharedVariables
 - BlisModel, 49
- update
 - BlisPseudocost, 67
- userFeasibleSolution
 - BlisModel, 44
- varLB
 - BlisModel, 42
- varLB_
 - BlisModel, 50
- violation
 - BlisConstraint, 25
- whenInfeasible
 - BlisConGenerator, 20
- whenInfeasible_
 - BlisConGenerator, 22
- writeParameters
 - BlisModel, 40