Vol

1.5

Generated by Doxygen 1.8.9.1

Thu Oct 8 2015 22:41:21

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

- _EKKfactinfo [external]
- forcing_constraint_action::action [external]
- doubleton_action::action [external]
- tripleton_action::action [external]
- remove_fixed_action::action [external]
- std::allocator< T >
- OsiSolverInterface::ApplyCutsReturnCode [external]
- std::array< T >
- std::auto_ptr< T >
- std::basic_string< Char >
  - std::string
  - std::wstring
- std::basic_string< char >
- std::basic_string< wchar_t >
- std::bitset< Bits >
- BitVector128 [external]
- CoinAbsFltEq [external]
- CoinArrayWithLength [external]
  - CoinArbitraryArrayWithLength [external]
  - CoinBigIndexArrayWithLength [external]
  - CoinDoubleArrayWithLength [external]
  - CoinFactorizationDoubleArrayWithLength [external]
  - CoinFactorizationLongDoubleArrayWithLength [external]
  - CoinIntArrayWithLength [external]
  - CoinUnsignedIntArrayWithLength [external]
  - CoinVoidStarArrayWithLength [external]
- CoinBaseModel [external]
  - CoinModel [external]
  - CoinStructuredModel [external]
- CoinBuild [external]
- CoinDenseVector< T > [external]
- CoinError [external]
- CoinExternalVectorFirstGreater_2< class, class, class > [external]
- CoinExternalVectorFirstGreater_3< class, class, class, class > [external]

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
std::multimap< K, T >::iterator
std::unordered_set< K >::iterator
std::unordered_multimap< K, T >::iterator
std::set< K >::iterator
std::list< T >::iterator
std::unordered_multiset< K >::iterator

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 OsiVolSolverInterface Class Reference

Vol(ume) Solver Interface.

`#include <OsiVolSolverInterface.hpp>`

Inheritance diagram for OsiVolSolverInterface:

4.2 VOL_alpha_factor Class Reference

4.2.1 Detailed Description

Definition at line 489 of file VolVolume.hpp.

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

- VolVolume.hpp

4.3 VOL_dual Class Reference

Collaboration diagram for VOL_dual:

4.3.1 Detailed Description

Definition at line 354 of file VolVolume.hpp.

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

- VolVolume.hpp

4.4 VOL_dvector Class Reference

vector of doubles.

`#include <VolVolume.hpp>`

Public Member Functions

- VOL_dvector (const int s)

  Construct a vector of size s.
- VOL_dvector ()

  Default constructor creates a vector of size 0.
- VOL_dvector (const VOL_dvector &x)

  Copy constructor makes a replica of x.
- ~VOL_dvector ()

  The destructor deletes the data array.
- int size () const

  Return the size of the vector.
- double & operator[ ] (const int i)

  Return a reference to the `i`-th entry.
- double operator[ ] (const int i) const

  Return the `i`-th entry.
- void clear ()

  Delete the content of the vector and replace it with a vector of length 0.
- void cc (const double gamma, const VOL_dvector &w)

  Convex combination.
- void allocate (const int s)

  delete the current vector and allocate space for a vector of size `s`.
- void swap (VOL_dvector &w)

  swaps the vector with `w`.
- VOL_dvector & operator= (const VOL_dvector &w)

  Copy `w` into the vector.
- VOL_dvector & operator= (const double w)

  Replace every entry in the vector with `w`.

Public Attributes

- double * v

  The array holding the vector.
- int sz

  The size of the vector.

4.4.1 Detailed Description

vector of doubles.

It is used for most vector operations.

Note: If `VOL_DEBUG` is `#defined` to be 1 then each time an entry is accessed in the vector the index of the entry is tested for nonnegativity and for being less than the size of the vector. It's good to turn this on while debugging, but in final runs it should be turned off (beause of the performance hit).

Definition at line 148 of file VolVolume.hpp.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 VOL_dvector::VOL_dvector (const int *s*) [inline]

Construct a vector of size s.

The content of the vector is undefined.

Definition at line 157 of file VolVolume.hpp.

4.4.2.2 VOL_dvector::VOL_dvector ( ) [inline]

Default constructor creates a vector of size 0.

Definition at line 162 of file VolVolume.hpp.

4.4.2.3 VOL_dvector::VOL_dvector (const VOL_dvector & *x*) [inline]

Copy constructor makes a replica of x.

Definition at line 164 of file VolVolume.hpp.

4.4.2.4 VOL_dvector::~VOL_dvector ( ) [inline]

The destructor deletes the data array.

Definition at line 172 of file VolVolume.hpp.

4.4.3 Member Function Documentation

4.4.3.1 int VOL_dvector::size ( ) const [inline]

Return the size of the vector.

Definition at line 175 of file VolVolume.hpp.

4.4.3.2 double& VOL_dvector::operator[ ] (const int *i*) [inline]

Return a reference to the `i`-th entry.

Definition at line 178 of file VolVolume.hpp.

4.4.3.3 double VOL_dvector::operator[ ] (const int *i*) const [inline]

Return the `i`-th entry.

Definition at line 184 of file VolVolume.hpp.

4.4.3.4 void VOL_dvector::clear ( ) [inline]

Delete the content of the vector and replace it with a vector of length 0.

Definition at line 191 of file VolVolume.hpp.

4.4.3.5 void VOL_dvector::cc (const double *gamma,* const VOL_dvector & *w*) [inline]

Convex combination.

Replace the current vector `v` with `v = (1-gamma) v + gamma w`.

Definition at line 198 of file VolVolume.hpp.

4.4.3.6 void VOL_dvector::allocate (const int *s*) [inline]

delete the current vector and allocate space for a vector of size `s`.

Definition at line 214 of file VolVolume.hpp.

4.4.3.7 void VOL_dvector::swap (VOL_dvector & *w*) [inline]

swaps the vector with `w`.

Definition at line 221 of file VolVolume.hpp.

4.4.3.8 VOL_dvector& VOL_dvector::operator= (const VOL_dvector & *w*)

Copy `w` into the vector.

4.4.3.9 VOL_dvector& VOL_dvector::operator= (const double *w*)

Replace every entry in the vector with `w`.

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

- VolVolume.hpp

4.5 VOL_indc Class Reference

4.5.1 Detailed Description

Definition at line 538 of file VolVolume.hpp.

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

- VolVolume.hpp

4.6 VOL_ivector Class Reference

vector of ints.

`#include <VolVolume.hpp>`

Public Member Functions

- VOL_ivector (const int s)

  Construct a vector of size s.
- VOL_ivector ()

  Default constructor creates a vector of size 0.
- VOL_ivector (const VOL_ivector &x)

  Copy constructor makes a replica of x.
- ~VOL_ivector ()

  The destructor deletes the data array.
- int size () const

  Return the size of the vector.
- int & operator[ ] (const int i)

  Return a reference to the `i`-th entry.
- int operator[ ] (const int i) const

  Return the `i`-th entry.
- void clear ()

  Delete the content of the vector and replace it with a vector of length 0.
- void allocate (const int s)

  delete the current vector and allocate space for a vector of size `s`.
- void swap (VOL_ivector &w)

  swaps the vector with `w`.
- VOL_ivector & operator= (const VOL_ivector &v)

  Copy `w` into the vector.
- VOL_ivector & operator= (const int w)

  Replace every entry in the vector with `w`.

Public Attributes

- int * v

  The array holding the vector.
- int sz

  The size of the vector.

4.6.1 Detailed Description

vector of ints.

It's used to store indices, it has similar functions as VOL_dvector.

Note: If `VOL_DEBUG` is `#defined` to be 1 then each time an entry is accessed in the vector the index of the entry is tested for nonnegativity and for being less than the size of the vector. It's good to turn this on while debugging, but in final runs it should be turned off (beause of the performance hit).

Definition at line 242 of file VolVolume.hpp.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 VOL_ivector::VOL_ivector (const int *s*) [inline]

Construct a vector of size s.

The content of the vector is undefined.

Definition at line 250 of file VolVolume.hpp.

4.6.2.2 VOL_ivector::VOL_ivector ( ) [inline]

Default constructor creates a vector of size 0.

Definition at line 255 of file VolVolume.hpp.

4.6.2.3 VOL_ivector::VOL_ivector (const VOL_ivector & *x*) [inline]

Copy constructor makes a replica of x.

Definition at line 257 of file VolVolume.hpp.

4.6.2.4 VOL_ivector::~VOL_ivector ( ) [inline]

The destructor deletes the data array.

Definition at line 265 of file VolVolume.hpp.

4.6.3 Member Function Documentation

4.6.3.1 int VOL_ivector::size ( ) const [inline]

Return the size of the vector.

Definition at line 270 of file VolVolume.hpp.

4.6.3.2 int& VOL_ivector::operator[ ] (const int *i*) [inline]

Return a reference to the `i`-th entry.

Definition at line 272 of file VolVolume.hpp.

4.6.3.3 int VOL_ivector::operator[ ] (const int *i*) const [inline]

Return the `i`-th entry.

Definition at line 278 of file VolVolume.hpp.

4.6.3.4 void VOL_ivector::clear ( ) [inline]

Delete the content of the vector and replace it with a vector of length 0.

Definition at line 285 of file VolVolume.hpp.

4.6.3.5 void VOL_ivector::allocate (const int *s*) [inline]

delete the current vector and allocate space for a vector of size `s`.

Definition at line 293 of file VolVolume.hpp.

4.6.3.6 void VOL_ivector::swap (VOL_ivector & *w*) [inline]

swaps the vector with `w`.

Definition at line 300 of file VolVolume.hpp.

4.6.3.7 VOL_ivector& VOL_ivector::operator= (const VOL_ivector & *v*)

Copy `w` into the vector.

4.6.3.8 VOL_ivector& VOL_ivector::operator= (const int *w*)

Replace every entry in the vector with `w`.

4.6.4 Member Data Documentation

4.6.4.1 int* VOL_ivector::v

The array holding the vector.

Definition at line 245 of file VolVolume.hpp.

4.6.4.2 int VOL_ivector::sz

The size of the vector.

Definition at line 247 of file VolVolume.hpp.

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

- VolVolume.hpp

4.7 VOL_parms Struct Reference

This class contains the parameters controlling the Volume Algorithm.

`#include <VolVolume.hpp>`

Public Attributes

- double lambdainit
  initial value of lambda
- double alphainit
  initial value of alpha

- double alphamin

  minimum value for alpha
- double alphafactor

  when little progress is being done, we multiply alpha by alphafactor
- double ubinit

  initial upper bound of the value of an integer solution
- double primal_abs_precision

  accept if max abs viol is less than this
- double gap_abs_precision

  accept if abs gap is less than this
- double gap_rel_precision

  accept if rel gap is less than this
- double granularity

  terminate if best_ub - lcost $<$ granularity
- double minimum_rel_ascent

  terminate if the relative increase in lcost through `ascent_check_invl` *steps is less than this*
- int ascent_first_check

  when to check for sufficient relative ascent the first time
- int ascent_check_invl

  through how many iterations does the relative ascent have to reach a minimum
- int maxsgriters

  maximum number of iterations
- int printflag

  controls the level of printing.
- int printinvl

  controls how often do we print
- int heurinvl

  controls how often we run the primal heuristic
- int greentestinvl

  how many consecutive green iterations are allowed before changing lambda
- int yellowtestinvl

  how many consecutive yellow iterations are allowed before changing lambda
- int redtestinvl

  how many consecutive red iterations are allowed before changing lambda
- int alphaint

  number of iterations before we check if alpha should be decreased
- char $*$ temp_dualfile

  name of file for saving dual solution

4.7.1 Detailed Description

This class contains the parameters controlling the Volume Algorithm.

Definition at line 71 of file VolVolume.hpp.

4.7.2 Member Data Documentation

4.7.2.1 int VOL_parms::printflag

controls the level of printing.

The flag should the the 'OR'-d value of the following options:

- 0 - print nothing
- 1 - print iteration information
- 2 - add lambda information
- 4 - add number of Red, Yellow, Green iterations

Default: 3

Definition at line 115 of file VolVolume.hpp.

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

- VolVolume.hpp

4.8 VOL_primal Class Reference

Collaboration diagram for VOL_primal:

4.8.1 Detailed Description

Definition at line 313 of file VolVolume.hpp.

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

- VolVolume.hpp

4.9 VOL_problem Class Reference

This class holds every data for the Volume Algorithm and its `solve` method must be invoked to solve the problem.

`#include <VolVolume.hpp>`

Collaboration diagram for VOL_problem:

Public Member Functions

Constructors and destructor

- VOL_problem ()
  Default constructor.
- VOL_problem (const char *filename)
  Create a a `VOL_problem` object and read in the parameters from `filename`.
- ~VOL_problem ()

Destruct the object.

Method to solve the problem.

- int solve (VOL_user_hooks &hooks, const bool use_preset_dual=false)
  Solve the problem using the `hooks`.

Methods returning final data

- int iter () const
  returns the iteration number
- double alpha () const
  returns the value of alpha
- double lambda () const
  returns the value of lambda

Public Attributes

- int iter_
  iteration number

External data (containing the result after solve)

- double value
  final lagrangian value (OUTPUT)
- VOL_dvector dsol
  final dual solution (INPUT/OUTPUT)
- VOL_dvector psol
  final primal solution (OUTPUT)
- VOL_dvector viol
  violations (b-Ax) for the relaxed constraints

External data (may be changed by the user before calling solve)

- VOL_parms parm
  The parameters controlling the Volume Algorithm (INPUT)
- int psize
  length of primal solution (INPUT)
- int dsize
  length of dual solution (INPUT)
- VOL_dvector dual_lb
  lower bounds for the duals (if 0 length, then filled with -inf) (INPUT)
- VOL_dvector dual_ub
  upper bounds for the duals (if 0 length, then filled with +inf) (INPUT)

4.9.1 Detailed Description

This class holds every data for the Volume Algorithm and its `solve` method must be invoked to solve the problem.

The INPUT fields must be filled out completely before `solve` is invoked. `dsol` have to be filled out if and only if the last argument to `solve` is `true`.

Definition at line 605 of file VolVolume.hpp.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 VOL_problem::VOL_problem ( )

Default constructor.

4.9.2.2 VOL_problem::VOL_problem (const char * *filename*)

Create a a `VOL_problem` object and read in the parameters from `filename`.

4.9.2.3 VOL_problem::~VOL_problem ( )

Destruct the object.

4.9.3 Member Function Documentation

4.9.3.1 int VOL_problem::solve (VOL_user_hooks & *hooks*, const bool *use_preset_dual* = `false`)

Solve the problem using the `hooks`.

Any information needed in the hooks must be stored in the structure `user_data` points to.

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

- VolVolume.hpp

4.10 VOL_swing Class Reference

4.10.1 Detailed Description

Definition at line 390 of file VolVolume.hpp.

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

- VolVolume.hpp

4.11 VOL_user_hooks Class Reference

The user hooks should be overridden by the user to provide the problem specific routines for the volume algorithm.

`#include <VolVolume.hpp>`

Inheritance diagram for VOL_user_hooks:

Public Member Functions

- virtual int compute_rc (const VOL_dvector &u, VOL_dvector &rc)=0

  compute reduced costs

- virtual int solve_subproblem (const VOL_dvector &dual, const VOL_dvector &rc, double &lcost, VOL_dvector &x, VOL_dvector &v, double &pcost)=0

  Solve the subproblem for the subgradient step.
- virtual int heuristics (const VOL_problem &p, const VOL_dvector &x, double &heur_val)=0

  Starting from the primal vector x, run a heuristic to produce an integer solution.

4.11.1 Detailed Description

The user hooks should be overridden by the user to provide the problem specific routines for the volume algorithm.

The user should derive a class ...

for all hooks: return value of -1 means that volume should quit

Definition at line 563 of file VolVolume.hpp.

4.11.2 Member Function Documentation

4.11.2.1 virtual int VOL_user_hooks::compute_rc (const VOL_dvector & *u*, VOL_dvector & *rc*) [pure virtual]

compute reduced costs

Parameters

u	(IN) the dual variables
rc	(OUT) the reduced cost with respect to the dual values

4.11.2.2 virtual int VOL_user_hooks::solve_subproblem (const VOL_dvector & *dual*, const VOL_dvector & *rc*, double & *lcost*, VOL_dvector & *x*, VOL_dvector & *v*, double & *pcost*) [pure virtual]

Solve the subproblem for the subgradient step.

Parameters

dual	(IN) the dual variables
rc	(IN) the reduced cost with respect to the dual values
lcost	(OUT) the lagrangean cost with respect to the dual values
x	(OUT) the primal result of solving the subproblem
v	(OUT) b-Ax for the relaxed constraints
pcost	(OUT) the primal objective value of `x`

4.11.2.3 virtual int VOL_user_hooks::heuristics (const VOL_problem & *p*, const VOL_dvector & *x*, double & *heur_val*) [pure virtual]

Starting from the primal vector x, run a heuristic to produce an integer solution.

Parameters

x	(IN) the primal vector

heur_val	(OUT) the value of the integer solution (return `DBL_MAX` here if no feas sol was found

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

- VolVolume.hpp

4.12 VOL_vh Class Reference

4.12.1 Detailed Description

Definition at line 515 of file VolVolume.hpp.

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

- VolVolume.hpp

File Documentation

Index