

Blis
0.94

Generated by Doxygen 1.8.9.1

Thu Oct 8 2015 22:51:24

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	13
2.1	Class List	13
3	File Index	15
3.1	File List	15
4	Class Documentation	17
4.1	BlisBranchObjectBilevel Class Reference	17
4.2	BlisBranchObjectInt Class Reference	17
4.2.1	Detailed Description	18
4.2.2	Constructor & Destructor Documentation	18
4.2.2.1	BlisBranchObjectInt	18
4.2.2.2	BlisBranchObjectInt	18
4.2.2.3	BlisBranchObjectInt	19
4.2.2.4	BlisBranchObjectInt	20
4.2.2.5	BlisBranchObjectInt	20
4.2.2.6	~BlisBranchObjectInt	20
4.2.3	Member Function Documentation	20
4.2.3.1	operator=	20
4.2.3.2	clone	20
4.2.3.3	branch	20
4.2.3.4	print	21
4.2.3.5	getDown	21
4.2.3.6	getUp	21
4.2.3.7	encodeBlis	21
4.2.3.8	decodeBlis	21

4.2.3.9	encode	21
4.2.3.10	decode	21
4.3	BlisBranchStrategyBilevel Class Reference	21
4.3.1	Detailed Description	22
4.3.2	Constructor & Destructor Documentation	22
4.3.2.1	BlisBranchStrategyBilevel	22
4.3.2.2	BlisBranchStrategyBilevel	22
4.3.2.3	~BlisBranchStrategyBilevel	22
4.3.2.4	BlisBranchStrategyBilevel	23
4.3.3	Member Function Documentation	23
4.3.3.1	clone	23
4.3.3.2	createCandBranchObjects	23
4.3.3.3	betterBranchObject	23
4.4	BlisBranchStrategyMaxInf Class Reference	23
4.4.1	Detailed Description	24
4.4.2	Constructor & Destructor Documentation	24
4.4.2.1	BlisBranchStrategyMaxInf	24
4.4.2.2	BlisBranchStrategyMaxInf	24
4.4.2.3	~BlisBranchStrategyMaxInf	24
4.4.2.4	BlisBranchStrategyMaxInf	24
4.4.3	Member Function Documentation	24
4.4.3.1	clone	24
4.4.3.2	createCandBranchObjects	25
4.4.3.3	betterBranchObject	25
4.5	BlisBranchStrategyPseudo Class Reference	25
4.5.1	Detailed Description	26
4.5.2	Constructor & Destructor Documentation	26
4.5.2.1	BlisBranchStrategyPseudo	26
4.5.2.2	BlisBranchStrategyPseudo	26
4.5.2.3	~BlisBranchStrategyPseudo	26
4.5.2.4	BlisBranchStrategyPseudo	26
4.5.3	Member Function Documentation	26
4.5.3.1	setReliability	26
4.5.3.2	clone	26
4.5.3.3	betterBranchObject	26
4.5.3.4	createCandBranchObjects	27
4.6	BlisBranchStrategyRel Class Reference	27

4.6.1	Detailed Description	27
4.6.2	Constructor & Destructor Documentation	28
4.6.2.1	BlisBranchStrategyRel	28
4.6.2.2	BlisBranchStrategyRel	28
4.6.2.3	~BlisBranchStrategyRel	28
4.6.2.4	BlisBranchStrategyRel	28
4.6.3	Member Function Documentation	28
4.6.3.1	setRelibility	28
4.6.3.2	clone	28
4.6.3.3	betterBranchObject	28
4.6.3.4	createCandBranchObjects	28
4.7	BlisBranchStrategyStrong Class Reference	29
4.7.1	Detailed Description	29
4.7.2	Constructor & Destructor Documentation	29
4.7.2.1	BlisBranchStrategyStrong	29
4.7.2.2	BlisBranchStrategyStrong	29
4.7.2.3	~BlisBranchStrategyStrong	30
4.7.2.4	BlisBranchStrategyStrong	30
4.7.3	Member Function Documentation	30
4.7.3.1	clone	30
4.7.3.2	createCandBranchObjects	30
4.7.3.3	betterBranchObject	30
4.8	BlisConGenerator Class Reference	30
4.8.1	Detailed Description	33
4.8.2	Constructor & Destructor Documentation	33
4.8.2.1	BlisConGenerator	33
4.8.2.2	BlisConGenerator	33
4.8.2.3	BlisConGenerator	33
4.8.2.4	~BlisConGenerator	33
4.8.3	Member Function Documentation	33
4.8.3.1	operator=	33
4.8.3.2	generateConstraints	33
4.8.3.3	getModel	34
4.8.3.4	setName	34
4.8.3.5	name	34
4.8.3.6	setStrategy	34
4.8.3.7	strategy	34

4.8.3.8	setCutGenerationFreq	34
4.8.3.9	cutGenerationFreq	34
4.8.3.10	normal	34
4.8.3.11	setNormal	35
4.8.3.12	atSolution	35
4.8.3.13	setAtSolution	35
4.8.3.14	whenInfeasible	35
4.8.3.15	setWhenInfeasible	35
4.8.3.16	generator	35
4.8.3.17	numConsGenerated	35
4.8.3.18	addNumConsGenerated	35
4.8.3.19	numConsUsed	35
4.8.3.20	addNumConsUsed	36
4.8.3.21	time	36
4.8.3.22	addTime	36
4.8.3.23	calls	36
4.8.3.24	addCalls	36
4.8.3.25	noConsCalls	36
4.8.3.26	addNoConsCalls	36
4.8.4	Member Data Documentation	36
4.8.4.1	model_	36
4.8.4.2	generator_	37
4.8.4.3	strategy_	37
4.8.4.4	cutGenerationFrequency_	37
4.8.4.5	name_	37
4.8.4.6	normal_	37
4.8.4.7	atSolution_	37
4.8.4.8	whenInfeasible_	37
4.8.4.9	numConsGenerated_	37
4.8.4.10	numConsUsed_	38
4.8.4.11	time_	38
4.8.4.12	calls_	38
4.8.4.13	noConsCalls_	38
4.9	BlisConstraint Class Reference	38
4.9.1	Detailed Description	39
4.9.2	Constructor & Destructor Documentation	39
4.9.2.1	BlisConstraint	39

4.9.2.2	BlisConstraint	39
4.9.2.3	BlisConstraint	40
4.9.2.4	BlisConstraint	40
4.9.2.5	~BlisConstraint	40
4.9.2.6	BlisConstraint	40
4.9.3	Member Function Documentation	40
4.9.3.1	encodeBlis	40
4.9.3.2	decodeBlis	40
4.9.3.3	createOsiRowCut	40
4.9.3.4	hashing	40
4.9.3.5	violation	40
4.9.3.6	encode	40
4.9.3.7	decode	41
4.10	BlisHeuristic Class Reference	41
4.10.1	Detailed Description	42
4.10.2	Constructor & Destructor Documentation	42
4.10.2.1	BlisHeuristic	42
4.10.2.2	BlisHeuristic	43
4.10.2.3	~BlisHeuristic	43
4.10.2.4	BlisHeuristic	43
4.10.3	Member Function Documentation	43
4.10.3.1	setModel	43
4.10.3.2	setStrategy	43
4.10.3.3	setHeurCallFrequency	43
4.10.3.4	clone	43
4.10.3.5	name	43
4.10.3.6	addNumSolutions	44
4.10.3.7	numSolutions	44
4.10.3.8	addTime	44
4.10.3.9	time	44
4.10.3.10	addCalls	44
4.10.3.11	calls	44
4.10.3.12	noSolCalls	44
4.10.3.13	addNoSolCalls	44
4.10.4	Member Data Documentation	44
4.10.4.1	strategy_	45
4.10.4.2	numSolutions_	45

4.10.4.3	time_	45
4.10.4.4	calls_	45
4.10.4.5	noSolsCalls_	45
4.11	BlisHeurRound Class Reference	45
4.11.1	Detailed Description	46
4.11.2	Constructor & Destructor Documentation	46
4.11.2.1	BlisHeurRound	46
4.11.2.2	BlisHeurRound	46
4.11.2.3	~BlisHeurRound	46
4.11.2.4	BlisHeurRound	47
4.11.3	Member Function Documentation	47
4.11.3.1	setModel	47
4.11.4	Member Data Documentation	47
4.11.4.1	matrix_	47
4.11.4.2	matrixByRow_	47
4.11.4.3	seed_	47
4.12	BlisMessage Class Reference	47
4.12.1	Detailed Description	48
4.13	BlisModel Class Reference	48
4.13.1	Detailed Description	56
4.13.2	Constructor & Destructor Documentation	56
4.13.2.1	BlisModel	56
4.13.2.2	~BlisModel	56
4.13.3	Member Function Documentation	56
4.13.3.1	createObjects	56
4.13.3.2	gutsOfDestructor	56
4.13.3.3	readInstance	57
4.13.3.4	importModel	57
4.13.3.5	readParameters	57
4.13.3.6	writeParameters	57
4.13.3.7	createRoot	57
4.13.3.8	setUpSelf	57
4.13.3.9	preprocess	57
4.13.3.10	postprocess	58
4.13.3.11	setSolver	58
4.13.3.12	getSolver	58
4.13.3.13	solver	58

4.13.3.14 resolve	58
4.13.3.15 setActiveNode	58
4.13.3.16 setSolEstimate	58
4.13.3.17 getNumStrong	58
4.13.3.18 addNumStrong	58
4.13.3.19 getNumBranchResolve	59
4.13.3.20 setNumBranchResolve	59
4.13.3.21 getObjCoef	59
4.13.3.22 getColLower	59
4.13.3.23 getColUpper	59
4.13.3.24 getNumCols	59
4.13.3.25 getNumRows	59
4.13.3.26 varLB	59
4.13.3.27 conLB	59
4.13.3.28 startVarLB	60
4.13.3.29 startConLB	60
4.13.3.30 tempVarLBPos	60
4.13.3.31 getLpObjValue	60
4.13.3.32 getLpSolution	60
4.13.3.33 getNumSolutions	60
4.13.3.34 getNumHeurSolutions	60
4.13.3.35 incumbent	60
4.13.3.36 storeSolution	60
4.13.3.37 getCutoff	61
4.13.3.38 setCutoff	61
4.13.3.39 feasibleSolutionHeur	61
4.13.3.40 feasibleSolution	61
4.13.3.41 userFeasibleSolution	61
4.13.3.42 branchStrategy	61
4.13.3.43 setBranchingMethod	61
4.13.3.44 setBranchingMethod	61
4.13.3.45 numObjects	62
4.13.3.46 setNumObjects	62
4.13.3.47 objects	62
4.13.3.48 objects	62
4.13.3.49 setSharedObjectMark	62
4.13.3.50 clearSharedObjectMark	62

4.13.3.51 deleteObjects	62
4.13.3.52 addObjects	62
4.13.3.53 createIntgerObjects	62
4.13.3.54 getIntObjIndices	63
4.13.3.55 getNumIntObjects	63
4.13.3.56 getIntCollIndices	63
4.13.3.57 checkInteger	63
4.13.3.58 addHeuristic	63
4.13.3.59 heuristics	63
4.13.3.60 numHeuristics	63
4.13.3.61 addCutGenerator	63
4.13.3.62 addCutGenerator	63
4.13.3.63 cutGenerators	64
4.13.3.64 numCutGenerators	64
4.13.3.65 getMaxNumCons	64
4.13.3.66 setMaxNumCons	64
4.13.3.67 constraintPool	64
4.13.3.68 constraintPoolReceive	64
4.13.3.69 constraintPoolSend	64
4.13.3.70 getNumOldConstraints	64
4.13.3.71 setNumOldConstraints	64
4.13.3.72 getOldConstraintsSize	65
4.13.3.73 setOldConstraintsSize	65
4.13.3.74 oldConstraints	65
4.13.3.75 setOldConstraints	65
4.13.3.76 delOldConstraints	65
4.13.3.77 getCutStrategy	65
4.13.3.78 setCutStrategy	65
4.13.3.79 getCutGenerationFrequency	65
4.13.3.80 setCutStrategy	65
4.13.3.81 getDenseConCutoff	66
4.13.3.82 setDenseConCutoff	66
4.13.3.83 getConRandoms	66
4.13.3.84 passInPriorities	66
4.13.3.85 priority	66
4.13.3.86 modelLog	66
4.13.3.87 getNumNodes	66

4.13.3.88	getNumIterations	66
4.13.3.89	getAveIterations	67
4.13.3.90	addNumNodes	67
4.13.3.91	addNumIterations	67
4.13.3.92	blisMessageHandler	67
4.13.3.93	blisMessages	67
4.13.3.94	BlisPar	67
4.13.3.95	nodeLog	67
4.13.3.96	fathomAllNodes	67
4.13.3.97	encodeBlis	67
4.13.3.98	decodeBlis	68
4.13.3.99	packSharedPseudocost	68
4.13.3.100	packSharedConstraints	68
4.13.3.101	unpackSharedConstraints	68
4.13.3.102	packSharedVariables	68
4.13.3.103	unpackSharedVariables	68
4.13.3.104	registerKnowledge	68
4.13.3.105	encode	68
4.13.3.106	decodeToSelf	68
4.13.3.107	packSharedKnowledge	68
4.13.3.108	unpackSharedKnowledge	69
4.13.4	Member Data Documentation	69
4.13.4.1	origLpSolver_	69
4.13.4.2	presolvedLpSolver_	69
4.13.4.3	lpSolver_	69
4.13.4.4	colMatrix_	69
4.13.4.5	varLB_	69
4.13.4.6	objSense_	69
4.13.4.7	numIntObjects_	69
4.13.4.8	inputVar_	70
4.13.4.9	incObjValue_	70
4.13.4.10	cutoff_	70
4.13.4.11	cutoffInc_	70
4.13.4.12	branchStrategy_	70
4.13.4.13	numObjects_	70
4.13.4.14	objects_	70
4.13.4.15	sharedObjectMark_	70

4.13.4.16 priority_	70
4.13.4.17 activeNode_	71
4.13.4.18 numStrong_	71
4.13.4.19 numBranchResolve_	71
4.13.4.20 numHeuristics_	71
4.13.4.21 heuristics_	71
4.13.4.22 cutStrategy_	71
4.13.4.23 numCutGenerators_	71
4.13.4.24 maxNumCons_	71
4.13.4.25 generators_	71
4.13.4.26 constraintPool_	72
4.13.4.27 oldConstraints_	72
4.13.4.28 oldConstraintsSize_	72
4.13.4.29 numOldConstraints_	72
4.13.4.30 conRandoms_	72
4.13.4.31 BlisPar_	72
4.13.4.32 blisMessageHandler_	72
4.13.4.33 blisMessages_	72
4.13.4.34 numNodes_	72
4.13.4.35 numIterations_	73
4.13.4.36 avelterations_	73
4.13.4.37 tempVarLBPos_	73
4.13.4.38 constraintPoolSend_	73
4.13.4.39 constraintPoolReceive_	73
4.13.4.40 isRoot_	73
4.13.4.41 boundingPass_	73
4.13.4.42 integerTol_	73
4.13.4.43 optimalRelGap_	73
4.13.4.44 optimalAbsGap_	74
4.13.4.45 currRelGap_	74
4.13.4.46 currAbsGap_	74
4.13.4.47 heurStrategy_	74
4.13.4.48 heurCallFrequency_	74
4.13.4.49 newCutPool_	74
4.13.4.50 leafToRootPath	74
4.14 BlisNodeDesc Class Reference	74
4.14.1 Detailed Description	75

4.14.2	Constructor & Destructor Documentation	75
4.14.2.1	BlisNodeDesc	75
4.14.2.2	BlisNodeDesc	76
4.14.2.3	~BlisNodeDesc	76
4.14.3	Member Function Documentation	76
4.14.3.1	setBasis	76
4.14.3.2	getBasis	76
4.14.3.3	setBranchedDir	76
4.14.3.4	getBranchedDir	76
4.14.3.5	setBranchedInd	76
4.14.3.6	getBranchedInd	76
4.14.3.7	setBranchedVal	77
4.14.3.8	getBranchedVal	77
4.14.3.9	encodeBlis	77
4.14.3.10	decodeBlis	77
4.14.3.11	encode	77
4.14.3.12	decode	77
4.15	BlisObjectInt Class Reference	77
4.15.1	Detailed Description	79
4.15.2	Constructor & Destructor Documentation	79
4.15.2.1	BlisObjectInt	79
4.15.2.2	BlisObjectInt	79
4.15.2.3	~BlisObjectInt	79
4.15.2.4	BlisObjectInt	79
4.15.3	Member Function Documentation	79
4.15.3.1	clone	79
4.15.3.2	operator=	80
4.15.3.3	infeasibility	80
4.15.3.4	feasibleRegion	80
4.15.3.5	createBranchObject	80
4.15.3.6	preferredNewFeasible	80
4.15.3.7	notPreferredNewFeasible	80
4.15.3.8	resetBounds	81
4.15.3.9	columnIndex	81
4.15.3.10	breakEven	81
4.15.3.11	setBreakEven	81
4.15.3.12	pseudocost	81

4.15.4	Member Data Documentation	81
4.15.4.1	columnIndex_	81
4.15.4.2	originalLower_	81
4.15.4.3	originalUpper_	81
4.15.4.4	breakEven_	82
4.15.4.5	pseudocost_	82
4.16	BlisParams Class Reference	82
4.16.1	Detailed Description	83
4.16.2	Member Enumeration Documentation	83
4.16.2.1	chrParams	83
4.16.2.2	intParams	84
4.16.2.3	dblParams	84
4.16.2.4	strParams	85
4.16.2.5	strArrayParams	85
4.16.3	Constructor & Destructor Documentation	85
4.16.3.1	BlisParams	85
4.16.4	Member Function Documentation	85
4.16.4.1	createKeywordList	85
4.16.4.2	setDefaultEntries	85
4.16.4.3	pack	85
4.16.4.4	unpack	85
4.17	BlisPresolve Class Reference	86
4.17.1	Detailed Description	86
4.18	BlisPseudocost Class Reference	86
4.18.1	Detailed Description	87
4.18.2	Constructor & Destructor Documentation	87
4.18.2.1	BlisPseudocost	87
4.18.2.2	BlisPseudocost	87
4.18.3	Member Function Documentation	88
4.18.3.1	setWeight	88
4.18.3.2	update	88
4.18.3.3	update	88
4.18.3.4	update	88
4.18.3.5	getUpCount	88
4.18.3.6	getUpCost	88
4.18.3.7	getDownCount	88
4.18.3.8	getDownCost	88

4.18.3.9	getScore	88
4.18.3.10	setScore	89
4.18.3.11	encodeTo	89
4.18.3.12	decodeFrom	89
4.18.3.13	encode	89
4.18.3.14	decode	89
4.19	BlisSolution Class Reference	89
4.19.1	Detailed Description	90
4.19.2	Constructor & Destructor Documentation	90
4.19.2.1	BlisSolution	90
4.19.2.2	BlisSolution	90
4.19.2.3	~BlisSolution	90
4.19.3	Member Function Documentation	90
4.19.3.1	print	90
4.19.3.2	encode	90
4.19.3.3	decode	91
4.20	BlisStrong Struct Reference	91
4.20.1	Detailed Description	91
4.21	BlisTreeNode Class Reference	91
4.21.1	Detailed Description	92
4.21.2	Constructor & Destructor Documentation	92
4.21.2.1	BlisTreeNode	92
4.21.2.2	BlisTreeNode	92
4.21.2.3	BlisTreeNode	93
4.21.2.4	~BlisTreeNode	93
4.21.3	Member Function Documentation	93
4.21.3.1	init	93
4.21.3.2	createNewTreeNode	93
4.21.3.3	process	93
4.21.3.4	branch	93
4.21.3.5	selectBranchObject	93
4.21.3.6	chooseBranchingObject	93
4.21.3.7	generateConstraints	94
4.21.3.8	callHeuristics	94
4.21.3.9	getViolatedConstraints	94
4.21.3.10	applyConstraints	94
4.21.3.11	reducedCostFix	94

4.21.3.12 encode	94
4.21.3.13 decode	94
4.22 BlisVariable Class Reference	94
4.22.1 Detailed Description	95
4.22.2 Member Function Documentation	95
4.22.2.1 encodeBlis	95
4.22.2.2 decodeBlis	95
4.22.2.3 encode	95
4.22.2.4 decode	96
Index	97

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 >	
ALPS_PS_STATS [external]	
AlpsEncoded [external]	
AlpsKnowledge [external]	
AlpsModel [external]	
BcpsModel [external]	
BlisModel	48
AlpsSolution [external]	
BcpsSolution [external]	
BlisSolution	89
AlpsSubTree [external]	
BcpsSubTree [external]	
AlpsTreeNode [external]	
BcpsTreeNode [external]	
BlisTreeNode	91
BlisTreeNode	91

BcpsObject[external]	
BcpsConstraint[external]	
BlisConstraint	38
BcpsVariable[external]	
BlisVariable	94
BlisObjectInt	77
BlisPseudocost	86
AlpsKnowledgeBroker[external]	
AlpsKnowledgeBrokerMPI[external]	
AlpsKnowledgeBrokerSerial[external]	
AlpsKnowledgePool[external]	
AlpsNodePool[external]	
AlpsSolutionPool[external]	
AlpsSubTreePool[external]	
BcpsObjectPool[external]	
BcpsConstraintPool[external]	
BcpsVariablePool[external]	
AlpsNodeDesc[external]	
BcpsNodeDesc[external]	
BlisNodeDesc	74
AlpsNodeSelection[external]	
AlpsNodeSelectionBest[external]	
AlpsNodeSelectionBreadth[external]	
AlpsNodeSelectionDepth[external]	
AlpsNodeSelectionEstimate[external]	
AlpsNodeSelectionHybrid[external]	
AlpsParameter[external]	
AlpsParameterSet[external]	
AlpsParams[external]	
BlisParams	82
AlpsPriorityQueue< T >[external]	
AlpsPriorityQueue< AlpsSubTree * >[external]	
AlpsPriorityQueue< AlpsTreeNode * >[external]	
AlpsStrLess[external]	
AlpsTimer[external]	
AlpsTreeSelection[external]	
AlpsTreeSelectionBest[external]	
AlpsTreeSelectionBreadth[external]	
AlpsTreeSelectionDepth[external]	
AlpsTreeSelectionEstimate[external]	
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 >	
BcpsBranchObject[external]	
BlisBranchObjectBilevel	17
BlisBranchObjectInt	17
BcpsBranchStrategy[external]	

BlisBranchStrategyBilevel	21
BlisBranchStrategyMaxInf	23
BlisBranchStrategyPseudo	25
BlisBranchStrategyRel	27
BlisBranchStrategyStrong	29
BcpsFieldListMod< T > [external]	
BcpsFieldListMod< double > [external]	
BcpsFieldListMod< int > [external]	
BcpsObjectListMod [external]	
std::bitset< Bits >	
BitVector128 [external]	
BlisConGenerator	30
BlisHeuristic	41
BlisHeurRound	45
BlisStrong	91
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]
- ClpLsq [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]
    CoinAbcTypeFactorization [external]
CoinAbcStack [external]
CoinAbcStatistics [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]
    CgILandP::NoBasisError [external]
    CgILandP::SimplexInterfaceError [external]
CoinExternalVectorFirstGreater_2< class, class, class > [external]
CoinExternalVectorFirstGreater_3< class, class, class, class > [external]
CoinExternalVectorFirstLess_2< class, class, class > [external]
CoinExternalVectorFirstLess_3< class, class, class, class > [external]
CoinFactorization [external]
CoinFileIOBase [external]
    CoinFileInput [external]
    CoinFileOutput [external]
CoinFirstAbsGreater_2< class, class > [external]
CoinFirstAbsGreater_3< class, class, class > [external]
CoinFirstAbsLess_2< class, class > [external]
CoinFirstAbsLess_3< class, class, class > [external]
CoinFirstGreater_2< class, class > [external]
CoinFirstGreater_3< class, class, class > [external]
CoinFirstLess_2< class, class > [external]
```

CoinFirstLess_3< class, class, class > [external]
 ClpHashValue::CoinHashLink [external]
 CoinLpIO::CoinHashLink [external]
 CoinMpsIO::CoinHashLink [external]
 CoinIndexedVector [external]
 CoinPartitionedVector [external]
 LAP::TabRow [external]
 CoinLpIO [external]
 CoinMessageHandler [external]
 MyMessageHandler [external]
 CoinMessages [external]
 AlpsMessage [external]
 BcpsMessage [external]

 BlisMessage 47
 CglMessage [external]
 ClpMessage [external]
 CoinMessage [external]
 LAP::LandPMessages [external]
 LAP::LapMessages [external]
 CoinModelHash [external]
 CoinModelHash2 [external]
 CoinModelHashLink [external]
 CoinModelInfo2 [external]
 CoinModelLink [external]
 CoinModelLinkedList [external]
 CoinModelTriple [external]
 CoinMpsCardReader [external]
 CoinMpsIO [external]
 CoinOneMessage [external]
 CoinOtherFactorization [external]
 CoinDenseFactorization [external]
 CoinOsIFactorization [external]
 CoinSimpFactorization [external]
 CoinPackedMatrix [external]
 CoinPackedVectorBase [external]
 CoinPackedVector [external]
 CoinShallowPackedVector [external]
 CoinPair< S, T > [external]
 CoinParam [external]
 CoinPrePostsolveMatrix [external]
 CoinPostsolveMatrix [external]
 CoinPresolveMatrix [external]
 CoinPresolveAction [external]
 do_tighten_action [external]
 doubleton_action [external]
 drop_empty_cols_action [external]
 drop_empty_rows_action [external]
 drop_zero_coefficients_action [external]
 dupcol_action [external]
 duprow3_action [external]
 duprow_action [external]
 forcing_constraint_action [external]
 gubrow_action [external]
 implied_free_action [external]

```

isolated_constraint_action[external]
make_fixed_action[external]
remove_dual_action[external]
remove_fixed_action[external]
slack_doubleton_action[external]
slack_singleton_action[external]
subst_constraint_action[external]
tripleton_action[external]
twoxtwo_action[external]
useless_constraint_action[external]
CoinPresolveMonitor[external]
CoinRational[external]
CoinRelFltEq[external]
CoinSearchTreeBase[external]
    CoinSearchTree< class >[external]
CoinSearchTreeCompareBest[external]
CoinSearchTreeCompareBreadth[external]
CoinSearchTreeCompareDepth[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
std::vector< T >::const_iterator
std::forward_list< T >::const_iterator
std::unordered_multimap< K, T >::const_iterator
std::list< T >::const_iterator
std::unordered_multiset< K >::const_iterator
std::multiset< K >::const_iterator
std::wstring::const_iterator

```

```

OsiCuts::const_iterator[external]
std::multimap< K, T >::const_iterator
std::basic_string< Char >::const_iterator
std::string::const_iterator
std::deque< T >::const_iterator
std::map< K, T >::const_iterator
std::unordered_map< K, T >::const_iterator
std::set< K >::const_iterator
std::unordered_set< K >::const_iterator
std::wstring::const_reverse_iterator
std::deque< T >::const_reverse_iterator
std::unordered_set< K >::const_reverse_iterator
std::vector< T >::const_reverse_iterator
std::basic_string< Char >::const_reverse_iterator
std::map< K, T >::const_reverse_iterator
std::string::const_reverse_iterator
std::unordered_map< K, T >::const_reverse_iterator
std::list< T >::const_reverse_iterator
std::forward_list< T >::const_reverse_iterator
std::multimap< K, T >::const_reverse_iterator
std::unordered_multimap< K, T >::const_reverse_iterator
std::set< K >::const_reverse_iterator
std::multiset< K >::const_reverse_iterator
std::unordered_multiset< K >::const_reverse_iterator
cut[external]
cut_list[external]
cutParams[external]
LAP::Cuts[external]
cycle[external]
cycle_list[external]
DeletePtrObject[external]
std::deque< T >
std::deque< int >
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_istreamstream< char >
      basic_istreamstream< 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_istreamstream
        std::istreamstream
        std::wistreamstream
      std::istream
      std::wistream
    std::basic_ostream
      basic_ostream< char >
      basic_ostream< wchar_t >
      basic_ofstream< char >
      basic_ofstream< wchar_t >
      basic_ostreamstream< char >
```

```

        basic_ostringstream< wchar_t >
        std::basic_istream
        std::basic_ofstream
            std::ofstream
            std::wofstream
        std::basic_ostringstream
            std::ostringstream
            std::wostringstream
        std::ostream
        std::wostream
    std::ios
    std::wios
std::unordered_set< K >::iterator
std::deque< T >::iterator
OsiCuts::iterator [external]
std::unordered_multiset< K >::iterator
std::set< K >::iterator
std::vector< T >::iterator
std::wstring::iterator
std::list< T >::iterator
std::unordered_map< K, T >::iterator
std::map< K, T >::iterator
std::unordered_multimap< K, T >::iterator
std::multimap< K, T >::iterator
std::forward_list< T >::iterator
std::string::iterator
std::basic_string< Char >::iterator
std::multiset< K >::iterator
std::list< T >
log_var [external]
std::map< K, T >
std::map< AlpsKnowledgeType, AlpsKnowledgePool * >
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]	
BlisPresolve	86
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]	
ClpSimplexOther::parametricsData[external]	
parity_ilp[external]	
AbcSimplexPrimal::pivotStruct[external]	
pool_cut[external]	
pool_cut_list[external]	
presolvehlink[external]	
std::priority_queue< T >	
std::queue< T >	
Coin::ReferencedObject[external]	
std::basic_string< Char >::reverse_iterator	
std::multiset< K >::reverse_iterator	
std::forward_list< T >::reverse_iterator	
std::set< K >::reverse_iterator	
std::unordered_map< K, T >::reverse_iterator	
std::unordered_multimap< K, T >::reverse_iterator	
std::deque< T >::reverse_iterator	
std::wstring::reverse_iterator	
std::string::reverse_iterator	
std::map< K, T >::reverse_iterator	
std::unordered_multiset< K >::reverse_iterator	
std::list< T >::reverse_iterator	
std::vector< T >::reverse_iterator	
std::unordered_set< K >::reverse_iterator	
std::multimap< K, T >::reverse_iterator	
scatterStruct[external]	
select_cut[external]	
separation_graph[external]	
std::set< K >	
short_path_node[external]	
std::smart_ptr< T >	
Coin::SmartPtr< T >[external]	
std::stack< T >	
symrec[external]	
std::system_error	
OsiUnitTest::TestOutcome[external]	
OsiUnitTest::TestOutcomes[external]	
std::thread	

TotalWorkload[external]
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< AlpsTreeNode * >
std::vector< BcpsConstraint * >
std::vector< BcpsVariable * >
std::vector< ColumnSelectionStrategy >
std::vector< double >
std::vector< int >
std::vector< RowSelectionStrategy >
std::vector< std::pair< std::string, AlpsParameter > >
std::vector< std::string >
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:

BlisBranchObjectBilevel	17
BlisBranchObjectInt	17
BlisBranchStrategyBilevel	
This class implements maximum infeasibility branching	21
BlisBranchStrategyMaxInf	
This class implements maximum infeasibility branching	23
BlisBranchStrategyPseudo	
Blis branching strategy	25
BlisBranchStrategyRel	
Blis branching strategy	27
BlisBranchStrategyStrong	
This class implements strong branching	29
BlisConGenerator	
Interface between Blis and Cut Generation Library	30
BlisConstraint	38
BlisHeuristic	
Heuristic base class	41
BlisHeurRound	
Rounding Heuristic	45
BlisMessage	47
BlisModel	48
BlisNodeDesc	74
BlisObjectInt	77
BlisParams	82
BlisPresolve	
A interface to Osi/Coin Presolve	86
BlisPseudocost	86
BlisSolution	
This class contains the solutions generated by the LP solver (either primal or dual)	89
BlisStrong	91
BlisTreeNode	
This is the class in which we are finally able to concretely define the bounding procedure	91
BlisVariable	94

Chapter 3

File Index

3.1 File List

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

Blis.h	??
BlisBranchObjectBilevel.h	??
BlisBranchObjectInt.h	??
BlisBranchStrategyBilevel.h	??
BlisBranchStrategyMaxInf.h	??
BlisBranchStrategyPseudo.h	??
BlisBranchStrategyRel.h	??
BlisBranchStrategyStrong.h	??
BlisConfig.h	??
BlisConGenerator.h	??
BlisConstraint.h	??
BlisHelp.h	??
BlisHeuristic.h	??
BlisHeurRound.h	??
BlisLicense.h	??
BlisMessage.h	??
BlisModel.h	??
BlisNodeDesc.h	??
BlisObjectInt.h	??
BlisParams.h	??
BlisPresolve.h	??
BlisPseudo.h	??
BlisSolution.h	??
BlisSubTree.h	??
BlisTreeNode.h	??
BlisVariable.h	??
config_blis_default.h	??
config_default.h	??

Chapter 4

Class Documentation

4.1 BlisBranchObjectBilevel Class Reference

Inheritance diagram for BlisBranchObjectBilevel:

4.2 BlisBranchObjectInt Class Reference

Inheritance diagram for BlisBranchObjectInt:

Collaboration diagram for BlisBranchObjectInt:

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;.

4.2.1 Detailed Description

Definition at line 38 of file BlisBranchObjectInt.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 BlisBranchObjectInt::BlisBranchObjectInt () `[inline]`

Default constructor.

Definition at line 53 of file BlisBranchObjectInt.h.

4.2.2.2 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.

4.2.2.3 `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.

4.2.2.4 `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.

4.2.2.5 `BlisBranchObjectInt::BlisBranchObjectInt (const BlisBranchObjectInt &)`

Copy constructor.

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

Destructor.

Definition at line 141 of file BlisBranchObjectInt.h.

4.2.3 Member Function Documentation

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

Assignment operator.

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

Clone.

Implements **BcpsBranchObject**.

Definition at line 136 of file BlisBranchObjectInt.h.

4.2.3.3 `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.

Implements **BcpsBranchObject**.

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

Print something about branch - only if log level high.

Reimplemented from **BcpsBranchObject**.

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

Get down arm bounds.

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

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

Get upper arm bounds.

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

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

Pack Blis portion to an encoded object.

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

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

Unpack Blis portion from an encoded object.

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

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

Pack to an encoded object.

Reimplemented from **BcpsBranchObject**.

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

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

Unpack a branching object from an encoded object.

Reimplemented from **BcpsBranchObject**.

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

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

- `BlisBranchObjectInt.h`

4.3 BlisBranchStrategyBilevel Class Reference

This class implements maximum infeasibility branching.

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

Inheritance diagram for BlisBranchStrategyBilevel:

Collaboration diagram for BlisBranchStrategyBilevel:

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.

4.3.1 Detailed Description

This class implements maximum infeasibility branching.

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

4.3.2 Constructor & Destructor Documentation

4.3.2.1 [BlisBranchStrategyBilevel::BlisBranchStrategyBilevel](#) () [[inline](#)]

Bilevel Constructor.

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

4.3.2.2 [BlisBranchStrategyBilevel::BlisBranchStrategyBilevel](#) ([BlisModel](#) * *model*) [[inline](#)]

Bilevel Constructor.

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

4.3.2.3 virtual [BlisBranchStrategyBilevel::~~BlisBranchStrategyBilevel](#) () [[inline](#)], [[virtual](#)]

Destructor.

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

4.3.2.4 BlisBranchStrategyBilevel::BlisBranchStrategyBilevel (const BlisBranchStrategyBilevel &)

Copy constructor.

4.3.3 Member Function Documentation

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

Clone a branching strategy.

Implements **BcpsBranchStrategy**.

Definition at line 58 of file BlisBranchStrategyBilevel.h.

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

Create a set of candidate branching objects.

Reimplemented from **BcpsBranchStrategy**.

4.3.3.3 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).

Implements **BcpsBranchStrategy**.

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

- BlisBranchStrategyBilevel.h

4.4 BlisBranchStrategyMaxInf Class Reference

This class implements maximum infeasibility branching.

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

Inheritance diagram for BlisBranchStrategyMaxInf:

Collaboration diagram for BlisBranchStrategyMaxInf:

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.

4.4.1 Detailed Description

This class implements maximum infeasibility branching.

Definition at line 32 of file BlisBranchStrategyMaxInf.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 **BlisBranchStrategyMaxInf::BlisBranchStrategyMaxInf** () [inline]

MaxInf Constructor.

Definition at line 42 of file BlisBranchStrategyMaxInf.h.

4.4.2.2 **BlisBranchStrategyMaxInf::BlisBranchStrategyMaxInf** (**BlisModel** * *model*) [inline]

MaxInf Constructor.

Definition at line 47 of file BlisBranchStrategyMaxInf.h.

4.4.2.3 **virtual BlisBranchStrategyMaxInf::~~BlisBranchStrategyMaxInf** () [inline],[virtual]

Destructor.

Definition at line 52 of file BlisBranchStrategyMaxInf.h.

4.4.2.4 **BlisBranchStrategyMaxInf::BlisBranchStrategyMaxInf** (const **BlisBranchStrategyMaxInf** &)

Copy constructor.

4.4.3 Member Function Documentation

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

Clone a branching strategy.

Implements **BcpsBranchStrategy**.

Definition at line 58 of file BlisBranchStrategyMaxInf.h.

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

Create a set of candidate branching objects.

Reimplemented from **BcpsBranchStrategy**.

4.4.3.3 `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).

Implements **BcpsBranchStrategy**.

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

- BlisBranchStrategyMaxInf.h

4.5 BlisBranchStrategyPseudo Class Reference

Blis branching strategy.

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

Inheritance diagram for BlisBranchStrategyPseudo:

Collaboration diagram for BlisBranchStrategyPseudo:

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.

4.5.1 Detailed Description

Blis branching strategy.

This class implements pseudocost branching.

Definition at line 40 of file BlisBranchStrategyPseudo.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 `BlisBranchStrategyPseudo::BlisBranchStrategyPseudo ()` `[inline]`

Default Constructor.

Definition at line 51 of file BlisBranchStrategyPseudo.h.

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

Useful Constructor.

Definition at line 57 of file BlisBranchStrategyPseudo.h.

4.5.2.3 `virtual BlisBranchStrategyPseudo::~~BlisBranchStrategyPseudo ()` `[inline],[virtual]`

Destructor.

Definition at line 64 of file BlisBranchStrategyPseudo.h.

4.5.2.4 `BlisBranchStrategyPseudo::BlisBranchStrategyPseudo (const BlisBranchStrategyPseudo &)`

Copy constructor.

4.5.3 Member Function Documentation

4.5.3.1 `void BlisBranchStrategyPseudo::setReliability (int rel)` `[inline]`

Set reliability.

Definition at line 70 of file BlisBranchStrategyPseudo.h.

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

Clone a branching strategy.

Implements **BcpsBranchStrategy**.

Definition at line 73 of file BlisBranchStrategyPseudo.h.

4.5.3.3 `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).

Implements **BcpsBranchStrategy**.

4.5.3.4 `int BlisBranchStrategyPseudo::createCandBranchObjects (int numPassesLeft, double ub) [virtual]`

Create a set of candidate branching objects.

Reimplemented from **BcpsBranchStrategy**.

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

- BlisBranchStrategyPseudo.h

4.6 BlisBranchStrategyRel Class Reference

Blis branching strategy.

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

Inheritance diagram for BlisBranchStrategyRel:

Collaboration diagram for BlisBranchStrategyRel:

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.

4.6.1 Detailed Description

Blis branching strategy.

This class implements reliability branching.

Definition at line 40 of file BlisBranchStrategyRel.h.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 `BlisBranchStrategyRel::BlisBranchStrategyRel ()` `[inline]`

Default Constructor.

Definition at line 51 of file `BlisBranchStrategyRel.h`.

4.6.2.2 `BlisBranchStrategyRel::BlisBranchStrategyRel (BlisModel * model, int rel)` `[inline]`

Useful Constructor.

Definition at line 57 of file `BlisBranchStrategyRel.h`.

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

Destructor.

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

4.6.2.4 `BlisBranchStrategyRel::BlisBranchStrategyRel (const BlisBranchStrategyRel &)`

Copy constructor.

4.6.3 Member Function Documentation

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

Set reliability.

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

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

Clone a branching strategy.

Implements **BcpsBranchStrategy**.

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

4.6.3.3 `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 *bestSorFar* is NULL, then always return branching direction(1 or -1).

Implements **BcpsBranchStrategy**.

4.6.3.4 `int BlisBranchStrategyRel::createCandBranchObjects (int numPassesLeft, double ub)` `[virtual]`

Create a set of candidate branching objects.

Reimplemented from **BcpsBranchStrategy**.

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

- BlisBranchStrategyRel.h

4.7 BlisBranchStrategyStrong Class Reference

This class implements strong branching.

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

Inheritance diagram for BlisBranchStrategyStrong:

Collaboration diagram for BlisBranchStrategyStrong:

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.

4.7.1 Detailed Description

This class implements strong branching.

Definition at line 57 of file BlisBranchStrategyStrong.h.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 [BlisBranchStrategyStrong::BlisBranchStrategyStrong](#) () [[inline](#)]

Strong Constructor.

Definition at line 67 of file BlisBranchStrategyStrong.h.

4.7.2.2 [BlisBranchStrategyStrong::BlisBranchStrategyStrong](#) ([BlisModel](#) * model) [[inline](#)]

Strong Constructor.

Definition at line 72 of file BlisBranchStrategyStrong.h.

4.7.2.3 `virtual BlisBranchStrategyStrong::~~BlisBranchStrategyStrong () [inline],[virtual]`

Destructor.

Definition at line 78 of file `BlisBranchStrategyStrong.h`.

4.7.2.4 `BlisBranchStrategyStrong::BlisBranchStrategyStrong (const BlisBranchStrategyStrong &)`

Copy constructor.

4.7.3 Member Function Documentation

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

Clone a branching strategy.

Implements **BcpsBranchStrategy**.

Definition at line 84 of file `BlisBranchStrategyStrong.h`.

4.7.3.2 `virtual int BlisBranchStrategyStrong::createCandBranchObjects (int numPassesLeft, double ub) [virtual]`

Create a set of candidate branching objects.

Reimplemented from **BcpsBranchStrategy**.

4.7.3.3 `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).

Implements **BcpsBranchStrategy**.

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

- `BlisBranchStrategyStrong.h`

4.8 BlisConGenerator Class Reference

Interface between Blis and Cut Generation Library.

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

Collaboration diagram for `BlisConGenerator`:

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.*

4.8.1 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`.

4.8.2 Constructor & Destructor Documentation

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

Default constructor.

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

4.8.2.2 `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.

4.8.2.3 `BlisConGenerator::BlisConGenerator (const BlisConGenerator &)`

Copy constructor.

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

Destructor.

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

4.8.3 Member Function Documentation

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

Assignment operator.

4.8.3.2 `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).

4.8.3.3 **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`.

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

return name of generator.

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

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

return name of generator.

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

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

Set the con generation strategy.

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

4.8.3.7 **BlisCutStrategy** **BlisConGenerator::strategy () const** `[inline]`

Get the con generation interval.

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

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

Set the con generation strategy.

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

4.8.3.9 **int** **BlisConGenerator::cutGenerationFreq () const** `[inline]`

Get the con generation interval.

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

4.8.3.10 **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`.

4.8.3.11 `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.

4.8.3.12 `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.

4.8.3.13 `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.

4.8.3.14 `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.

4.8.3.15 `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.

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

Get the **CglCutGenerator** bound to this [BlisConGenerator](#).

Definition at line 231 of file BlisConGenerator.h.

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

Get number of generated cons.

Definition at line 234 of file BlisConGenerator.h.

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

Increase the number of generated cons.

Definition at line 237 of file BlisConGenerator.h.

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

Get number of used cons.

Definition at line 240 of file BlisConGenerator.h.

4.8.3.20 void BlisConGenerator::addNumConsUsed (int *n*) [inline]

Increase the number of generated cons.

Definition at line 243 of file BlisConGenerator.h.

4.8.3.21 double BlisConGenerator::time () const [inline]

Cpu time used.

Definition at line 246 of file BlisConGenerator.h.

4.8.3.22 void BlisConGenerator::addTime (double *t*) [inline]

Increase Cpu time used.

Definition at line 249 of file BlisConGenerator.h.

4.8.3.23 int BlisConGenerator::calls () const [inline]

Number called.

Definition at line 252 of file BlisConGenerator.h.

4.8.3.24 void BlisConGenerator::addCalls (int *n* = 1) [inline]

Increase the number of called.

Definition at line 255 of file BlisConGenerator.h.

4.8.3.25 int BlisConGenerator::noConsCalls () const [inline]

Number called and no cons found.

Definition at line 258 of file BlisConGenerator.h.

4.8.3.26 void BlisConGenerator::addNoConsCalls (int *n* = 1) [inline]

Increase the number of no cons called.

Definition at line 261 of file BlisConGenerator.h.

4.8.4 Member Data Documentation

4.8.4.1 BlisModel* BlisConGenerator::model_ [protected]

The client model.

Definition at line 62 of file BlisConGenerator.h.

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

The **CglCutGenerator** object.

Definition at line 65 of file `BlisConGenerator.h`.

4.8.4.3 `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`.

4.8.4.4 `int` `BlisConGenerator::cutGenerationFrequency_` [protected]

The frequency of calls to the cut generator.

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

4.8.4.5 `std::string` `BlisConGenerator::name_` [protected]

Name of generator.

Definition at line 83 of file `BlisConGenerator.h`.

4.8.4.6 `bool` `BlisConGenerator::normal_` [protected]

Whether to call the generator in the normal place.

Definition at line 86 of file `BlisConGenerator.h`.

4.8.4.7 `bool` `BlisConGenerator::atSolution_` [protected]

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

Definition at line 89 of file `BlisConGenerator.h`.

4.8.4.8 `bool` `BlisConGenerator::whenInfeasible_` [protected]

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

Definition at line 93 of file `BlisConGenerator.h`.

4.8.4.9 `int` `BlisConGenerator::numConsGenerated_` [protected]

Number of cons generated.

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

4.8.4.10 `int BlisConGenerator::numConsUsed_` [protected]

Number of cons used.

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

4.8.4.11 `double BlisConGenerator::time_` [protected]

Used CPU/User time.

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

4.8.4.12 `int BlisConGenerator::calls_` [protected]

The times of calling this generator.

Definition at line 109 of file `BlisConGenerator.h`.

4.8.4.13 `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`

4.9 BlisConstraint Class Reference

Inheritance diagram for `BlisConstraint`:

Collaboration diagram for `BlisConstraint`:

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.

4.9.1 Detailed Description

Definition at line 33 of file BlisConstraint.h.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 BlisConstraint::BlisConstraint ()

Default constructor.

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

Useful constructor.

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

Useful constructor.

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

Useful constructor.

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

Destructor.

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

Copy constructor.

4.9.3 Member Function Documentation

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

Pack Blis part into an encoded object.

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

Unpack Blis part from a encode object.

4.9.3.3 `OsiRowCut* BlisConstraint::createOsiRowCut ()`

Create a **OsiRowCut** based on this constraint.

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

Compute a hash key.

Reimplemented from **BcpsObject**.

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

Check if violates a given lp solution.

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

Pack into a encode object.

Reimplemented from **BcpsObject**.

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

Decode a constraint from an encoded object.

Reimplemented from **BcpsObject**.

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

- **BlisConstraint.h**

4.10 BlisHeuristic Class Reference

Heuristic base class.

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

Inheritance diagram for **BlisHeuristic**:

Collaboration diagram for **BlisHeuristic**:

Public Member Functions

- **BlisHeuristic** ()
Default Constructor.
- **BlisHeuristic** (**BlisModel** *model, const char *name, **BlisHeurStrategy** strategy, int heurCallFrequency)
Useful constructor.
- virtual **~BlisHeuristic** ()
Destructor.
- **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.

4.10.1 Detailed Description

Heuristic base class.

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

4.10.2 Constructor & Destructor Documentation

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

Default Constructor.

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

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

Useful constructor.

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

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

Distructor.

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

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

Copy constructor.

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

4.10.3 Member Function Documentation

4.10.3.1 `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`.

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

Get/set strategy.

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

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

Get/set call frequency.

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

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

Clone a heuristic.

Reimplemented in [BlisHeurRound](#).

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

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

return name of generator.

Definition at line 177 of file `BlisHeuristic.h`.

4.10.3.6 void BlisHeuristic::addNumSolutions (int *num* = 1) [inline]

Record number of solutions found.

Definition at line 180 of file BlisHeuristic.h.

4.10.3.7 int BlisHeuristic::numSolutions () const [inline]

Number of solutions found.

Definition at line 183 of file BlisHeuristic.h.

4.10.3.8 void BlisHeuristic::addTime (double *t* = 0.0) [inline]

Record Cpu time used.

Definition at line 186 of file BlisHeuristic.h.

4.10.3.9 double BlisHeuristic::time () const [inline]

Cpu time used.

Definition at line 189 of file BlisHeuristic.h.

4.10.3.10 void BlisHeuristic::addCalls (int *c* = 1) [inline]

Record number of times called.

Definition at line 192 of file BlisHeuristic.h.

4.10.3.11 int BlisHeuristic::calls () const [inline]

Number of times called.

Definition at line 195 of file BlisHeuristic.h.

4.10.3.12 int BlisHeuristic::noSolCalls () const [inline]

Number called and no cons found.

Definition at line 198 of file BlisHeuristic.h.

4.10.3.13 void BlisHeuristic::addNoSolCalls (int *n* = 1) [inline]

Increase the number of no cons called.

Definition at line 201 of file BlisHeuristic.h.

4.10.4 Member Data Documentation

4.10.4.1 `BlisHeuristic::strategy_` [protected]

When to call findSolution() routine.

BlisHeuristicStrategyNone: disable BlisHeuristicStrategyRoot: just root BlisHeuristicStrategyAuto: automatically decided by BLIS
BlisHeuristicStrategyPeriodic: every 't' nodes BlisHeuristicStrategyBeforeRoot: before solving first LP

Definition at line 70 of file BlisHeuristic.h.

4.10.4.2 `int BlisHeuristic::numSolutions_` [protected]

Number of solutions found.

Definition at line 76 of file BlisHeuristic.h.

4.10.4.3 `double BlisHeuristic::time_` [protected]

Used CPU/User time.

Definition at line 79 of file BlisHeuristic.h.

4.10.4.4 `int BlisHeuristic::calls_` [protected]

The times of calling this heuristic.

Definition at line 82 of file BlisHeuristic.h.

4.10.4.5 `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

4.11 BlisHeurRound Class Reference

Rounding Heuristic.

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

Inheritance diagram for BlisHeurRound:

Collaboration diagram for BlisHeurRound:

Public Member Functions

- [BlisHeurRound](#) ()
Default Constructor.
- [BlisHeurRound](#) ([BlisModel](#) *model, const char *name, BlisHeuristicStrategy 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.

4.11.1 Detailed Description

Rounding Heuristic.

Definition at line 44 of file [BlisHeurRound.h](#).

4.11.2 Constructor & Destructor Documentation

4.11.2.1 [BlisHeurRound::BlisHeurRound](#) () [\[inline\]](#)

Default Constructor.

Definition at line 61 of file [BlisHeurRound.h](#).

4.11.2.2 [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](#).

4.11.2.3 [BlisHeurRound::~~BlisHeurRound](#) () [\[inline\]](#)

Destructor.

Definition at line 73 of file [BlisHeurRound.h](#).

4.11.2.4 `BlisHeurRound::BlisHeurRound (const BlisHeurRound &)`

Copy constructor.

4.11.3 Member Function Documentation

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

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

Reimplemented from [BlisHeuristic](#).

4.11.4 Member Data Documentation

4.11.4.1 `CoinPackedMatrix BlisHeurRound::matrix_ [protected]`

Column majored matrix.

Definition at line 51 of file `BlisHeurRound.h`.

4.11.4.2 `CoinPackedMatrix BlisHeurRound::matrixByRow_ [protected]`

Row majored matrix.

Definition at line 54 of file `BlisHeurRound.h`.

4.11.4.3 `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`

4.12 BlisMessage Class Reference

Inheritance diagram for `BlisMessage`:

Collaboration diagram for `BlisMessage`:

Public Member Functions

Constructors etc

- [BlisMessage](#) (`Language language=us_en`)

Constructor.

4.12.1 Detailed Description

Definition at line 58 of file BlisMessage.h.

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

- BlisMessage.h

4.13 BlisModel Class Reference

Inheritance diagram for BlisModel:

Collaboration diagram for BlisModel:

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 &ostream) 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 thresheld to be considered as a dense constraint.
- void **setDenseConCutoff** (int cutoff)
Set the thresheld 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** * [packSharedKnowlege](#) ()

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 Strategys

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 pseudocost.*
- 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 lp 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 [aveIterations_](#)
- 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_](#)
Temporary storage for var/con indices.

4.13.1 Detailed Description

Definition at line 69 of file `BlisModel.h`.

4.13.2 Constructor & Destructor Documentation

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

Default construtor.

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

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

Destructor.

4.13.3 Member Function Documentation

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

Create variables and constraints.

4.13.3.2 `void BlisModel::gutsOfDestructor ()`

Actual destructor.

4.13.3.3 `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_

Reimplemented from **AlpsModel**.

4.13.3.4 `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.

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

Read in Alps, Blis parameters.

Reimplemented from **AlpsModel**.

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

Write out parameters.

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

For parallel code, only the master calls this function.

Create the root node based on model.

Reimplemented from **AlpsModel**.

4.13.3.8 `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

Reimplemented from **AlpsModel**.

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

Preprocessing the model.

Reimplemented from **AlpsModel**.

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

Postprocessing the searching results.

Reimplemented from **AlpsModel**.

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

Set lp solver.

Definition at line 440 of file BlisModel.h.

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

Get lp solver.

Definition at line 443 of file BlisModel.h.

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

Get lp solver.

Definition at line 446 of file BlisModel.h.

4.13.3.14 `bool BlisModel::resolve ()`

Resolving a lp.

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

Set active node.

Definition at line 452 of file BlisModel.h.

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

Set the solution estimate of the active node.

Definition at line 455 of file BlisModel.h.

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

Get number of strong branchings.

Definition at line 458 of file BlisModel.h.

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

Add num to number of strong branchings.

Definition at line 461 of file BlisModel.h.

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

Get the maximum number of resolve during branching.

Definition at line 464 of file BlisModel.h.

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

Set the maximum number of resolve during branching.

Definition at line 467 of file BlisModel.h.

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

Get objective coefficients.

Definition at line 474 of file BlisModel.h.

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

Get column lower bound.

Definition at line 477 of file BlisModel.h.

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

Get column upper bound.

Definition at line 480 of file BlisModel.h.

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

Get number of columns.

Definition at line 483 of file BlisModel.h.

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

Get number of rows.

Definition at line 486 of file BlisModel.h.

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

Get variable bounds array.

Definition at line 489 of file BlisModel.h.

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

Get original constraint bounds array.

Definition at line 493 of file BlisModel.h.

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

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

Definition at line 497 of file BlisModel.h.

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

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

Definition at line 501 of file BlisModel.h.

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

Temporary storage.

Definition at line 505 of file BlisModel.h.

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

Get current objective function value.

Definition at line 515 of file BlisModel.h.

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

Get active lp solution.

Definition at line 518 of file BlisModel.h.

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

Get number of solutions.

Definition at line 525 of file BlisModel.h.

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

Get number of heuristic solutions.

Definition at line 528 of file BlisModel.h.

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

Return best ip solution found so far.

Definition at line 531 of file BlisModel.h.

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

Record a new incumbent solution and update objectiveValue.

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

Get cut off value.

Definition at line 537 of file BlisModel.h.

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

Set cut off value.

Definition at line 540 of file BlisModel.h.

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

Test if a solution found by heuristic is feasible.

4.13.3.40 `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*).

4.13.3.41 `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.

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

Get the current branching strategy.

Definition at line 587 of file BlisModel.h.

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

Set the branching strategy.

Definition at line 591 of file BlisModel.h.

4.13.3.44 `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.

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

Get the number of objects.

Definition at line 609 of file BlisModel.h.

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

Set the number of objects.

Definition at line 612 of file BlisModel.h.

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

Get the array of objects.

Definition at line 615 of file BlisModel.h.

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

Get the specified object.

Definition at line 618 of file BlisModel.h.

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

Mark object to be shared.

Definition at line 621 of file BlisModel.h.

4.13.3.50 `void BlisModel::clearSharedObjectMark () [inline]`

Clear all the share mark.

Definition at line 624 of file BlisModel.h.

4.13.3.51 `void BlisModel::deleteObjects ()`

Delete all object information.

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

Add in object information.

Objects are cloned; the owner can delete the originals.

4.13.3.53 `void BlisModel::createIntegerObjects (bool startAgain)`

Identify integer variable.

4.13.3.54 `int* BlisModel::getIntObjIndices () const [inline]`

Get integers' object indices.

Definition at line 642 of file BlisModel.h.

4.13.3.55 `int BlisModel::getNumIntObjects () const [inline]`

Get number of integers.

Definition at line 645 of file BlisModel.h.

4.13.3.56 `int* BlisModel::getIntColIndices () const [inline]`

Get integers' column indices.

Definition at line 648 of file BlisModel.h.

4.13.3.57 `bool BlisModel::checkInteger (double value) const [inline]`

Check if a value is integer.

Definition at line 651 of file BlisModel.h.

4.13.3.58 `void BlisModel::addHeuristic (BlisHeuristic * heur)`

Add a heuristic.

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

Get a specific heuristic.

Definition at line 672 of file BlisModel.h.

4.13.3.60 `int BlisModel::numHeuristics () const [inline]`

Get the number of heuristics.

Definition at line 675 of file BlisModel.h.

4.13.3.61 `void BlisModel::addCutGenerator (BlisConGenerator * generator)`

Add a Blis cut generator.

4.13.3.62 `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.

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

Get a specific cut generator.

Definition at line 694 of file `BlisModel.h`.

4.13.3.64 **int** **BlisModel::numCutGenerators** () **const** [inline]

Get the number of cut generators.

Definition at line 697 of file `BlisModel.h`.

4.13.3.65 **int** **BlisModel::getMaxNumCons** () **const** [inline]

Get the max number of cuts can be generated.

Definition at line 700 of file `BlisModel.h`.

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

Set the max number of cuts can be generated.

Definition at line 703 of file `BlisModel.h`.

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

Access constraint pool.

Definition at line 706 of file `BlisModel.h`.

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

Access receive constraint pool.

Definition at line 709 of file `BlisModel.h`.

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

Access send constraint pool.

Definition at line 713 of file `BlisModel.h`.

4.13.3.70 **int** **BlisModel::getNumOldConstraints** () **const** [inline]

Get number of old constraints.

Definition at line 717 of file `BlisModel.h`.

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

Set number of old constraints.

Definition at line 720 of file `BlisModel.h`.

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

Get max number of old constraints.

Definition at line 723 of file BlisModel.h.

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

Set max number of old constraints.

Definition at line 726 of file BlisModel.h.

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

Access old constraints.

Definition at line 729 of file BlisModel.h.

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

set old constraints.

Definition at line 732 of file BlisModel.h.

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

Set max number of old constraints.

Definition at line 735 of file BlisModel.h.

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

Query constraint generation strategy.

Definition at line 742 of file BlisModel.h.

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

Set constraint generation strategy.

Definition at line 747 of file BlisModel.h.

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

Query constraint generation frequency.

Definition at line 750 of file BlisModel.h.

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

Set constraint generation frequency.

Definition at line 753 of file BlisModel.h.

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

Get the threshold to be considered as a dense constraint.

Definition at line 756 of file `BlisModel.h`.

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

Set the threshold to be considered as a dense constraint.

Definition at line 759 of file `BlisModel.h`.

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

Get randoms for check parallel constraints.

Definition at line 762 of file `BlisModel.h`.

4.13.3.84 `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

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

Priorities.

Definition at line 787 of file `BlisModel.h`.

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

Log of specific models.

Reimplemented from **AlpsModel**.

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

Get how many Nodes it took to solve the problem.

Definition at line 808 of file `BlisModel.h`.

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

Get how many iterations it took to solve the problem.

Definition at line 811 of file `BlisModel.h`.

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

Get the average iterations it took to solve a lp.

Definition at line 814 of file `BlisModel.h`.

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

Increment node count.

Definition at line 817 of file `BlisModel.h`.

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

Increment iteration count.

Definition at line 820 of file `BlisModel.h`.

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

Get the message handler.

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

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

Return messages.

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

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

Access parameters.

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

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

Node log.

Reimplemented from **AlpsModel**.

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

Return true, if all nodes can be fathomed.

Reimplemented from **AlpsModel**.

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

Pack Blis portion of the model into an encoded object.

4.13.3.98 **AlpsReturnStatus** **BlisModel::decodeBlis** (**AlpsEncoded** & *encoded*) [protected]

Unpack Blis portion of the model from an encoded object.

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

Retrieve and pack shared pseudocost.

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

Retrieve and pack shared constraints.

4.13.3.101 void **BlisModel::unpackSharedConstraints** (**AlpsEncoded** & *encoded*) [protected]

Unpack and store shared constraints.

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

Retrieve and pack shared variables.

4.13.3.103 void **BlisModel::unpackSharedVariables** (**AlpsEncoded** & *encoded*) [protected]

Unpack and store shared variables.

4.13.3.104 virtual void **BlisModel::registerKnowledge** () [virtual]

Register knowledge.

Reimplemented from **AlpsModel**.

4.13.3.105 virtual **AlpsEncoded*** **BlisModel::encode** () const [virtual]

The method that encodes the model into an encoded object.

Reimplemented from **AlpsKnowledge**.

4.13.3.106 virtual void **BlisModel::decodeToSelf** (**AlpsEncoded** &) [virtual]

The method that decodes the model from an encoded object.

Reimplemented from **AlpsModel**.

4.13.3.107 virtual **AlpsEncoded*** **BlisModel::packSharedKnowlege** () [virtual]

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

Return NULL means that no knowledge can be shared.

Reimplemented from **AlpsModel**.

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

Unpack and store shared knowledge from an encoded object.

Reimplemented from **AlpsModel**.

4.13.4 Member Data Documentation

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

Input by user.

Definition at line 78 of file `BlisModel.h`.

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

Presolved.

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

4.13.4.3 `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`.

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

Column majored matrix.

(For MPS file, etc.)

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

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

Variable and constraint bounds.

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

4.13.4.6 `double BlisModel::objSense_` [protected]

Objective function.

Definition at line 109 of file `BlisModel.h`.

4.13.4.7 `int BlisModel::numIntObjects_` [protected]

Column types.

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

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

User's input objects.

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

4.13.4.9 `double BlisModel::incObjValue_` [protected]

Incumbent objective value.

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

4.13.4.10 `double BlisModel::cutoff_` [protected]

Cutoff in lp solver.

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

4.13.4.11 `double BlisModel::cutoffInc_` [protected]

Cutoff increment.

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

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

Variable selection function.

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

4.13.4.13 `int BlisModel::numObjects_` [protected]

Number of objects.

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

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

The set of objects.

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

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

The objects that can be shared.

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

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

Priorities of integer object.

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

4.13.4.17 AlpsTreeNode* BlisModel::activeNode_ [protected]

Active node.

Definition at line 191 of file BlisModel.h.

4.13.4.18 int BlisModel::numStrong_ [protected]

Number of strong branching.

Definition at line 194 of file BlisModel.h.

4.13.4.19 int BlisModel::numBranchResolve_ [protected]

Maximum number of resolve during branching.

Definition at line 200 of file BlisModel.h.

4.13.4.20 int BlisModel::numHeuristics_ [protected]

Number of heuristics.

Definition at line 207 of file BlisModel.h.

4.13.4.21 BlisHeuristic BlisModel::heuristics_** [protected]

The list of heuristics.

Definition at line 210 of file BlisModel.h.

4.13.4.22 BlisCutStrategy BlisModel::cutStrategy_ [protected]

If use cut generators.

Definition at line 217 of file BlisModel.h.

4.13.4.23 int BlisModel::numCutGenerators_ [protected]

Number of cut generators used.

Definition at line 223 of file BlisModel.h.

4.13.4.24 int BlisModel::maxNumCons_ [protected]

Number of cuts can be generators.

Definition at line 226 of file BlisModel.h.

4.13.4.25 BlisConGenerator BlisModel::generators_** [protected]

The list of cut generators used.

Definition at line 229 of file BlisModel.h.

4.13.4.26 BcpsConstraintPool* **BlisModel::constraintPool_** [protected]

Store all the cuts.

Definition at line 232 of file BlisModel.h.

4.13.4.27 BlisConstraint** **BlisModel::oldConstraints_** [protected]

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

Definition at line 235 of file BlisModel.h.

4.13.4.28 int **BlisModel::oldConstraintsSize_** [protected]

The memory size allocated for oldConstraints_.

Definition at line 238 of file BlisModel.h.

4.13.4.29 int **BlisModel::numOldConstraints_** [protected]

Number of old constraints.

Definition at line 241 of file BlisModel.h.

4.13.4.30 double* **BlisModel::conRandoms_** [protected]

Random keys.

Definition at line 244 of file BlisModel.h.

4.13.4.31 BlisParams* **BlisModel::BlisPar_** [protected]

Blis parameters.

Definition at line 254 of file BlisModel.h.

4.13.4.32 CoinMessageHandler* **BlisModel::blisMessageHandler_** [protected]

Message handler.

Definition at line 257 of file BlisModel.h.

4.13.4.33 CoinMessages **BlisModel::blisMessages_** [protected]

Blis messages.

Definition at line 260 of file BlisModel.h.

4.13.4.34 int **BlisModel::numNodes_** [protected]

Number of processed nodes.

Definition at line 263 of file BlisModel.h.

4.13.4.35 `int BlisModel::numIterations_` `[protected]`

Number of Ip(Simplex) iterations.

Definition at line 266 of file BlisModel.h.

4.13.4.36 `int BlisModel::aveIterations_` `[protected]`

Average number of Ip iterations to solve a subproblem.

Definition at line 269 of file BlisModel.h.

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

Temporary storage for var/con indices.

Definition at line 277 of file BlisModel.h.

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

Constraints that can be sent/broadcasted to other processes.

Definition at line 288 of file BlisModel.h.

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

Constraints that are received from other processes.

Definition at line 291 of file BlisModel.h.

4.13.4.40 `bool BlisModel::isRoot_`

If root node.

Definition at line 296 of file BlisModel.h.

4.13.4.41 `int BlisModel::boundingPass_`

The number of passes during bounding procedure.

Definition at line 299 of file BlisModel.h.

4.13.4.42 `double BlisModel::integerTol_`

Integer tolerance.

Definition at line 302 of file BlisModel.h.

4.13.4.43 `double BlisModel::optimalRelGap_`

Input relative optimal gap.

Definition at line 305 of file BlisModel.h.

4.13.4.44 double BlisModel::optimalAbsGap_

Input absolute optimal gap.

Definition at line 308 of file BlisModel.h.

4.13.4.45 double BlisModel::currRelGap_

Current relative optimal gap.

Definition at line 311 of file BlisModel.h.

4.13.4.46 double BlisModel::currAbsGap_

Current absolute optimal gap.

Definition at line 314 of file BlisModel.h.

4.13.4.47 BlisHeurStrategy BlisModel::heurStrategy_

If use heuristics.

Definition at line 317 of file BlisModel.h.

4.13.4.48 int BlisModel::heurCallFrequency_

Frequency of using heuristics.

Definition at line 320 of file BlisModel.h.

4.13.4.49 OsiCuts BlisModel::newCutPool_

Store new cuts in each pass.

Definition at line 323 of file BlisModel.h.

4.13.4.50 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

4.14 BlisNodeDesc Class Reference

Inheritance diagram for BlisNodeDesc:

Collaboration diagram for BlisNodeDesc:

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.

4.14.1 Detailed Description

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

4.14.2 Constructor & Destructor Documentation

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

Default constructor.

Definition at line 59 of file [BlisNodeDesc.h](#).

4.14.2.2 `BlisNodeDesc::BlisNodeDesc (BlisModel * m) [inline]`

Useful constructor.

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

4.14.2.3 `virtual BlisNodeDesc::~BlisNodeDesc () [inline],[virtual]`

Destructor.

Definition at line 78 of file `BlisNodeDesc.h`.

4.14.3 Member Function Documentation

4.14.3.1 `void BlisNodeDesc::setBasis (CoinWarmStartBasis *& ws) [inline]`

Set basis.

Definition at line 81 of file `BlisNodeDesc.h`.

4.14.3.2 `CoinWarmStartBasis* BlisNodeDesc::getBasis () const [inline]`

Get warm start basis.

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

4.14.3.3 `void BlisNodeDesc::setBranchedDir (int d) [inline]`

Set branching direction.

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

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

Get branching direction.

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

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

Set branching object index.

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

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

Get branching object index.

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

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

Set branching value.

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

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

Get branching direction.

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

4.14.3.9 `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`.

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

Unpack blis portion of node description from an encoded.

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

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

Pack node description into an encoded.

Reimplemented from **AlpsNodeDesc**.

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

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

Unpack a node description from an encoded.

Fill member data.

Reimplemented from **AlpsNodeDesc**.

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

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

- `BlisNodeDesc.h`

4.15 BlisObjectInt Class Reference

Inheritance diagram for `BlisObjectInt`:

Collaboration diagram for `BlisObjectInt`:

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.

4.15.1 Detailed Description

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

4.15.2 Constructor & Destructor Documentation

4.15.2.1 `BlisObjectInt::BlisObjectInt ()`

Default Constructor.

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

Useful constructor - passed integer index and model index.

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

Destructor.

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

4.15.2.4 `BlisObjectInt::BlisObjectInt (const BlisObjectInt &)`

Copy constructor.

4.15.3 Member Function Documentation

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

Clone an object.

Reimplemented from **BcpsObject**.

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

4.15.3.2 **BlisObjectInt**& **BlisObjectInt::operator=** (**const** **BlisObjectInt** & *rhs*)

Assignment operator.

4.15.3.3 **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.
---------------------	------------------------------------

Reimplemented from **BcpsObject**.

4.15.3.4 **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.

Reimplemented from **BcpsObject**.

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

Creates a branching object.

Reimplemented from **BcpsObject**.

4.15.3.6 **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.

Reimplemented from **BcpsObject**.

4.15.3.7 **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.

Reimplemented from **BcpsObject**.

4.15.3.8 `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.

Reimplemented from **BcpsObject**.

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

Column number if single column object, otherwise.

Definition at line 129 of file BlisObjectInt.h.

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

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

Definition at line 140 of file BlisObjectInt.h.

4.15.3.11 `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.

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

Access pseudocost.

Definition at line 146 of file BlisObjectInt.h.

4.15.4 Member Data Documentation

4.15.4.1 `int BlisObjectInt::columnIndex_ [protected]`

Column index in the lp model.

Definition at line 41 of file BlisObjectInt.h.

4.15.4.2 `double BlisObjectInt::originalLower_ [protected]`

Original lower bound.

Definition at line 44 of file BlisObjectInt.h.

4.15.4.3 `double BlisObjectInt::originalUpper_ [protected]`

Original upper bound.

Definition at line 47 of file BlisObjectInt.h.

4.15.4.4 `double BlisObjectInt::breakEven_` [protected]

Breakeven i.e.

\geq this preferred is up.

Definition at line 50 of file `BlisObjectInt.h`.

4.15.4.5 `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`

4.16 BlisParams Class Reference

Inheritance diagram for `BlisParams`:

Collaboration diagram for `BlisParams`:

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`, `cutoff`, `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 **endOfChrParams** 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 "`param.entry(USER_par::parameter_name)`" 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.

4.16.1 Detailed Description

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

4.16.2 Member Enumeration Documentation

4.16.2.1 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.

4.16.2.2 enum BlisParams::intParams

Integer paramters.

Enumerator

branchStrategy Branching strategy. 0: max infeasibilty, 1: pseudocost, 2: reliability, 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, BlisHuerStrategy↔ None: disable, BlisHeurStrategyPeriodic: every 't' nodes

heurCallFrequency All heuristics.

lookAhead The look ahead of pseudocost.

pseudoReliability The reliability of pseudocost.

sharePcCostDepth Maximum tree depth of sharing pseudocost.

sharePcCostFrequency Frequency of sharing pseudocost.

strongCandSize The number of candidate used in strong branching. Default: 10.

Definition at line 62 of file BlisParams.h.

4.16.2.3 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.

4.16.2.4 enum `BlisParams::strParams`

String parameters.

Definition at line 174 of file `BlisParams.h`.

4.16.2.5 enum `BlisParams::strArrayParams`

There are no string array parameters.

Definition at line 181 of file `BlisParams.h`.

4.16.3 Constructor & Destructor Documentation

4.16.3.1 `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`.

4.16.4 Member Function Documentation

4.16.4.1 `virtual void BlisParams::createKeywordList ()` `[virtual]`

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

Implements **AlpsParameterSet**.

4.16.4.2 `virtual void BlisParams::setDefaultEntries ()` `[virtual]`

Method for setting the default values for the parameters.

Implements **AlpsParameterSet**.

4.16.4.3 `void BlisParams::pack (AlpsEncoded & buf)` `[inline],[virtual]`

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

Reimplemented from **AlpsParameterSet**.

Definition at line 284 of file `BlisParams.h`.

4.16.4.4 `void BlisParams::unpack (AlpsEncoded & buf)` `[inline],[virtual]`

Unpack the parameter set from the buffer.

Reimplemented from **AlpsParameterSet**.

Definition at line 297 of file `BlisParams.h`.

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

- `BlisParams.h`

4.17 BlisPresolve Class Reference

A interface to Osi/Coin Presolve.

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

Inheritance diagram for BlisPresolve:

Collaboration diagram for BlisPresolve:

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.

4.17.1 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

4.18 BlisPseudocost Class Reference

Inheritance diagram for BlisPseudocost:

Collaboration diagram for BlisPseudocost:

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 weighth.

- 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.

4.18.1 Detailed Description

Definition at line 32 of file BlisPseudo.h.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 **BlisPseudocost::BlisPseudocost** () `[inline]`

Default constructor.

Definition at line 58 of file BlisPseudo.h.

4.18.2.2 **BlisPseudocost::BlisPseudocost** (double *uc*, int *un*, double *dc*, int *dn*, double *s*) `[inline]`

Useful constructor.

Definition at line 68 of file BlisPseudo.h.

4.18.3 Member Function Documentation

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

Set weigth.

Definition at line 104 of file BlisPseudo.h.

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

Update pseudocost.

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

Update pseudocost.

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

Update pseudocost.

4.18.3.5 int BlisPseudocost::getUpCount () [inline]

Get up branching count.

Definition at line 130 of file BlisPseudo.h.

4.18.3.6 double BlisPseudocost::getUpCost () [inline]

Get up branching cost.

Definition at line 133 of file BlisPseudo.h.

4.18.3.7 int BlisPseudocost::getDownCount () [inline]

Get down branching count.

Definition at line 136 of file BlisPseudo.h.

4.18.3.8 double BlisPseudocost::getDownCost () [inline]

Get down branching cost.

Definition at line 139 of file BlisPseudo.h.

4.18.3.9 double BlisPseudocost::getScore () [inline]

Get importance.

Definition at line 142 of file BlisPseudo.h.

4.18.3.10 `void BlisPseudocost::setScore (double s) [inline]`

Set importance.

Definition at line 145 of file BlisPseudo.h.

4.18.3.11 `AlpsReturnStatus BlisPseudocost::encodeTo (AlpsEncoded * encoded) const`

Pack pseudocost to the given object.

4.18.3.12 `AlpsReturnStatus BlisPseudocost::decodeFrom (AlpsEncoded & encoded)`

Unpack pseudocost from the given encode object.

4.18.3.13 `virtual AlpsEncoded* BlisPseudocost::encode () const [virtual]`

Encode this node for message passing.

Reimplemented from **AlpsKnowledge**.

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

Decode a node from an encoded object.

Reimplemented from **AlpsKnowledge**.

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

- BlisPseudo.h

4.19 BlisSolution Class Reference

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

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

Inheritance diagram for BlisSolution:

Collaboration diagram for BlisSolution:

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.

4.19.1 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.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 `BlisSolution::BlisSolution ()` `[inline]`

Default constructor.

Definition at line 43 of file BlisSolution.h.

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

Useful constructor.

Definition at line 49 of file BlisSolution.h.

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

Destructor.

Definition at line 55 of file BlisSolution.h.

4.19.3 Member Function Documentation

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

Print out the solution.

Print the solution.

Reimplemented from **BcpsSolution**.

Definition at line 59 of file BlisSolution.h.

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

The method that encodes the solution into a encoded object.

Reimplemented from **AlpsKnowledge**.

Definition at line 80 of file BlisSolution.h.

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

The method that decodes the solution from a encoded object.

Reimplemented from **AlpsKnowledge**.

Definition at line 88 of file BlisSolution.h.

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

- BlisSolution.h

4.20 BlisStrong Struct Reference

Collaboration diagram for BlisStrong:

4.20.1 Detailed Description

Definition at line 41 of file BlisBranchStrategyStrong.h.

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

- BlisBranchStrategyStrong.h

4.21 BlisTreeNode Class Reference

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

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

Inheritance diagram for BlisTreeNode:

Collaboration diagram for BlisTreeNode:

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 variables 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.

4.21.1 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.

4.21.2 Constructor & Destructor Documentation

4.21.2.1 **BlisTreeNode::BlisTreeNode** () `[inline]`

Default constructor.

Definition at line 79 of file BlisTreeNode.h.

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

Useful constructor.

Definition at line 85 of file BlisTreeNode.h.

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

Useful constructor.

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

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

Destructor.

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

4.21.3 Member Function Documentation

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

Initilize member data when constructing a node.

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

4.21.3.2 `AlpsTreeNode* BlisTreeNode::createNewTreeNode (AlpsNodeDesc *& desc) const [virtual]`

Create a new node based on given desc.

Implements **AlpsTreeNode**.

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

Performing the bounding operation.

Reimplemented from **BcpsTreeNode**.

4.21.3.4 `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`.

Implements **BcpsTreeNode**.

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

Select a branching object based on give branching strategy.

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

To be defined.

Implements **BcpsTreeNode**.

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

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

Generate constraints.

4.21.3.8 `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.

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

Get violated constraints.

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

Select and apply constraints.

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

Fix and tighten variables based optimality conditions.

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

Encode this node for message passing.

Reimplemented from **AlpsKnowledge**.

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

Decode a node from an encoded object.

Reimplemented from **AlpsKnowledge**.

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

- BlisSubTree.h
- BlisTreeNode.h

4.22 BlisVariable Class Reference

Inheritance diagram for BlisVariable:

Collaboration diagram for BlisVariable:

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.

4.22.1 Detailed Description

Definition at line 31 of file BlisVariable.h.

4.22.2 Member Function Documentation

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

Pack Blis part into an encoded object.

Definition at line 106 of file BlisVariable.h.

4.22.2.2 AlpsReturnStatus BlisVariable::decodeBlis (**AlpsEncoded** & *encoded*) [inline],[protected]

Unpack Blis part from a encode object.

Definition at line 119 of file BlisVariable.h.

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

Pack to a encode object.

Reimplemented from **BcpsObject**.

Definition at line 135 of file BlisVariable.h.

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

Decode a variable from an encoded object.

Reimplemented from **BcpsObject**.

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

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

- `BlisVariable.h`

File Documentation

Index

- ~BlisBranchObjectInt
 - BlisBranchObjectInt, [20](#)
- ~BlisBranchStrategyBilevel
 - BlisBranchStrategyBilevel, [22](#)
- ~BlisBranchStrategyMaxInf
 - BlisBranchStrategyMaxInf, [24](#)
- ~BlisBranchStrategyPseudo
 - BlisBranchStrategyPseudo, [26](#)
- ~BlisBranchStrategyRel
 - BlisBranchStrategyRel, [28](#)
- ~BlisBranchStrategyStrong
 - BlisBranchStrategyStrong, [29](#)
- ~BlisConGenerator
 - BlisConGenerator, [33](#)
- ~BlisConstraint
 - BlisConstraint, [40](#)
- ~BlisHeurRound
 - BlisHeurRound, [46](#)
- ~BlisHeuristic
 - BlisHeuristic, [43](#)
- ~BlisModel
 - BlisModel, [56](#)
- ~BlisNodeDesc
 - BlisNodeDesc, [76](#)
- ~BlisObjectInt
 - BlisObjectInt, [79](#)
- ~BlisSolution
 - BlisSolution, [90](#)
- ~BlisTreeNode
 - BlisTreeNode, [93](#)
- activeNode_
 - BlisModel, [70](#)
- addCalls
 - BlisConGenerator, [36](#)
 - BlisHeuristic, [44](#)
- addCutGenerator
 - BlisModel, [63](#)
- addHeuristic
 - BlisModel, [63](#)
- addNoConsCalls
 - BlisConGenerator, [36](#)
- addNoSolCalls
 - BlisHeuristic, [44](#)
- addNumConsGenerated
 - BlisConGenerator, [35](#)
- addNumConsUsed
 - BlisConGenerator, [35](#)
- addNumIterations
 - BlisModel, [67](#)
- addNumNodes
 - BlisModel, [67](#)
- addNumSolutions
 - BlisHeuristic, [43](#)
- addNumStrong
 - BlisModel, [58](#)
- addObjects
 - BlisModel, [62](#)
- addTime
 - BlisConGenerator, [36](#)
 - BlisHeuristic, [44](#)
- applyConstraints
 - BlisTreeNode, [94](#)
- atSolution
 - BlisConGenerator, [35](#)
- atSolution_
 - BlisConGenerator, [37](#)
- aveIterations_
 - BlisModel, [73](#)
- betterBranchObject
 - BlisBranchStrategyBilevel, [23](#)
 - BlisBranchStrategyMaxInf, [25](#)
 - BlisBranchStrategyPseudo, [26](#)
 - BlisBranchStrategyRel, [28](#)
 - BlisBranchStrategyStrong, [30](#)
- BlisBranchObjectBilevel, [17](#)
- BlisBranchObjectInt, [17](#)
 - ~BlisBranchObjectInt, [20](#)
 - BlisBranchObjectInt, [18](#), [20](#)
 - branch, [20](#)
 - clone, [20](#)
 - decode, [21](#)
 - decodeBlis, [21](#)
 - encode, [21](#)
 - encodeBlis, [21](#)
 - getDown, [21](#)
 - getUp, [21](#)
 - operator=, [20](#)
 - print, [20](#)
- BlisBranchStrategyBilevel, [21](#)
 - ~BlisBranchStrategyBilevel, [22](#)

- betterBranchObject, 23
 - BlisBranchStrategyBilevel, 22
 - clone, 23
 - createCandBranchObjects, 23
- BlisBranchStrategyMaxInf, 23
 - ~BlisBranchStrategyMaxInf, 24
 - betterBranchObject, 25
 - BlisBranchStrategyMaxInf, 24
 - clone, 24
 - createCandBranchObjects, 24
- BlisBranchStrategyPseudo, 25
 - ~BlisBranchStrategyPseudo, 26
 - betterBranchObject, 26
 - BlisBranchStrategyPseudo, 26
 - clone, 26
 - createCandBranchObjects, 27
 - setReliability, 26
- BlisBranchStrategyRel, 27
 - ~BlisBranchStrategyRel, 28
 - betterBranchObject, 28
 - BlisBranchStrategyRel, 28
 - clone, 28
 - createCandBranchObjects, 28
 - setReliability, 28
- BlisBranchStrategyStrong, 29
 - ~BlisBranchStrategyStrong, 29
 - betterBranchObject, 30
 - BlisBranchStrategyStrong, 29, 30
 - clone, 30
 - createCandBranchObjects, 30
- BlisConGenerator, 30
 - ~BlisConGenerator, 33
 - addCalls, 36
 - addNoConsCalls, 36
 - addNumConsGenerated, 35
 - addNumConsUsed, 35
 - addTime, 36
 - atSolution, 35
 - atSolution_, 37
 - BlisConGenerator, 33
 - calls, 36
 - calls_, 38
 - cutGenerationFreq, 34
 - cutGenerationFrequency_, 37
 - generateConstraints, 33
 - generator, 35
 - generator_, 36
 - getModel, 33
 - model_, 36
 - name, 34
 - name_, 37
 - noConsCalls, 36
 - noConsCalls_, 38
 - normal, 34
 - normal_, 37
 - numConsGenerated, 35
 - numConsGenerated_, 37
 - numConsUsed, 35
 - numConsUsed_, 37
 - operator=, 33
 - setAtSolution, 35
 - setCutGenerationFreq, 34
 - setName, 34
 - setNormal, 34
 - setStrategy, 34
 - setWhenInfeasible, 35
 - strategy, 34
 - strategy_, 37
 - time, 36
 - time_, 38
 - whenInfeasible, 35
 - whenInfeasible_, 37
- BlisConstraint, 38
 - ~BlisConstraint, 40
 - BlisConstraint, 39, 40
 - createOsiRowCut, 40
 - decode, 40
 - decodeBlis, 40
 - encode, 40
 - encodeBlis, 40
 - hashing, 40
 - violation, 40
- BlisHeurRound, 45
 - ~BlisHeurRound, 46
 - BlisHeurRound, 46
 - matrix_, 47
 - matrixByRow_, 47
 - seed_, 47
 - setModel, 47
- BlisHeuristic, 41
 - ~BlisHeuristic, 43
 - addCalls, 44
 - addNoSolCalls, 44
 - addNumSolutions, 43
 - addTime, 44
 - BlisHeuristic, 42, 43
 - calls, 44
 - calls_, 45
 - clone, 43
 - name, 43
 - noSolCalls, 44
 - noSolsCalls_, 45
 - numSolutions, 44
 - numSolutions_, 45
 - setHeurCallFrequency, 43
 - setModel, 43
 - setStrategy, 43
 - strategy_, 44

- time, [44](#)
- time_, [45](#)
- BlisMessage, [47](#)
- blisMessageHandler
 - BlisModel, [67](#)
- blisMessageHandler_
 - BlisModel, [72](#)
- blisMessages
 - BlisModel, [67](#)
- blisMessages_
 - BlisModel, [72](#)
- BlisModel, [48](#)
 - ~BlisModel, [56](#)
 - activeNode_, [70](#)
 - addCutGenerator, [63](#)
 - addHeuristic, [63](#)
 - addNumIterations, [67](#)
 - addNumNodes, [67](#)
 - addNumStrong, [58](#)
 - addObjects, [62](#)
 - avelterations_, [73](#)
 - blisMessageHandler, [67](#)
 - blisMessageHandler_, [72](#)
 - blisMessages, [67](#)
 - blisMessages_, [72](#)
 - BlisModel, [56](#)
 - BlisPar, [67](#)
 - BlisPar_, [72](#)
 - boundingPass_, [73](#)
 - branchStrategy, [61](#)
 - branchStrategy_, [70](#)
 - checkInteger, [63](#)
 - clearSharedObjectMark, [62](#)
 - colMatrix_, [69](#)
 - conLB, [59](#)
 - conRandoms_, [72](#)
 - constraintPool, [64](#)
 - constraintPool_, [71](#)
 - constraintPoolReceive, [64](#)
 - constraintPoolReceive_, [73](#)
 - constraintPoolSend, [64](#)
 - constraintPoolSend_, [73](#)
 - createIntgerObjects, [62](#)
 - createObjects, [56](#)
 - createRoot, [57](#)
 - currAbsGap_, [74](#)
 - currRelGap_, [74](#)
 - cutGenerators, [63](#)
 - cutStrategy_, [71](#)
 - cutoff_, [70](#)
 - cutoffInc_, [70](#)
 - decodeBlis, [67](#)
 - decodeToSelf, [68](#)
 - delOldConstraints, [65](#)
 - deleteObjects, [62](#)
 - encode, [68](#)
 - encodeBlis, [67](#)
 - fathomAllNodes, [67](#)
 - feasibleSolution, [61](#)
 - feasibleSolutionHeur, [61](#)
 - generators_, [71](#)
 - getAvelterations, [66](#)
 - getColLower, [59](#)
 - getColUpper, [59](#)
 - getConRandoms, [66](#)
 - getCutGenerationFrequency, [65](#)
 - getCutStrategy, [65](#)
 - getCutoff, [60](#)
 - getDenseConCutoff, [65](#)
 - getIntColIndices, [63](#)
 - getIntObjIndices, [62](#)
 - getLpObjValue, [60](#)
 - getLpSolution, [60](#)
 - getMaxNumCons, [64](#)
 - getNumBranchResolve, [58](#)
 - getNumCols, [59](#)
 - getNumHeurSolutions, [60](#)
 - getNumIntObjects, [63](#)
 - getNumIterations, [66](#)
 - getNumNodes, [66](#)
 - getNumOldConstraints, [64](#)
 - getNumRows, [59](#)
 - getNumSolutions, [60](#)
 - getNumStrong, [58](#)
 - getObjCoef, [59](#)
 - getOldConstraintsSize, [64](#)
 - getSolver, [58](#)
 - gutsOfDestructor, [56](#)
 - heurCallFrequency_, [74](#)
 - heurStrategy_, [74](#)
 - heuristics, [63](#)
 - heuristics_, [71](#)
 - importModel, [57](#)
 - incObjValue_, [70](#)
 - incumbent, [60](#)
 - inputVar_, [69](#)
 - integerTol_, [73](#)
 - isRoot_, [73](#)
 - leafToRootPath, [74](#)
 - lpSolver_, [69](#)
 - maxNumCons_, [71](#)
 - modelLog, [66](#)
 - newCutPool_, [74](#)
 - nodeLog, [67](#)
 - numBranchResolve_, [71](#)
 - numCutGenerators, [64](#)
 - numCutGenerators_, [71](#)
 - numHeuristics, [63](#)

- numHeuristics_, 71
- numIntObjects_, 69
- numIterations_, 72
- numNodes_, 72
- numObjects, 61
- numObjects_, 70
- numOldConstraints_, 72
- numStrong_, 71
- objSense_, 69
- objects, 62
- objects_, 70
- oldConstraints, 65
- oldConstraints_, 72
- oldConstraintsSize_, 72
- optimalAbsGap_, 73
- optimalRelGap_, 73
- origLpSolver_, 69
- packSharedConstraints, 68
- packSharedKnowlege, 68
- packSharedPseudocost, 68
- packSharedVariables, 68
- passInPriorities, 66
- postprocess, 57
- preprocess, 57
- presolvedLpSolver_, 69
- priority, 66
- priority_, 70
- readInstance, 56
- readParameters, 57
- registerKnowledge, 68
- resolve, 58
- setActiveNode, 58
- setBranchingMethod, 61
- setCutStrategy, 65
- setCutoff, 61
- setDenseConCutoff, 66
- setMaxNumCons, 64
- setNumBranchResolve, 59
- setNumObjects, 62
- setNumOldConstraints, 64
- setOldConstraints, 65
- setOldConstraintsSize, 65
- setSharedObjectMark, 62
- setSolEstimate, 58
- setSolver, 58
- setupSelf, 57
- sharedObjectMark_, 70
- solver, 58
- startConLB, 60
- startVarLB, 59
- storeSolution, 60
- tempVarLBPos, 60
- tempVarLBPos_, 73
- unpackSharedConstraints, 68
- unpackSharedKnowledge, 68
- unpackSharedVariables, 68
- userFeasibleSolution, 61
- varLB, 59
- varLB_, 69
- writeParameters, 57
- BlisNodeDesc, 74
 - ~BlisNodeDesc, 76
 - BlisNodeDesc, 75
 - decode, 77
 - decodeBlis, 77
 - encode, 77
 - encodeBlis, 77
 - getBasis, 76
 - getBranchedDir, 76
 - getBranchedInd, 76
 - getBranchedVal, 77
 - setBasis, 76
 - setBranchedDir, 76
 - setBranchedInd, 76
 - setBranchedVal, 76
- BlisObjectInt, 77
 - ~BlisObjectInt, 79
 - BlisObjectInt, 79
 - breakEven, 81
 - breakEven_, 81
 - clone, 79
 - columnIndex, 81
 - columnIndex_, 81
 - createBranchObject, 80
 - feasibleRegion, 80
 - infeasibility, 80
 - notPreferredNewFeasible, 80
 - operator=, 79
 - originalLower_, 81
 - originalUpper_, 81
 - preferredNewFeasible, 80
 - pseudocost, 81
 - pseudocost_, 82
 - resetBounds, 80
 - setBreakEven, 81
- BlisPar
 - BlisModel, 67
- BlisPar_
 - BlisModel, 72
- BlisParams, 82
 - BlisParams, 85
 - branchStrategy, 84
 - chrParams, 83
 - createKeywordList, 85
 - cutCliqueStrategy, 84
 - cutFactor, 84
 - cutGenerationFrequency, 84
 - cutRampUp, 83

- cutStrategy, 84
- cutoff, 84
- cutoffInc, 84
- dblParams, 84
- denseConFactor, 84
- difference, 84
- heurCallFrequency, 84
- heurStrategy, 84
- intParams, 84
- integerTol, 84
- lookAhead, 84
- objSense, 84
- optimalAbsGap, 84
- optimalRelGap, 84
- pack, 85
- presolve, 83
- pseudoReliability, 84
- pseudoWeight, 84
- quickCutPass, 84
- scaleConFactor, 84
- setDefaultEntries, 85
- shareConstraints, 83
- sharePcostDepth, 84
- sharePcostFrequency, 84
- sharePseudocostRampUp, 84
- sharePseudocostSearch, 84
- shareVariables, 83
- strArrayParams, 85
- strParams, 84
- strongCandSize, 84
- tailOff, 84
- unpack, 85
- BlisPresolve, 86
- BlisPseudocost, 86
 - BlisPseudocost, 87
 - decode, 89
 - decodeFrom, 89
 - encode, 89
 - encodeTo, 89
 - getDownCost, 88
 - getDownCount, 88
 - getScore, 88
 - getUpCost, 88
 - getUpCount, 88
 - setScore, 88
 - setWeight, 88
 - update, 88
- BlisSolution, 89
 - ~BlisSolution, 90
 - BlisSolution, 90
 - decode, 90
 - encode, 90
 - print, 90
- BlisStrong, 91
- BlisTreeNode, 91
 - ~BlisTreeNode, 93
 - applyConstraints, 94
 - BlisTreeNode, 92
 - branch, 93
 - callHeuristics, 94
 - chooseBranchingObject, 93
 - createNewTreeNode, 93
 - decode, 94
 - encode, 94
 - generateConstraints, 93
 - getViolatedConstraints, 94
 - init, 93
 - process, 93
 - reducedCostFix, 94
 - selectBranchObject, 93
- BlisVariable, 94
 - decode, 95
 - decodeBlis, 95
 - encode, 95
 - encodeBlis, 95
- boundingPass_
 - BlisModel, 73
- branch
 - BlisBranchObjectInt, 20
 - BlisTreeNode, 93
- branchStrategy
 - BlisModel, 61
 - BlisParams, 84
- branchStrategy_
 - BlisModel, 70
- breakEven
 - BlisObjectInt, 81
- breakEven_
 - BlisObjectInt, 81
- callHeuristics
 - BlisTreeNode, 94
- calls
 - BlisConGenerator, 36
 - BlisHeuristic, 44
- calls_
 - BlisConGenerator, 38
 - BlisHeuristic, 45
- checkInteger
 - BlisModel, 63
- chooseBranchingObject
 - BlisTreeNode, 93
- chrParams
 - BlisParams, 83
- clearSharedObjectMark
 - BlisModel, 62
- clone
 - BlisBranchObjectInt, 20

- BlisBranchStrategyBilevel, 23
- BlisBranchStrategyMaxInf, 24
- BlisBranchStrategyPseudo, 26
- BlisBranchStrategyRel, 28
- BlisBranchStrategyStrong, 30
- BlisHeuristic, 43
- BlisObjectInt, 79
- colMatrix_
 - BlisModel, 69
- columnIndex
 - BlisObjectInt, 81
- columnIndex_
 - BlisObjectInt, 81
- conLB
 - BlisModel, 59
- conRandoms_
 - BlisModel, 72
- constraintPool
 - BlisModel, 64
- constraintPool_
 - BlisModel, 71
- constraintPoolReceive
 - BlisModel, 64
- constraintPoolReceive_
 - BlisModel, 73
- constraintPoolSend
 - BlisModel, 64
- constraintPoolSend_
 - BlisModel, 73
- createBranchObject
 - BlisObjectInt, 80
- createCandBranchObjects
 - BlisBranchStrategyBilevel, 23
 - BlisBranchStrategyMaxInf, 24
 - BlisBranchStrategyPseudo, 27
 - BlisBranchStrategyRel, 28
 - BlisBranchStrategyStrong, 30
- createIntgerObjects
 - BlisModel, 62
- createKeywordList
 - BlisParams, 85
- createNewTreeNode
 - BlisTreeNode, 93
- createObjects
 - BlisModel, 56
- createOsiRowCut
 - BlisConstraint, 40
- createRoot
 - BlisModel, 57
- currAbsGap_
 - BlisModel, 74
- currRelGap_
 - BlisModel, 74
- cutCliqueStrategy
 - BlisParams, 84
- cutFactor
 - BlisParams, 84
- cutGenerationFreq
 - BlisConGenerator, 34
- cutGenerationFrequency
 - BlisParams, 84
- cutGenerationFrequency_
 - BlisConGenerator, 37
- cutGenerators
 - BlisModel, 63
- cutRampUp
 - BlisParams, 83
- cutStrategy
 - BlisParams, 84
- cutStrategy_
 - BlisModel, 71
- cutoff
 - BlisParams, 84
- cutoff_
 - BlisModel, 70
- cutoffInc
 - BlisParams, 84
- cutoffInc_
 - BlisModel, 70
- dblParams
 - BlisParams, 84
- decode
 - BlisBranchObjectInt, 21
 - BlisConstraint, 40
 - BlisNodeDesc, 77
 - BlisPseudocost, 89
 - BlisSolution, 90
 - BlisTreeNode, 94
 - BlisVariable, 95
- decodeBlis
 - BlisBranchObjectInt, 21
 - BlisConstraint, 40
 - BlisModel, 67
 - BlisNodeDesc, 77
 - BlisVariable, 95
- decodeFrom
 - BlisPseudocost, 89
- decodeToSelf
 - BlisModel, 68
- delOldConstraints
 - BlisModel, 65
- deleteObjects
 - BlisModel, 62
- denseConFactor
 - BlisParams, 84
- difference
 - BlisParams, 84

- encode
 - BlisBranchObjectInt, 21
 - BlisConstraint, 40
 - BlisModel, 68
 - BlisNodeDesc, 77
 - BlisPseudocost, 89
 - BlisSolution, 90
 - BlisTreeNode, 94
 - BlisVariable, 95
- encodeBlis
 - BlisBranchObjectInt, 21
 - BlisConstraint, 40
 - BlisModel, 67
 - BlisNodeDesc, 77
 - BlisVariable, 95
- encodeTo
 - BlisPseudocost, 89
- fathomAllNodes
 - BlisModel, 67
- feasibleRegion
 - BlisObjectInt, 80
- feasibleSolution
 - BlisModel, 61
- feasibleSolutionHeur
 - BlisModel, 61
- generateConstraints
 - BlisConGenerator, 33
 - BlisTreeNode, 93
- generator
 - BlisConGenerator, 35
- generator_
 - BlisConGenerator, 36
- generators_
 - BlisModel, 71
- getAvelterations
 - BlisModel, 66
- getBasis
 - BlisNodeDesc, 76
- getBranchedDir
 - BlisNodeDesc, 76
- getBranchedInd
 - BlisNodeDesc, 76
- getBranchedVal
 - BlisNodeDesc, 77
- getColLower
 - BlisModel, 59
- getColUpper
 - BlisModel, 59
- getConRandoms
 - BlisModel, 66
- getCutGenerationFrequency
 - BlisModel, 65
- getCutStrategy
 - BlisModel, 65
- getCutoff
 - BlisModel, 60
- getDenseConCutoff
 - BlisModel, 65
- getDown
 - BlisBranchObjectInt, 21
- getDownCost
 - BlisPseudocost, 88
- getDownCount
 - BlisPseudocost, 88
- getIntColIndices
 - BlisModel, 63
- getIntObjIndices
 - BlisModel, 62
- getLpObjValue
 - BlisModel, 60
- getLpSolution
 - BlisModel, 60
- getMaxNumCons
 - BlisModel, 64
- getModel
 - BlisConGenerator, 33
- getNumBranchResolve
 - BlisModel, 58
- getNumCols
 - BlisModel, 59
- getNumHeurSolutions
 - BlisModel, 60
- getNumIntObjects
 - BlisModel, 63
- getNumIterations
 - BlisModel, 66
- getNumNodes
 - BlisModel, 66
- getNumOldConstraints
 - BlisModel, 64
- getNumRows
 - BlisModel, 59
- getNumSolutions
 - BlisModel, 60
- getNumStrong
 - BlisModel, 58
- getObjCoef
 - BlisModel, 59
- getOldConstraintsSize
 - BlisModel, 64
- getScore
 - BlisPseudocost, 88
- getSolver
 - BlisModel, 58
- getUp
 - BlisBranchObjectInt, 21
- getUpCost

- BlisPseudocost, 88
- getUpCount
 - BlisPseudocost, 88
- getViolatedConstraints
 - BlisTreeNode, 94
- gutsOfDestructor
 - BlisModel, 56
- hashing
 - BlisConstraint, 40
- heurCallFrequency
 - BlisParams, 84
- heurCallFrequency_
 - BlisModel, 74
- heurStrategy
 - BlisParams, 84
- heurStrategy_
 - BlisModel, 74
- heuristics
 - BlisModel, 63
- heuristics_
 - BlisModel, 71
- importModel
 - BlisModel, 57
- incObjValue_
 - BlisModel, 70
- incumbent
 - BlisModel, 60
- infeasibility
 - BlisObjectInt, 80
- init
 - BlisTreeNode, 93
- inputVar_
 - BlisModel, 69
- intParams
 - BlisParams, 84
- integerTol
 - BlisParams, 84
- integerTol_
 - BlisModel, 73
- isRoot_
 - BlisModel, 73
- leafToRootPath
 - BlisModel, 74
- lookAhead
 - BlisParams, 84
- lpSolver_
 - BlisModel, 69
- matrix_
 - BlisHeurRound, 47
- matrixByRow_
 - BlisHeurRound, 47
- maxNumCons_
 - BlisModel, 71
- model_
 - BlisConGenerator, 36
- modelLog
 - BlisModel, 66
- name
 - BlisConGenerator, 34
 - BlisHeuristic, 43
- name_
 - BlisConGenerator, 37
- newCutPool_
 - BlisModel, 74
- noConsCalls
 - BlisConGenerator, 36
- noConsCalls_
 - BlisConGenerator, 38
- noSolCalls
 - BlisHeuristic, 44
- noSolsCalls_
 - BlisHeuristic, 45
- nodeLog
 - BlisModel, 67
- normal
 - BlisConGenerator, 34
- normal_
 - BlisConGenerator, 37
- notPreferredNewFeasible
 - BlisObjectInt, 80
- numBranchResolve_
 - BlisModel, 71
- numConsGenerated
 - BlisConGenerator, 35
- numConsGenerated_
 - BlisConGenerator, 37
- numConsUsed
 - BlisConGenerator, 35
- numConsUsed_
 - BlisConGenerator, 37
- numCutGenerators
 - BlisModel, 64
- numCutGenerators_
 - BlisModel, 71
- numHeuristics
 - BlisModel, 63
- numHeuristics_
 - BlisModel, 71
- numIntObjects_
 - BlisModel, 69
- numIterations_
 - BlisModel, 72
- numNodes_
 - BlisModel, 72

- numObjects
 - BlisModel, 61
- numObjects_
 - BlisModel, 70
- numOldConstraints_
 - BlisModel, 72
- numSolutions
 - BlisHeuristic, 44
- numSolutions_
 - BlisHeuristic, 45
- numStrong_
 - BlisModel, 71
- objSense
 - BlisParams, 84
- objSense_
 - BlisModel, 69
- objects
 - BlisModel, 62
- objects_
 - BlisModel, 70
- oldConstraints
 - BlisModel, 65
- oldConstraints_
 - BlisModel, 72
- oldConstraintsSize_
 - BlisModel, 72
- operator=
 - BlisBranchObjectInt, 20
 - BlisConGenerator, 33
 - BlisObjectInt, 79
- optimalAbsGap
 - BlisParams, 84
- optimalAbsGap_
 - BlisModel, 73
- optimalRelGap
 - BlisParams, 84
- optimalRelGap_
 - BlisModel, 73
- origLpSolver_
 - BlisModel, 69
- originalLower_
 - BlisObjectInt, 81
- originalUpper_
 - BlisObjectInt, 81
- pack
 - BlisParams, 85
- packSharedConstraints
 - BlisModel, 68
- packSharedKnowledge
 - BlisModel, 68
- packSharedPseudocost
 - BlisModel, 68
- packSharedVariables
 - BlisModel, 68
- passInPriorities
 - BlisModel, 66
- postprocess
 - BlisModel, 57
- preferredNewFeasible
 - BlisObjectInt, 80
- preprocess
 - BlisModel, 57
- presolve
 - BlisParams, 83
- presolvedLpSolver_
 - BlisModel, 69
- print
 - BlisBranchObjectInt, 20
 - BlisSolution, 90
- priority
 - BlisModel, 66
- priority_
 - BlisModel, 70
- process
 - BlisTreeNode, 93
- pseudoReliability
 - BlisParams, 84
- pseudoWeight
 - BlisParams, 84
- pseudocost
 - BlisObjectInt, 81
- pseudocost_
 - BlisObjectInt, 82
- quickCutPass
 - BlisParams, 84
- readInstance
 - BlisModel, 56
- readParameters
 - BlisModel, 57
- reducedCostFix
 - BlisTreeNode, 94
- registerKnowledge
 - BlisModel, 68
- resetBounds
 - BlisObjectInt, 80
- resolve
 - BlisModel, 58
- scaleConFactor
 - BlisParams, 84
- seed_
 - BlisHeurRound, 47
- selectBranchObject
 - BlisTreeNode, 93
- setActiveNode
 - BlisModel, 58

setAtSolution
 BlisConGenerator, 35
 setBasis
 BlisNodeDesc, 76
 setBranchedDir
 BlisNodeDesc, 76
 setBranchedInd
 BlisNodeDesc, 76
 setBranchedVal
 BlisNodeDesc, 76
 setBranchingMethod
 BlisModel, 61
 setBreakEven
 BlisObjectInt, 81
 setCutGenerationFreq
 BlisConGenerator, 34
 setCutStrategy
 BlisModel, 65
 setCutoff
 BlisModel, 61
 setDefaultEntries
 BlisParams, 85
 setDenseConCutoff
 BlisModel, 66
 setHeurCallFrequency
 BlisHeuristic, 43
 setMaxNumCons
 BlisModel, 64
 setModel
 BlisHeurRound, 47
 BlisHeuristic, 43
 setName
 BlisConGenerator, 34
 setNormal
 BlisConGenerator, 34
 setNumBranchResolve
 BlisModel, 59
 setNumObjects
 BlisModel, 62
 setNumOldConstraints
 BlisModel, 64
 setOldConstraints
 BlisModel, 65
 setOldConstraintsSize
 BlisModel, 65
 setReliability
 BlisBranchStrategyPseudo, 26
 BlisBranchStrategyRel, 28
 setScore
 BlisPseudocost, 88
 setSharedObjectMark
 BlisModel, 62
 setSolEstimate
 BlisModel, 58
 setSolver
 BlisModel, 58
 setStrategy
 BlisConGenerator, 34
 BlisHeuristic, 43
 setWeight
 BlisPseudocost, 88
 setWhenInfeasible
 BlisConGenerator, 35
 setupSelf
 BlisModel, 57
 shareConstraints
 BlisParams, 83
 sharePcostDepth
 BlisParams, 84
 sharePcostFrequency
 BlisParams, 84
 sharePseudocostRampUp
 BlisParams, 84
 sharePseudocostSearch
 BlisParams, 84
 shareVariables
 BlisParams, 83
 sharedObjectMark_
 BlisModel, 70
 solver
 BlisModel, 58
 startConLB
 BlisModel, 60
 startVarLB
 BlisModel, 59
 storeSolution
 BlisModel, 60
 strArrayParams
 BlisParams, 85
 strParams
 BlisParams, 84
 strategy
 BlisConGenerator, 34
 strategy_
 BlisConGenerator, 37
 BlisHeuristic, 44
 strongCandSize
 BlisParams, 84
 tailOff
 BlisParams, 84
 tempVarLBPos
 BlisModel, 60
 tempVarLBPos_
 BlisModel, 73
 time
 BlisConGenerator, 36
 BlisHeuristic, 44

time_
 BlisConGenerator, [38](#)
 BlisHeuristic, [45](#)

unpack
 BlisParams, [85](#)

unpackSharedConstraints
 BlisModel, [68](#)

unpackSharedKnowledge
 BlisModel, [68](#)

unpackSharedVariables
 BlisModel, [68](#)

update
 BlisPseudocost, [88](#)

userFeasibleSolution
 BlisModel, [61](#)

varLB
 BlisModel, [59](#)

varLB_
 BlisModel, [69](#)

violation
 BlisConstraint, [40](#)

whenInfeasible
 BlisConGenerator, [35](#)

whenInfeasible_
 BlisConGenerator, [37](#)

writeParameters
 BlisModel, [57](#)