Smi

0.95

Generated by Doxygen 1.8.9.1

Thu Oct 8 2015 22:44:46

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

- _EKKfactinfo [external]
- AbcDualRowPivot [external]
  - AbcDualRowDantzig [external]
  - AbcDualRowSteepest [external]
- AbcMatrix [external]
- AbcMatrix2 [external]
- AbcMatrix3 [external]
- AbcNonLinearCost [external]
- AbcPrimalColumnPivot [external]
  - AbcPrimalColumnDantzig [external]
  - AbcPrimalColumnSteepest [external]
- AbcSimplexFactorization [external]
- AbcTolerancesEtc [external]
- AbcWarmStartOrganizer [external]
- forcing_constraint_action::action [external]
- doubleton_action::action [external]
- tripleton_action::action [external]
- remove_fixed_action::action [external]
- std::allocator< T >
- ampl_info [external]
- OsiSolverInterface::ApplyCutsReturnCode [external]
- std::array< T >
- std::auto_ptr< T >
- auxiliary_graph [external]
- std::basic_string< Char >
  - std::string
  - std::wstring
- std::basic_string< char >
- std::basic_string< wchar_t >
- std::bitset< Bits >
- BitVector128 [external]
- blockStruct [external]
- blockStruct3 [external]
- ClpNode::branchState [external]

- CbcOrClpParam [external]
- Cgl012Cut [external]
- cgl_arc [external]
- cgl_graph [external]
- cgl_node [external]
- CglBK [external]
- CglCutGenerator [external]
    - CglAllDifferent [external]
    - CglClique [external]
        - CglFakeClique [external]
    - CglDuplicateRow [external]
    - CglFlowCover [external]
    - CglGMI [external]
    - CglGomory [external]
    - CglImplication [external]
    - CglKnapsackCover [external]
    - CglLandP [external]
    - CglLiftAndProject [external]
    - CglMixedIntegerRounding [external]
    - CglMixedIntegerRounding2 [external]
    - CglOddHole [external]
    - CglProbing [external]
    - CglRedSplit [external]
    - CglRedSplit2 [external]
    - CglResidualCapacity [external]
    - CglSimpleRounding [external]
    - CglStored [external]
    - CglTwomir [external]
    - CglZeroHalf [external]
- CglFlowVUB [external]
- CglHashLink [external]
- LAP::CglLandPSimplex [external]
- CglMixIntRoundVUB [external]
- CglMixIntRoundVUB2 [external]
- CglParam [external]
    - CglGMIParam [external]
    - CglLandP::Parameters [external]
    - CglRedSplit2Param [external]
    - CglRedSplitParam [external]
- CglPreProcess [external]
- CglTreeInfo [external]
    - CglTreeProbingInfo [external]
- CglUniqueRowCuts [external]
- CliqueEntry [external]
- CglProbing::CliqueType [external]
- ClpCholeskyBase [external]
    - ClpCholeskyDense [external]
    - ClpCholeskyMumps [external]
    - ClpCholeskyTaucs [external]
    - ClpCholeskyUfl [external]
    - ClpCholeskyWssmp [external]
    - ClpCholeskyWssmpKKT [external]
- ClpCholeskyDenseC [external]
- ClpConstraint [external]

- ClpConstraintLinear [external]
- ClpConstraintQuadratic [external]
- ClpDataSave [external]
- ClpDisasterHandler [external]
  - OsiClpDisasterHandler [external]
- ClpDualRowPivot [external]
  - ClpDualRowDantzig [external]
  - ClpDualRowSteepest [external]
- ClpEventHandler [external]
  - MyEventHandler [external]
- ClpFactorization [external]
- ClpHashValue [external]
- ClpLsqr [external]
- ClpMatrixBase [external]
  - ClpDummyMatrix [external]
  - ClpNetworkMatrix [external]
  - ClpPackedMatrix [external]
    - ClpDynamicMatrix [external]
      - ClpDynamicExampleMatrix [external]
    - ClpGubMatrix [external]
      - ClpGubDynamicMatrix [external]
  - ClpPlusMinusOneMatrix [external]
- ClpModel [external]
  - ClpInterior [external]
    - ClpPdco [external]
    - ClpPredictorCorrector [external]
  - ClpSimplex [external]
    - AbcSimplex [external]
      - AbcSimplexDual [external]
      - AbcSimplexPrimal [external]
    - ClpSimplexDual [external]
    - ClpSimplexOther [external]
    - ClpSimplexPrimal [external]
      - ClpSimplexNonlinear [external]
- ClpNetworkBasis [external]
- ClpNode [external]
- ClpNodeStuff [external]
- ClpNonLinearCost [external]
- ClpObjective [external]
  - ClpLinearObjective [external]
  - ClpQuadraticObjective [external]
- ClpPackedMatrix2 [external]
- ClpPackedMatrix3 [external]
- ClpPdcoBase [external]
- ClpPresolve [external]
- ClpPrimalColumnPivot [external]
  - ClpPrimalColumnDantzig [external]
  - ClpPrimalColumnSteepest [external]
  - ClpPrimalQuadraticDantzig [external]
- ClpSimplexProgress [external]
- ClpSolve [external]
- ClpTrustedData [external]
- CoinAbcAnyFactorization [external]
  - CoinAbcDenseFactorization [external]

- CoinSearchTreeComparePreferred [external]
- CoinSearchTreeManager [external]
- CoinSet [external]
  - CoinSosSet [external]
- CoinSnapshot [external]
- CoinThreadRandom [external]
- CoinTimer [external]
- CoinTreeNode [external]
- CoinTreeSiblings [external]
- CoinTriple< S, T, U > [external]
- CoinWarmStart [external]
  - CoinWarmStartBasis [external]
    - AbcWarmStart [external]
  - CoinWarmStartDual [external]
  - CoinWarmStartPrimalDual [external]
  - CoinWarmStartVector< T > [external]
  - CoinWarmStartVector< double > [external]
  - CoinWarmStartVector< U > [external]
  - CoinWarmStartVectorPair< T, U > [external]
- CoinWarmStartDiff [external]
  - CoinWarmStartBasisDiff [external]
  - CoinWarmStartDualDiff [external]
  - CoinWarmStartPrimalDualDiff [external]
  - CoinWarmStartVectorDiff< T > [external]
  - CoinWarmStartVectorDiff< double > [external]
  - CoinWarmStartVectorDiff< U > [external]
  - CoinWarmStartVectorPairDiff< T, U > [external]
- CoinYacc [external]
- std::complex
- OsiCuts::const_iterator [external]
- std::multiset< K >::const_iterator
- std::unordered_set< K >::const_iterator
- std::map< K, T >::const_iterator
- std::basic_string< Char >::const_iterator
- std::string::const_iterator
- std::multimap< K, T >::const_iterator
- std::wstring::const_iterator
- std::deque< T >::const_iterator
- std::list< T >::const_iterator
- std::unordered_map< K, T >::const_iterator
- std::forward_list< T >::const_iterator
- std::unordered_multimap< K, T >::const_iterator
- std::set< K >::const_iterator
- std::unordered_multiset< K >::const_iterator
- std::vector< T >::const_iterator
- std::vector< T >::const_reverse_iterator
- std::unordered_multiset< K >::const_reverse_iterator
- std::set< K >::const_reverse_iterator
- std::unordered_multimap< K, T >::const_reverse_iterator
- std::basic_string< Char >::const_reverse_iterator
- std::wstring::const_reverse_iterator
- std::deque< T >::const_reverse_iterator
- std::map< K, T >::const_reverse_iterator
- std::forward_list< T >::const_reverse_iterator

- std::unordered_map< K, T >::const_reverse_iterator
- std::multimap< K, T >::const_reverse_iterator
- std::list< T >::const_reverse_iterator
- std::unordered_set< K >::const_reverse_iterator
- std::multiset< K >::const_reverse_iterator
- std::string::const_reverse_iterator
- cut [external]
- cut_list [external]
- cutParams [external]
- LAP::Cuts [external]
- cycle [external]
- cycle_list [external]
- std::deque< T >
- std::deque< StdVectorDouble >
- DGG_constraint_t [external]
- DGG_data_t [external]
- DGG_list_t [external]
- disaggregationAction [external]
- dropped_zero [external]
- dualColumnResult [external]
- edge [external]
- EKKHlink [external]
- std::error_category
- std::error_code
- std::error_condition
- std::exception
  - std::bad_alloc
  - std::bad_cast
  - std::bad_exception
  - std::bad_typeid
  - std::ios_base::failure
  - std::logic_error
    - std::domain_error
    - std::invalid_argument
    - std::length_error
    - std::out_of_range
  - std::runtime_error
    - std::overflow_error
    - std::range_error
    - std::underflow_error
- FactorPointers [external]
- std::forward_list< T >
- glp_prob [external]
- Idiot [external]
- IdiotResult [external]
- ilp [external]
- Info [external]
- info_weak [external]
- std::ios_base
  - basic_ios< char >
  - basic_ios< wchar_t >
  - std::basic_ios
    - basic_istream< char >
    - basic_istream< wchar_t >

    - basic_ostream< char >
    - basic_ostream< wchar_t >
    - std::basic_istream
      - basic_ifstream< char >
      - basic_ifstream< wchar_t >
      - basic_iostream< char >
      - basic_iostream< wchar_t >
      - basic_istringstream< char >
      - basic_istringstream< wchar_t >
      - std::basic_ifstream
        - std::ifstream
        - std::wifstream
      - std::basic_iostream
        - basic_fstream< char >
        - basic_fstream< wchar_t >
        - basic_stringstream< char >
        - basic_stringstream< wchar_t >
        - std::basic_fstream
          - std::fstream
          - std::wfstream
        - std::basic_stringstream
          - std::stringstream
          - std::wstringstream
      - std::basic_istringstream
        - std::istringstream
        - std::wistringstream
      - std::istream
      - std::wistream
    - std::basic_ostream
      - basic_iostream< char >
      - basic_iostream< wchar_t >
      - basic_ofstream< char >
      - basic_ofstream< wchar_t >
      - basic_ostringstream< char >
      - basic_ostringstream< wchar_t >
      - std::basic_iostream
      - std::basic_ofstream
        - std::ofstream
        - std::wofstream
      - std::basic_ostringstream
        - std::ostringstream
        - std::wostringstream
      - std::ostream
      - std::wostream
    - std::ios
    - std::wios
- OsiCuts::iterator `[external]`
- std::multiset< K >::iterator
- std::set< K >::iterator
- std::vector< T >::iterator
- std::unordered_multiset< K >::iterator
- std::unordered_multimap< K, T >::iterator
- std::unordered_set< K >::iterator
- std::string::iterator

- std::deque< T >::iterator
- std::map< K, T >::iterator
- std::forward_list< T >::iterator
- std::unordered_map< K, T >::iterator
- std::list< T >::iterator
- std::wstring::iterator
- std::multimap< K, T >::iterator
- std::basic_string< Char >::iterator
- std::list< T >
- log_var [external]
- std::map< K, T >
- std::map< int, CoinPackedVector * >
- std::map< int, double * >
- std::map< int, SmiTreeNode< SmiScnNode * > * >
- std::map< int, SmiTreeNode< T > * >
- std::map< string, int >
- std::multimap< K, T >
- std::multiset< K >
- Options [external]
- OsiAuxInfo [external]
  - OsiBabSolver [external]
- OsiBranchingInformation [external]
- OsiBranchingObject [external]
  - OsiTwoWayBranchingObject [external]
    - OsiIntegerBranchingObject [external]
    - OsiLotsizeBranchingObject [external]
    - OsiSOSBranchingObject [external]
- OsiChooseVariable [external]
  - OsiChooseStrong [external]
- OsiCut [external]
  - OsiColCut [external]
  - OsiRowCut [external]
    - OsiRowCut2 [external]
- OsiCuts [external]
- OsiHotInfo [external]
- OsiObject [external]
  - OsiObject2 [external]
    - OsiLotsize [external]
    - OsiSimpleInteger [external]
    - OsiSOS [external]
- OsiPresolve [external]
- OsiPseudoCosts [external]
- OsiRowCutDebugger [external]
- OsiSolverBranch [external]
- OsiSolverInterface [external]
  - OsiClpSolverInterface [external]
  - OsiCpxSolverInterface [external]
  - OsiGlpkSolverInterface [external]
  - OsiGrbSolverInterface [external]
  - OsiMskSolverInterface [external]
  - OsiSpxSolverInterface [external]
  - OsiXprSolverInterface [external]
- OsiSolverResult [external]
- Outfo [external]

OsiUnitTest::TestOutcome [external]
OsiUnitTest::TestOutcomes [external]
std::thread
std::unique_ptr< T >
std::unordered_map< K, T >
std::unordered_multimap< K, T >
std::unordered_multiset< K >
std::unordered_set< K >
std::valarray< T >
LAP::Validator [external]
std::vector< T >
std::vector< ColumnSelectionStrategy >
std::vector< double * >
std::vector< double >
std::vector< int >
std::vector< RowSelectionStrategy >
std::vector< SmiDiscreteEvent * >
std::vector< SmiDiscreteRV * >
std::vector< SmiNodeData * >
std::vector< SmiScnNode * >
std::vector< SmiTreeNode< SmiScnNode * > * >
std::vector< SmiTreeNode< T > * >
std::vector< std::string >
std::vector< std::vector< int > >
std::weak_ptr< T >
K
S
T
U

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 SmiCoreCombineAdd Class Reference

Inheritance diagram for SmiCoreCombineAdd:

4.2 SmiCoreCombineReplace Class Reference

Inheritance diagram for SmiCoreCombineReplace:

Collaboration diagram for SmiCoreCombineReplace:

Public Member Functions

- virtual void Process (double ∗d1, int o1, const **CoinPackedVector** &cpv2, char ∗type=0)
  Process.

4.2.1 Detailed Description

Definition at line 54 of file SmiCoreCombineRule.hpp.

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

- SmiCoreCombineRule.hpp

4.3 SmiCoreCombineRule Class Reference

This deals with combining Core and Stochastic data.

`#include <SmiCoreCombineRule.hpp>`

Inheritance diagram for SmiCoreCombineRule:

Public Member Functions

Virtual Functions: Process and Diff

- virtual void Process (double ∗d1, int o1, const **CoinPackedVector** &cpv2, char ∗type=0)=0

  Process.
- virtual void **Process** (double ∗d1, int o1, const int len, const int ∗inds, const double ∗dels, char ∗type=0)=0
- virtual **CoinPackedVector** ∗ **Process** (**CoinPackedVector** ∗cpv1, **CoinPackedVector** ∗cpv2, char ∗type=0)=0
- virtual int **Process** (double ∗dr, const int dr_len, **CoinPackedVector** ∗cpv, double ∗dels, int ∗indx)=0
- virtual int **Process** (double ∗dr, const int dr_len, const int cpv_nels, const int ∗cpv_ind, const double ∗cpv_els, double ∗dels, int ∗indx)=0
- virtual ∼**SmiCoreCombineRule** ()

4.3.1 Detailed Description

This deals with combining Core and Stochastic data.

In the Stochastic MPS standard, stochastic data updates the underlying core lp data. To specify a new scenario, one only has to identify those data that are different. So, in a sense, the stochastic data is really a "diff" between the scenario and the core data. This class specifies how to perform the "undiff", that is, how to combine core and stochastic data.

And of course, a complete implementation specifies the "diff" part as well. Now during a fit of original confusion in the birth of the SMPS standard, we decided to make default combine rule "replace", which has a rather special "diff", but we've learned to live with it.

There only needs to be one of these classes. so they're singletons.

Definition at line 36 of file SmiCoreCombineRule.hpp.

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

- SmiCoreCombineRule.hpp

4.4 SmiCoreData Class Reference

Public Member Functions

- void addQuadraticObjectiveToCore (int ∗starts, int ∗indx, double ∗dels)

  Adds QP data after the constructor has been called.

4.4.1 Detailed Description

Definition at line 202 of file SmiScnData.hpp.

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

- SmiScnData.hpp

4.5 SmiDiscreteDistribution Class Reference

Public Member Functions

- void addDiscreteRV (SmiDiscreteRV ∗s)

  add discrete RV

- SmiDiscreteRV ∗ getDiscreteRV (int i)

  get discrete RV
- int getNumRV ()

  get number of RV
- SmiCoreData ∗ getCore ()

  get core model
- void setCombineWithCoreRule (SmiCoreCombineRule ∗r)

  set combine rule
- SmiCoreCombineRule ∗ getCombineWithCoreRule ()

  get combine rule
- SmiDiscreteDistribution (SmiCoreData ∗c, SmiCoreCombineRule ∗r=SmiCoreCombineReplace::Instance())

  constructor requires core data and combine rule

4.5.1 Detailed Description

Definition at line 24 of file SmiDiscreteDistribution.hpp.

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

- SmiDiscreteDistribution.hpp

4.6 SmiDiscreteEvent Class Reference

Inheritance diagram for SmiDiscreteEvent:

Collaboration diagram for SmiDiscreteEvent:

4.6.1 Detailed Description

Definition at line 69 of file SmiDiscreteDistribution.hpp.

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

- SmiDiscreteDistribution.hpp

4.7 SmiDiscreteRV Class Reference

4.7.1 Detailed Description

Definition at line 82 of file SmiDiscreteDistribution.hpp.

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

- SmiDiscreteDistribution.hpp

4.8 SmiLinearData Class Reference

Inheritance diagram for SmiLinearData:

4.8.1 Detailed Description

Definition at line 23 of file SmiLinearData.hpp.

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

- SmiLinearData.hpp

4.9 SmiMessage Class Reference

This deals with Clp messages (as against Osi messages etc)

`#include <SmiMessage.hpp>`

Inheritance diagram for SmiMessage:

Collaboration diagram for SmiMessage:

Public Member Functions

Constructors etc

- SmiMessage (**Language language**=us_en)
  Constructor.

4.9.1 Detailed Description

This deals with Clp messages (as against Osi messages etc)

Definition at line 20 of file SmiMessage.hpp.

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

- SmiMessage.hpp

4.10 SmiNodeData Class Reference

4.10.1 Detailed Description

Definition at line 27 of file SmiScnData.hpp.

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

- SmiScnData.hpp

4.11 SmiQuadraticData Class Reference

Inheritance diagram for SmiQuadraticData:

4.11.1 Detailed Description

Definition at line 21 of file SmiQuadratic.hpp.

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

- SmiQuadratic.hpp

4.12 SmiQuadraticDataDC Class Reference

Inheritance diagram for SmiQuadraticDataDC:

Collaboration diagram for SmiQuadraticDataDC:

4.12.1 Detailed Description

Definition at line 55 of file SmiQuadratic.hpp.

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

- SmiQuadratic.hpp

4.13 SmiScenarioTree< T > Class Template Reference

Inheritance diagram for SmiScenarioTree< T >:

Public Member Functions

Iterators

- std::vector< T >::iterator treeBegin ()
  begin
- std::vector< T >::iterator treeEnd ()
  end
- std::vector< T > & wholeTree ()
  whole tree
- std::vector< T >::iterator scenBegin (int s)
  scenario iterators TODO: native code for these iterators that does not depend on copying.
- std::vector< T >::iterator **scenEnd** (int s)

Query members

- SmiTreeNode< T > ∗ getRoot ()
  Get root node.
- SmiTreeNode< T > ∗ getLeaf (int scn)
  Get leaf node.
- SmiTreeNode< T > ∗ find (unsigned int scenario, int stage)
  Get node identified by scenario/stage.
- int getNumScenarios ()
  get number of scenarios

- SmiTreeNode< T > ∗ find (std::vector< int > &label)

  Get node identified by longest match to array of labels.
- SmiTreeNode< T > ∗ find (int ∗label, unsigned int sz)

  Get node identified by longest match to array of labels.
- std::vector< T > & getScenario (int scenario)

  Get vector of node data for given scenario.

Tree modification members

- int addPathtoLeaf (int brscenario, int stage, std::vector< T > &pathdata, unsigned int start=0)

  Add new path from branching node to leaf.
- void setChildLabels (SmiTreeNode< T > ∗n, std::vector< int > labels)

  Set child labels.
- int addPathtoLeaf (std::vector< int > &labels, std::vector< T > &pathdata)

  Add new path using labels to find branching node.
- SmiTreeNode< T > ∗ **addNodesToTree** (SmiTreeNode< T > ∗parent, int scenario, std::vector< T > &pathdata, int start)

Constructors, destructors and major modifying methods

- SmiScenarioTree ()

  Default Constructor creates an empty scenario tree.
- virtual ~SmiScenarioTree ()

  Destructor.

4.13.1 Detailed Description

template<class T>class SmiScenarioTree< T >

Definition at line 196 of file SmiScenarioTree.hpp.

4.13.2 Member Function Documentation

4.13.2.1 template<class T> std::vector<T>::iterator **SmiScenarioTree**< T >::scenBegin (int *s*) [inline]

scenario iterators TODO: native code for these iterators that does not depend on copying.

Definition at line 221 of file SmiScenarioTree.hpp.

4.13.2.2 template<class T> **SmiTreeNode**<T>∗ **SmiScenarioTree**< T >::getRoot ( ) [inline]

Get root node.

Definition at line 236 of file SmiScenarioTree.hpp.

4.13.2.3 template<class T> **SmiTreeNode**<T>∗ **SmiScenarioTree**< T >::getLeaf (int *scn*) [inline]

Get leaf node.

Definition at line 241 of file SmiScenarioTree.hpp.

4.13.2.4 template<class T> SmiTreeNode<T>∗ SmiScenarioTree< T >::find (unsigned int *scenario,* int *stage*) `[inline]`

Get node identified by scenario/stage.

Definition at line 246 of file SmiScenarioTree.hpp.

4.13.2.5 template<class T> int SmiScenarioTree< T >::addPathtoLeaf (int *brscenario,* int *stage,* std::vector< T > & *pathdata,* unsigned int *start* = 0) `[inline]`

Add new path from branching node to leaf.

The branching node is the one belonging to "brscenario" at depth "stage". Length of incoming "pathdata" vector is leaf->depth() - stage. Responsibility for memory management of SmiTreeNodeData elements is assigned to Smi←↩ScenarioTree. SmiTreeNodeData elements must be created with "new" operator.

Definition at line 322 of file SmiScenarioTree.hpp.

4.13.2.6 template<class T> int SmiScenarioTree< T >::addPathtoLeaf (std::vector< int > & *labels,* std::vector< T > & *pathdata*) `[inline]`

Add new path using labels to find branching node.

The length of the incoming path is leaf.depth(). Responsibility for memory management of SmiTreeNodeData elements is assigned to SmiScenarioTree. SmiTreeNodeData elements must be created with "new" operator.

Definition at line 353 of file SmiScenarioTree.hpp.

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

- SmiScenarioTree.hpp

4.14 SmiScnModel Class Reference

SmiScnModel: COIN-SMI Scenario Model Class.

```
#include <SmiScnModel.hpp>
```

Public Member Functions

Read SMPS files.

There should be three files: {name}.

[core, time, stoch]. If you have different extension conventions, then you can hack the method yourself. The files can be compressed. The object that reads the files is derived from **CoinMpsIO***.*

The optional argument SmiCoreCombineRule allows user to pass in a class to override the default methods to combine core and stochastic data.

- int **readSmps** (const char ∗name, SmiCoreCombineRule ∗r=NULL)

Writes SMPS files.

This method generates three files {name}.

[core, time, stoch] or {name}.[cor, tim, sto] (see second parameter).

Parameters

name	*The name for the model and the written files*
winFile↩ Extensions	*optional; false by default, so extensions will be [core, time, stoch]. With this parameter set to true, it will be [cor, tim, sto].*
strictFormat	*optional, true by default. Set to false if SMPS files should be written in free format.*

Returns

-1 in case of no existing SMI model, otherwise 0

- int **writeSmps** (const char ∗name, bool winFileExtensions=false, bool strictFormat=true)
- SmiCoreData ∗ **getCore** ()

Direct methods.

Direct methods require the user to create instances of Core data and Scenario data.

Currently, the dimension of the core nodes determines the dimension of the scenario nodes, but this is something that could easily be changed.

- void processDiscreteDistributionIntoScenarios (SmiDiscreteDistribution ∗s, bool test=false)

  generate scenarios from discrete distribution
- void **setModelProb** (double p)
- int **addNodeToSubmodel** (SmiScnNode ∗smiScnNode)
- SmiScenarioIndex generateScenario (SmiCoreData ∗core, **CoinPackedMatrix** ∗matrix, **CoinPackedVector** ∗dclo, **CoinPackedVector** ∗dcup, **CoinPackedVector** ∗dobj, **CoinPackedVector** ∗drlo, **CoinPackedVector** ∗drup, SmiStageIndex branch, SmiScenarioIndex anc, double prob, SmiCoreCombineRule ∗r=SmiCore↩ CombineReplace::Instance())

  generate scenario with ancestor/branch node identification
- SmiScenarioIndex generateScenario (SmiCoreData ∗core, **CoinPackedMatrix** ∗matrix, **CoinPackedVector** ∗dclo, **CoinPackedVector** ∗dcup, **CoinPackedVector** ∗dobj, **CoinPackedVector** ∗drlo, **CoinPackedVector** ∗drup, std::vector< int >labels, double prob, SmiCoreCombineRule ∗r=SmiCoreCombineReplace::Instance())

  generate scenario with labels information
- SmiScenarioIndex generateScenario (**CoinPackedMatrix** ∗matrix, **CoinPackedVector** ∗dclo, **CoinPacked↩ Vector** ∗dcup, **CoinPackedVector** ∗dobj, **CoinPackedVector** ∗drlo, **CoinPackedVector** ∗drup, SmiStage↩ Index branch, SmiScenarioIndex anc, double prob, SmiCoreCombineRule ∗r=SmiCoreCombineReplace::↩ Instance())

  generate scenario with ancestor/branch node identification
- SmiScenarioIndex generateScenario (**CoinPackedMatrix** ∗matrix, **CoinPackedVector** ∗dclo, **CoinPacked↩ Vector** ∗dcup, **CoinPackedVector** ∗dobj, **CoinPackedVector** ∗drlo, **CoinPackedVector** ∗drup, std::vector< int >labels, double prob, SmiCoreCombineRule ∗r=SmiCoreCombineReplace::Instance())

  generate scenario with labels information
- SmiScenarioIndex **generateScenarioFromCore** (SmiCoreData ∗core, double prob, SmiCoreCombineRule ∗r=SmiCoreCombineReplace::Instance())
- SmiScenarioIndex **generateScenarioFromCore** (double prob, SmiCoreCombineRule ∗r=SmiCoreCombine↩ Replace::Instance())

loadOsiSolverData

Loads deterministic equivalent model into internal osi data structures and return handle.

Note: this uses a callback class SmiCoreCombineRule to decide how to combine the core and stochastic data. The user can override the default methods when the scenario is generated (see SmiScnModel::generateScenario) or when SMPS files are processed (see SmiScnModel::readSmps).

- **OsiSolverInterface** ∗ **loadOsiSolverData** ()

- **OsiSolverInterface** ∗ **loadOsiSolverDataForSubproblem** (int stage, int scenStart)
- std::vector< std::pair< double, double > > **solveWS** (**OsiSolverInterface** ∗osiSolver, double objSense)
- std::pair< double, double ∗ > **solveEV** (**OsiSolverInterface** ∗osiSolver, double objSense)
- double **solveEEV** (**OsiSolverInterface** ∗osiSolver, double objSense)
- double **getWSValue** (**OsiSolverInterface** ∗osiSolver, double objSense)
- double **getEVValue** (**OsiSolverInterface** ∗osiSolver, double objSense)
- double **getEEVValue** (**OsiSolverInterface** ∗osiSolver, double objSense)
- void **setCore** (SmiCoreData ∗val)
- SmiScenarioIndex **getNumScenarios** ()
- double **getScenarioProb** (SmiScenarioIndex ns)
- SmiScnNode ∗ **getLeafNode** (SmiScenarioIndex i)
- SmiScnNode ∗ **getRootNode** ()
- int ∗ **getIntegerInd** ()
- int **getIntegerLen** ()
- int ∗ **getBinaryInd** ()
- int **getBinaryLen** ()
- std::vector< int > **getIntIndices** ()
- void **addIntIndice** (int indice)
- double **getObjectiveValue** (SmiScenarioIndex ns)
- double ∗ **getColSolution** (SmiScenarioIndex ns, int ∗length)
- double ∗ **getRowSolution** (SmiScenarioIndex ns, int ∗length)
- double ∗ **getRowDuals** (SmiScenarioIndex ns, int ∗length)
- double **getColSolution** (SmiScenarioIndex ns, int stage, int colIndex)
- double **getRowSolution** (SmiScenarioIndex ns, int stage, int rowIndex)
- double **getRowDuals** (SmiScenarioIndex ns, int stage, int rowIndex)
- double ∗ **getColValue** (const double ∗d, SmiScenarioIndex ns, int ∗length)
- double **getColValue** (const double ∗d, SmiScenarioIndex ns, int stage, int rowIndex)
- double ∗ **getRowValue** (const double ∗d, SmiScenarioIndex ns, int ∗length, bool isDual)
- double **getRowValue** (const double ∗d, SmiScenarioIndex ns, int stage, int rowIndex, bool isDual)
- void **setOsiSolverHandle** (**OsiSolverInterface** &osi)
- void **setOsiSolverHandle** (**OsiSolverInterface** ∗osi)
- **OsiSolverInterface** ∗ **getOsiSolverInterface** ()
- void **releaseSolver** ()
- void **releaseCore** ()
- void **setQuadraticSolver** (**ClpModel** ∗clp)
- **ClpModel** ∗ **getQuadraticSolver** ()
- **ClpModel** ∗ **loadQuadraticSolverData** ()
- **SmiScnModel** ()
- ∼**SmiScnModel** ()
- void **addNode** (SmiScnNode ∗node, bool notDetEq=false)
- void **deleteNode** (SmiScnNode ∗tnode)
- void **addNode** (SmiNodeData ∗node)
- SmiScenarioTree< SmiScnNode ∗ > ∗ **getSmiTree** ()

4.14.1 Detailed Description

SmiScnModel: COIN-SMI Scenario Model Class.

Concrete class for generating scenario stochastic linear programs.

This class implements the Scenarios format of the Stochastic MPS modeling system (TODO: web pointer). Core data and Scenarios data can be passed using COIN/OSI structures, or can be read from SMPS formatted files.

Typical driver fragment looks like this

```
SmiScnModel smi;
smi.readSmps("app0110R");
smi.setOsiSolverHandle(new OsiClpSolverInterface());
OsiSolverInterface *osiStoch = smi.loadOsiSolverData();
osiStoch->initialSolve();
```

The setOsiSolverHandle method allows the user to pass in any OSI compatible solver.

Definition at line 49 of file SmiScnModel.hpp.

4.14.2 Member Function Documentation

4.14.2.1 SmiScenarioIndex SmiScnModel::generateScenario (**SmiCoreData** ∗ *core*, **CoinPackedMatrix** ∗ *matrix*, **CoinPackedVector** ∗ *dclo*, **CoinPackedVector** ∗ *dcup*, **CoinPackedVector** ∗ *dobj*, **CoinPackedVector** ∗ *drlo*, **CoinPackedVector** ∗ *drup*, SmiStageIndex *branch*, SmiScenarioIndex *anc*, double *prob*, **SmiCoreCombineRule** ∗ *r* = `SmiCoreCombineReplace::Instance()`)

generate scenario with ancestor/branch node identification

Core argument must be supplied. Data values combine with corresponding core values, if found, or creates them if not.

Scenario nodes need to have same dimensions as core nodes.

Data field arguments can be NULL, or empty.

branch, anc, arguments must be supplied. These identify the branching node according to the Stochastic MPS standard.

prob is unconditional probability of scenario

4.14.2.2 SmiScenarioIndex SmiScnModel::generateScenario (**SmiCoreData** ∗ *core*, **CoinPackedMatrix** ∗ *matrix*, **CoinPackedVector** ∗ *dclo*, **CoinPackedVector** ∗ *dcup*, **CoinPackedVector** ∗ *dobj*, **CoinPackedVector** ∗ *drlo*, **CoinPackedVector** ∗ *drup*, std::vector< int > *labels*, double *prob*, **SmiCoreCombineRule** ∗ *r* = `SmiCoreCombineReplace::Instance()`)

generate scenario with labels information

Core argument must be supplied. Data values combine with corresponding core values, if found, or creates them if not.

Scenario nodes need to have same dimensions as core nodes.

Data field arguments can be NULL, or empty.

Labels are passed as vector<int> array. Adds new path using labels to find branching node. The depth (root to leaf) of new path is labels.size().

4.14.2.3 SmiScenarioIndex SmiScnModel::generateScenario (**CoinPackedMatrix** ∗ *matrix,* **CoinPackedVector** ∗ *dclo,* **CoinPackedVector** ∗ *dcup,* **CoinPackedVector** ∗ *dobj,* **CoinPackedVector** ∗ *drlo,* **CoinPackedVector** ∗ *drup,* SmiStageIndex *branch,* SmiScenarioIndex *anc,* double *prob,* **SmiCoreCombineRule** ∗ *r =* `SmiCoreCombineReplace::Instance()`)

generate scenario with ancestor/branch node identification

Core argument must be supplied. Data values combine with corresponding core values, if found, or creates them if not.

Scenario nodes need to have same dimensions as core nodes.

Data field arguments can be NULL, or empty.

branch, anc, arguments must be supplied. These identify the branching node according to the Stochastic MPS standard.

prob is unconditional probability of scenario

4.14.2.4 SmiScenarioIndex SmiScnModel::generateScenario (**CoinPackedMatrix** ∗ *matrix,* **CoinPackedVector** ∗ *dclo,* **CoinPackedVector** ∗ *dcup,* **CoinPackedVector** ∗ *dobj,* **CoinPackedVector** ∗ *drlo,* **CoinPackedVector** ∗ *drup,* std::vector< int > *labels,* double *prob,* **SmiCoreCombineRule** ∗ *r =* `SmiCoreCombineReplace::Instance()`)

generate scenario with labels information

Core argument must be supplied. Data values combine with corresponding core values, if found, or creates them if not.

Scenario nodes need to have same dimensions as core nodes.

Data field arguments can be NULL, or empty.

Labels are passed as vector<int> array. Adds new path using labels to find branching node. The depth (root to leaf) of new path is labels.size().

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

- SmiScnModel.hpp

4.15 SmiScnModelAddNode Class Reference

4.15.1 Detailed Description

Definition at line 440 of file SmiScnModel.hpp.

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

- SmiScnModel.hpp

4.16 SmiScnModelDeleteNode Class Reference

4.16.1 Detailed Description

Definition at line 455 of file SmiScnModel.hpp.

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

- SmiScnModel.hpp

4.17 SmiScnNode Class Reference

Friends

- class **SmiScnModel**

4.17.1 Detailed Description

Definition at line 360 of file SmiScnModel.hpp.

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

- SmiScnModel.hpp

4.18 SmiSmpsCardReader Class Reference

Inheritance diagram for SmiSmpsCardReader:

Collaboration diagram for SmiSmpsCardReader:

Public Member Functions

- SmiSmpsCardReader (**CoinFileInput** ∗input, **CoinMpsIO** ∗reader)

  Constructor expects file to be open This one takes gzFile if fp null.

4.18.1 Detailed Description

Definition at line 41 of file SmiSmpsIO.hpp.

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

- SmiSmpsIO.hpp

4.19 SmiSmpsIO Class Reference

Inheritance diagram for SmiSmpsIO:

Collaboration diagram for SmiSmpsIO:

4.19.1 Detailed Description

Definition at line 76 of file SmiSmpsIO.hpp.

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

- SmiSmpsIO.hpp

4.20 SmiTreeNode< T > Class Template Reference

Scenario Tree.

```
#include <SmiScenarioTree.hpp>
```

Inheritance diagram for SmiTreeNode< T >:

Public Member Functions

Constructors, destructors and major modifying methods

- SmiTreeNode ()

  Default Constructor creates an empty node.
- SmiTreeNode (T p)

  Constructor from P.
- ∼SmiTreeNode ()

  Destructor.

4.20.1 Detailed Description

template<class T>class SmiTreeNode< T >

Scenario Tree.

This class is used for storing and accessing scenario trees. SmiTreeNode template class.

Manages pointers to parent, child and sibling for tree navigation. Template class instance is a pointer to an object that must be created with "new" operator.

Definition at line 27 of file SmiScenarioTree.hpp.

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

- SmiScenarioTree.hpp

File Documentation

Index