

Cgl  
trunk

Generated by Doxygen 1.8.5

Mon Oct 21 2013 18:59:11

## Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Namespace Index</b>                           | <b>1</b>  |
| 1.1      | Namespace List . . . . .                         | 1         |
| <b>2</b> | <b>Hierarchical Index</b>                        | <b>1</b>  |
| 2.1      | Class Hierarchy . . . . .                        | 1         |
| <b>3</b> | <b>Class Index</b>                               | <b>7</b>  |
| 3.1      | Class List . . . . .                             | 7         |
| <b>4</b> | <b>File Index</b>                                | <b>10</b> |
| 4.1      | File List . . . . .                              | 10        |
| <b>5</b> | <b>Namespace Documentation</b>                   | <b>12</b> |
| 5.1      | LAP Namespace Reference . . . . .                | 12        |
| 5.1.1    | Detailed Description . . . . .                   | 13        |
| 5.1.2    | Enumeration Type Documentation . . . . .         | 13        |
| 5.1.3    | Function Documentation . . . . .                 | 14        |
| <b>6</b> | <b>Class Documentation</b>                       | <b>14</b> |
| 6.1      | auxiliary_graph Struct Reference . . . . .       | 14        |
| 6.1.1    | Detailed Description . . . . .                   | 15        |
| 6.1.2    | Member Data Documentation . . . . .              | 15        |
| 6.2      | Cgl012Cut Class Reference . . . . .              | 15        |
| 6.2.1    | Detailed Description . . . . .                   | 16        |
| 6.2.2    | Constructor & Destructor Documentation . . . . . | 16        |
| 6.2.3    | Member Function Documentation . . . . .          | 16        |
| 6.3      | cgl_arc Struct Reference . . . . .               | 17        |
| 6.3.1    | Detailed Description . . . . .                   | 17        |
| 6.3.2    | Member Data Documentation . . . . .              | 17        |
| 6.4      | cgl_graph Struct Reference . . . . .             | 17        |
| 6.4.1    | Detailed Description . . . . .                   | 17        |
| 6.4.2    | Member Data Documentation . . . . .              | 17        |
| 6.5      | cgl_node Struct Reference . . . . .              | 18        |
| 6.5.1    | Detailed Description . . . . .                   | 18        |
| 6.5.2    | Member Data Documentation . . . . .              | 18        |
| 6.6      | CglAlldifferent Class Reference . . . . .        | 19        |
| 6.6.1    | Detailed Description . . . . .                   | 20        |

|        |  |    |
|--------|--|----|
| 6.6.2  | Constructor & Destructor Documentation     | 20 |
| 6.6.3  | Member Function Documentation              | 20 |
| 6.7    | CglBK Class Reference                      | 21 |
| 6.7.1  | Detailed Description                       | 22 |
| 6.7.2  | Constructor & Destructor Documentation     | 22 |
| 6.7.3  | Member Function Documentation              | 22 |
| 6.8    | CglClique Class Reference                  | 23 |
| 6.8.1  | Detailed Description                       | 25 |
| 6.8.2  | Member Enumeration Documentation           | 25 |
| 6.8.3  | Constructor & Destructor Documentation     | 25 |
| 6.8.4  | Member Function Documentation              | 26 |
| 6.8.5  | Friends And Related Function Documentation | 27 |
| 6.8.6  | Member Data Documentation                  | 27 |
| 6.9    | CglCutGenerator Class Reference            | 30 |
| 6.9.1  | Detailed Description                       | 32 |
| 6.9.2  | Constructor & Destructor Documentation     | 32 |
| 6.9.3  | Member Function Documentation              | 32 |
| 6.9.4  | Member Data Documentation                  | 34 |
| 6.10   | CglDuplicateRow Class Reference            | 34 |
| 6.10.1 | Detailed Description                       | 36 |
| 6.10.2 | Constructor & Destructor Documentation     | 37 |
| 6.10.3 | Member Function Documentation              | 37 |
| 6.10.4 | Member Data Documentation                  | 39 |
| 6.11   | CglFakeClique Class Reference              | 40 |
| 6.11.1 | Detailed Description                       | 41 |
| 6.11.2 | Constructor & Destructor Documentation     | 41 |
| 6.11.3 | Member Function Documentation              | 42 |
| 6.11.4 | Member Data Documentation                  | 42 |
| 6.12   | CglFlowCover Class Reference               | 42 |
| 6.12.1 | Detailed Description                       | 44 |
| 6.12.2 | Constructor & Destructor Documentation     | 44 |
| 6.12.3 | Member Function Documentation              | 44 |
| 6.12.4 | Friends And Related Function Documentation | 45 |
| 6.13   | CglFlowVUB Class Reference                 | 45 |
| 6.13.1 | Detailed Description                       | 46 |
| 6.13.2 | Constructor & Destructor Documentation     | 46 |
| 6.13.3 | Member Function Documentation              | 46 |

|        |  |    |
|--------|--|----|
| 6.13.4 | Member Data Documentation                  | 46 |
| 6.14   | CglGMI Class Reference                     | 47 |
| 6.14.1 | Detailed Description                       | 48 |
| 6.14.2 | Member Enumeration Documentation           | 48 |
| 6.14.3 | Constructor & Destructor Documentation     | 49 |
| 6.14.4 | Member Function Documentation              | 49 |
| 6.14.5 | Friends And Related Function Documentation | 51 |
| 6.15   | CglGMIParam Class Reference                | 51 |
| 6.15.1 | Detailed Description                       | 54 |
| 6.15.2 | Member Enumeration Documentation           | 55 |
| 6.15.3 | Constructor & Destructor Documentation     | 55 |
| 6.15.4 | Member Function Documentation              | 55 |
| 6.15.5 | Member Data Documentation                  | 61 |
| 6.16   | CglGomory Class Reference                  | 62 |
| 6.16.1 | Detailed Description                       | 65 |
| 6.16.2 | Constructor & Destructor Documentation     | 65 |
| 6.16.3 | Member Function Documentation              | 65 |
| 6.16.4 | Friends And Related Function Documentation | 67 |
| 6.17   | CglHashLink Struct Reference               | 68 |
| 6.17.1 | Detailed Description                       | 68 |
| 6.17.2 | Member Data Documentation                  | 68 |
| 6.18   | CglImplication Class Reference             | 68 |
| 6.18.1 | Detailed Description                       | 69 |
| 6.18.2 | Constructor & Destructor Documentation     | 69 |
| 6.18.3 | Member Function Documentation              | 70 |
| 6.19   | CglKnapsackCover Class Reference           | 70 |
| 6.19.1 | Detailed Description                       | 71 |
| 6.19.2 | Constructor & Destructor Documentation     | 71 |
| 6.19.3 | Member Function Documentation              | 72 |
| 6.19.4 | Friends And Related Function Documentation | 73 |
| 6.20   | CglLandP Class Reference                   | 73 |
| 6.20.1 | Detailed Description                       | 74 |
| 6.20.2 | Member Enumeration Documentation           | 74 |
| 6.20.3 | Constructor & Destructor Documentation     | 76 |
| 6.20.4 | Member Function Documentation              | 76 |
| 6.20.5 | Friends And Related Function Documentation | 77 |
| 6.21   | LAP::CglLandPSimplex Class Reference       | 77 |

|   |     |
|---|-----|
| 6.21.1 Detailed Description . . . . .                       | 80  |
| 6.21.2 Constructor & Destructor Documentation . . . . .     | 80  |
| 6.21.3 Member Function Documentation . . . . .              | 80  |
| 6.22 CglLiftAndProject Class Reference . . . . .            | 85  |
| 6.22.1 Detailed Description . . . . .                       | 86  |
| 6.22.2 Constructor & Destructor Documentation . . . . .     | 86  |
| 6.22.3 Member Function Documentation . . . . .              | 86  |
| 6.22.4 Friends And Related Function Documentation . . . . . | 87  |
| 6.23 CglMessage Class Reference . . . . .                   | 87  |
| 6.23.1 Detailed Description . . . . .                       | 88  |
| 6.23.2 Constructor & Destructor Documentation . . . . .     | 88  |
| 6.24 CglMixedIntegerRounding Class Reference . . . . .      | 88  |
| 6.24.1 Detailed Description . . . . .                       | 89  |
| 6.24.2 Constructor & Destructor Documentation . . . . .     | 89  |
| 6.24.3 Member Function Documentation . . . . .              | 90  |
| 6.24.4 Friends And Related Function Documentation . . . . . | 91  |
| 6.25 CglMixedIntegerRounding2 Class Reference . . . . .     | 91  |
| 6.25.1 Detailed Description . . . . .                       | 92  |
| 6.25.2 Constructor & Destructor Documentation . . . . .     | 93  |
| 6.25.3 Member Function Documentation . . . . .              | 93  |
| 6.25.4 Friends And Related Function Documentation . . . . . | 94  |
| 6.26 CglMixIntRoundVUB Class Reference . . . . .            | 94  |
| 6.26.1 Detailed Description . . . . .                       | 95  |
| 6.26.2 Constructor & Destructor Documentation . . . . .     | 95  |
| 6.26.3 Member Function Documentation . . . . .              | 95  |
| 6.26.4 Member Data Documentation . . . . .                  | 95  |
| 6.27 CglMixIntRoundVUB2 Class Reference . . . . .           | 96  |
| 6.27.1 Detailed Description . . . . .                       | 96  |
| 6.27.2 Constructor & Destructor Documentation . . . . .     | 96  |
| 6.27.3 Member Function Documentation . . . . .              | 97  |
| 6.27.4 Member Data Documentation . . . . .                  | 97  |
| 6.28 CglOddHole Class Reference . . . . .                   | 97  |
| 6.28.1 Detailed Description . . . . .                       | 99  |
| 6.28.2 Constructor & Destructor Documentation . . . . .     | 99  |
| 6.28.3 Member Function Documentation . . . . .              | 99  |
| 6.28.4 Friends And Related Function Documentation . . . . . | 100 |
| 6.29 CglParam Class Reference . . . . .                     | 100 |

|   |     |
|---|-----|
| 6.29.1 Detailed Description . . . . .                       | 102 |
| 6.29.2 Constructor & Destructor Documentation . . . . .     | 102 |
| 6.29.3 Member Function Documentation . . . . .              | 102 |
| 6.29.4 Member Data Documentation . . . . .                  | 103 |
| 6.30 CglPreProcess Class Reference . . . . .                | 103 |
| 6.30.1 Detailed Description . . . . .                       | 106 |
| 6.30.2 Constructor & Destructor Documentation . . . . .     | 106 |
| 6.30.3 Member Function Documentation . . . . .              | 106 |
| 6.31 CglProbing Class Reference . . . . .                   | 111 |
| 6.31.1 Detailed Description . . . . .                       | 113 |
| 6.31.2 Constructor & Destructor Documentation . . . . .     | 114 |
| 6.31.3 Member Function Documentation . . . . .              | 114 |
| 6.31.4 Friends And Related Function Documentation . . . . . | 118 |
| 6.32 CglRedSplit Class Reference . . . . .                  | 118 |
| 6.32.1 Detailed Description . . . . .                       | 120 |
| 6.32.2 Constructor & Destructor Documentation . . . . .     | 120 |
| 6.32.3 Member Function Documentation . . . . .              | 121 |
| 6.32.4 Friends And Related Function Documentation . . . . . | 123 |
| 6.33 CglRedSplit2 Class Reference . . . . .                 | 124 |
| 6.33.1 Detailed Description . . . . .                       | 125 |
| 6.33.2 Constructor & Destructor Documentation . . . . .     | 125 |
| 6.33.3 Member Function Documentation . . . . .              | 125 |
| 6.33.4 Friends And Related Function Documentation . . . . . | 126 |
| 6.34 CglRedSplit2Param Class Reference . . . . .            | 127 |
| 6.34.1 Detailed Description . . . . .                       | 131 |
| 6.34.2 Member Enumeration Documentation . . . . .           | 132 |
| 6.34.3 Constructor & Destructor Documentation . . . . .     | 134 |
| 6.34.4 Member Function Documentation . . . . .              | 134 |
| 6.34.5 Member Data Documentation . . . . .                  | 140 |
| 6.35 CglRedSplitParam Class Reference . . . . .             | 142 |
| 6.35.1 Detailed Description . . . . .                       | 145 |
| 6.35.2 Constructor & Destructor Documentation . . . . .     | 145 |
| 6.35.3 Member Function Documentation . . . . .              | 146 |
| 6.35.4 Member Data Documentation . . . . .                  | 148 |
| 6.36 CglResidualCapacity Class Reference . . . . .          | 150 |
| 6.36.1 Detailed Description . . . . .                       | 151 |
| 6.36.2 Constructor & Destructor Documentation . . . . .     | 151 |

|        |  |     |
|--------|--|-----|
| 6.36.3 | Member Function Documentation              | 152 |
| 6.36.4 | Friends And Related Function Documentation | 153 |
| 6.37   | CglSimpleRounding Class Reference          | 153 |
| 6.37.1 | Detailed Description                       | 154 |
| 6.37.2 | Constructor & Destructor Documentation     | 154 |
| 6.37.3 | Member Function Documentation              | 154 |
| 6.37.4 | Friends And Related Function Documentation | 155 |
| 6.38   | CglStored Class Reference                  | 155 |
| 6.38.1 | Detailed Description                       | 157 |
| 6.38.2 | Constructor & Destructor Documentation     | 157 |
| 6.38.3 | Member Function Documentation              | 157 |
| 6.38.4 | Member Data Documentation                  | 159 |
| 6.39   | CglTreeInfo Class Reference                | 159 |
| 6.39.1 | Detailed Description                       | 160 |
| 6.39.2 | Constructor & Destructor Documentation     | 160 |
| 6.39.3 | Member Function Documentation              | 161 |
| 6.39.4 | Member Data Documentation                  | 161 |
| 6.40   | CglTreeProbingInfo Class Reference         | 162 |
| 6.40.1 | Detailed Description                       | 164 |
| 6.40.2 | Constructor & Destructor Documentation     | 164 |
| 6.40.3 | Member Function Documentation              | 164 |
| 6.40.4 | Member Data Documentation                  | 166 |
| 6.41   | CglTwomir Class Reference                  | 167 |
| 6.41.1 | Detailed Description                       | 169 |
| 6.41.2 | Constructor & Destructor Documentation     | 169 |
| 6.41.3 | Member Function Documentation              | 169 |
| 6.41.4 | Friends And Related Function Documentation | 172 |
| 6.41.5 | Member Data Documentation                  | 172 |
| 6.42   | CglUniqueRowCuts Class Reference           | 172 |
| 6.42.1 | Detailed Description                       | 172 |
| 6.42.2 | Constructor & Destructor Documentation     | 173 |
| 6.42.3 | Member Function Documentation              | 173 |
| 6.43   | CglZeroHalf Class Reference                | 173 |
| 6.43.1 | Detailed Description                       | 174 |
| 6.43.2 | Constructor & Destructor Documentation     | 174 |
| 6.43.3 | Member Function Documentation              | 175 |
| 6.43.4 | Friends And Related Function Documentation | 175 |

|        |  |     |
|--------|--|-----|
| 6.44   | cliqueEntry Struct Reference           | 176 |
| 6.44.1 | Detailed Description                   | 176 |
| 6.44.2 | Member Data Documentation              | 176 |
| 6.45   | cut Struct Reference                   | 176 |
| 6.45.1 | Detailed Description                   | 177 |
| 6.45.2 | Member Data Documentation              | 177 |
| 6.46   | cut_list Struct Reference              | 177 |
| 6.46.1 | Detailed Description                   | 178 |
| 6.46.2 | Member Data Documentation              | 178 |
| 6.47   | cutParams Struct Reference             | 178 |
| 6.47.1 | Detailed Description                   | 178 |
| 6.47.2 | Member Data Documentation              | 178 |
| 6.48   | LAP::Cuts Struct Reference             | 179 |
| 6.48.1 | Detailed Description                   | 179 |
| 6.48.2 | Constructor & Destructor Documentation | 179 |
| 6.48.3 | Member Function Documentation          | 180 |
| 6.49   | cycle Struct Reference                 | 180 |
| 6.49.1 | Detailed Description                   | 180 |
| 6.49.2 | Member Data Documentation              | 180 |
| 6.50   | cycle_list Struct Reference            | 181 |
| 6.50.1 | Detailed Description                   | 181 |
| 6.50.2 | Member Data Documentation              | 181 |
| 6.51   | DGG_constraint_t Struct Reference      | 181 |
| 6.51.1 | Detailed Description                   | 182 |
| 6.51.2 | Member Data Documentation              | 182 |
| 6.52   | DGG_data_t Struct Reference            | 182 |
| 6.52.1 | Detailed Description                   | 183 |
| 6.52.2 | Member Data Documentation              | 183 |
| 6.53   | DGG_list_t Struct Reference            | 184 |
| 6.53.1 | Detailed Description                   | 184 |
| 6.53.2 | Member Data Documentation              | 184 |
| 6.54   | disaggregationAction Struct Reference  | 185 |
| 6.54.1 | Detailed Description                   | 185 |
| 6.54.2 | Member Data Documentation              | 185 |
| 6.55   | edge Struct Reference                  | 185 |
| 6.55.1 | Detailed Description                   | 185 |
| 6.55.2 | Member Data Documentation              | 185 |



|        |  |     |
|--------|--|-----|
| 6.56   | ilp Struct Reference                   | 186 |
| 6.56.1 | Detailed Description                   | 186 |
| 6.56.2 | Member Data Documentation              | 186 |
| 6.57   | info_weak Struct Reference             | 187 |
| 6.57.1 | Detailed Description                   | 188 |
| 6.57.2 | Member Data Documentation              | 188 |
| 6.58   | LAP::LandPMessages Class Reference     | 188 |
| 6.58.1 | Detailed Description                   | 188 |
| 6.58.2 | Constructor & Destructor Documentation | 189 |
| 6.59   | LAP::LapMessages Class Reference       | 189 |
| 6.59.1 | Detailed Description                   | 189 |
| 6.59.2 | Constructor & Destructor Documentation | 189 |
| 6.60   | log_var Struct Reference               | 190 |
| 6.60.1 | Detailed Description                   | 190 |
| 6.60.2 | Member Data Documentation              | 190 |
| 6.61   | CglLandP::NoBasisError Class Reference | 190 |
| 6.61.1 | Detailed Description                   | 190 |
| 6.61.2 | Constructor & Destructor Documentation | 190 |
| 6.62   | CglLandP::Parameters Class Reference   | 191 |
| 6.62.1 | Detailed Description                   | 192 |
| 6.62.2 | Constructor & Destructor Documentation | 192 |
| 6.62.3 | Member Function Documentation          | 192 |
| 6.62.4 | Member Data Documentation              | 193 |
| 6.63   | parity_ilp Struct Reference            | 195 |
| 6.63.1 | Detailed Description                   | 195 |
| 6.63.2 | Member Data Documentation              | 195 |
| 6.64   | pool_cut Struct Reference              | 197 |
| 6.64.1 | Detailed Description                   | 197 |
| 6.64.2 | Member Data Documentation              | 197 |
| 6.65   | pool_cut_list Struct Reference         | 198 |
| 6.65.1 | Detailed Description                   | 198 |
| 6.65.2 | Member Data Documentation              | 198 |
| 6.66   | select_cut Struct Reference            | 198 |
| 6.66.1 | Detailed Description                   | 199 |
| 6.66.2 | Member Data Documentation              | 199 |
| 6.67   | separation_graph Struct Reference      | 199 |
| 6.67.1 | Detailed Description                   | 199 |

|          |  |            |
|----------|--|------------|
| 6.67.2   | Member Data Documentation                                | 199        |
| 6.68     | short_path_node Struct Reference                         | 200        |
| 6.68.1   | Detailed Description                                     | 200        |
| 6.68.2   | Member Data Documentation                                | 200        |
| 6.69     | CglLandP::SimplexInterfaceError Class Reference          | 200        |
| 6.69.1   | Detailed Description                                     | 201        |
| 6.69.2   | Constructor & Destructor Documentation                   | 201        |
| 6.70     | LAP::TabRow Struct Reference                             | 201        |
| 6.70.1   | Detailed Description                                     | 202        |
| 6.70.2   | Constructor & Destructor Documentation                   | 202        |
| 6.70.3   | Member Function Documentation                            | 202        |
| 6.70.4   | Member Data Documentation                                | 203        |
| 6.71     | LAP::Validator Class Reference                           | 203        |
| 6.71.1   | Detailed Description                                     | 204        |
| 6.71.2   | Member Enumeration Documentation                         | 204        |
| 6.71.3   | Constructor & Destructor Documentation                   | 204        |
| 6.71.4   | Member Function Documentation                            | 205        |
| <b>7</b> | <b>File Documentation</b>                                | <b>206</b> |
| 7.1      | src/CglAllDifferent/CglAllDifferent.hpp File Reference   | 206        |
| 7.2      | src/CglClique/CglClique.hpp File Reference               | 206        |
| 7.2.1    | Function Documentation                                   | 206        |
| 7.3      | src/CglConfig.h File Reference                           | 207        |
| 7.4      | src/CglCutGenerator.hpp File Reference                   | 207        |
| 7.5      | src/CglDuplicateRow/CglDuplicateRow.hpp File Reference   | 207        |
| 7.6      | src/CglFlowCover/CglFlowCover.hpp File Reference         | 207        |
| 7.6.1    | Typedef Documentation                                    | 208        |
| 7.6.2    | Enumeration Type Documentation                           | 208        |
| 7.6.3    | Function Documentation                                   | 209        |
| 7.7      | src/CglGMI/CglGMI.hpp File Reference                     | 210        |
| 7.7.1    | Function Documentation                                   | 210        |
| 7.8      | src/CglGMI/CglGMIParam.hpp File Reference                | 210        |
| 7.9      | src/CglGomory/CglGomory.hpp File Reference               | 211        |
| 7.9.1    | Function Documentation                                   | 211        |
| 7.10     | src/CglKnapsackCover/CglKnapsackCover.hpp File Reference | 211        |
| 7.10.1   | Function Documentation                                   | 211        |
| 7.11     | src/CglLandP/CglLandP.hpp File Reference                 | 212        |

|        |  |     |
|--------|--|-----|
| 7.11.1 | Function Documentation   | 212 |
| 7.12   | src/CgLLandP/CgLLandPMessages.hpp File Reference                         | 212 |
| 7.13   | src/CgLLandP/CgLLandPSimplex.hpp File Reference                          | 213 |
| 7.13.1 | Macro Definition Documentation   | 214 |
| 7.14   | src/CgLLandP/CgLLandPTabRow.hpp File Reference                           | 214 |
| 7.15   | src/CgLLandP/CgLLandPUtills.hpp File Reference                           | 214 |
| 7.16   | src/CgLLandP/CgLLandPValidator.hpp File Reference                        | 215 |
| 7.17   | src/CgLLiftAndProject/CgLLiftAndProject.hpp File Reference               | 215 |
| 7.17.1 | Function Documentation   | 216 |
| 7.18   | src/CgLMessage.hpp File Reference  | 216 |
| 7.18.1 | Enumeration Type Documentation   | 216 |
| 7.19   | src/CgLMixedIntegerRounding/CgLMixedIntegerRounding.hpp File Reference   | 217 |
| 7.19.1 | Macro Definition Documentation   | 217 |
| 7.19.2 | Typedef Documentation  | 217 |
| 7.19.3 | Function Documentation   | 218 |
| 7.20   | src/CgLMixedIntegerRounding2/CgLMixedIntegerRounding2.hpp File Reference | 218 |
| 7.20.1 | Macro Definition Documentation   | 218 |
| 7.20.2 | Typedef Documentation  | 218 |
| 7.20.3 | Function Documentation   | 218 |
| 7.21   | src/CgLOddHole/CgLOddHole.hpp File Reference                             | 219 |
| 7.21.1 | Function Documentation   | 219 |
| 7.22   | src/CgLParam.hpp File Reference  | 219 |
| 7.23   | src/CgLPreProcess/CgLPreProcess.hpp File Reference                       | 219 |
| 7.24   | src/CgLProbing/CgLProbing.hpp File Reference                             | 220 |
| 7.24.1 | Function Documentation   | 220 |
| 7.25   | src/CgLRedSplit/CgLRedSplit.hpp File Reference                           | 221 |
| 7.25.1 | Function Documentation   | 221 |
| 7.26   | src/CgLRedSplit/CgLRedSplitParam.hpp File Reference                      | 222 |
| 7.27   | src/CgLRedSplit2/CgLRedSplit2.hpp File Reference                         | 222 |
| 7.27.1 | Function Documentation   | 222 |
| 7.28   | src/CgLRedSplit2/CgLRedSplit2Param.hpp File Reference                    | 222 |
| 7.29   | src/CgLResidualCapacity/CgLResidualCapacity.hpp File Reference           | 223 |
| 7.29.1 | Macro Definition Documentation   | 223 |
| 7.29.2 | Function Documentation   | 223 |
| 7.30   | src/CgLSimpleRounding/CgLSimpleRounding.hpp File Reference               | 223 |
| 7.30.1 | Function Documentation   | 224 |
| 7.31   | src/CgLStored.hpp File Reference   | 224 |

|        |  |     |
|--------|--|-----|
| 7.32   | <a href="#">src/CglTreeInfo.hpp File Reference</a>             | 224 |
| 7.32.1 | <a href="#">Function Documentation</a>                         | 225 |
| 7.33   | <a href="#">src/CglTwomir/CglTwomir.hpp File Reference</a>     | 225 |
| 7.33.1 | <a href="#">Macro Definition Documentation</a>                 | 228 |
| 7.33.2 | <a href="#">Function Documentation</a>                         | 234 |
| 7.34   | <a href="#">src/CglZeroHalf/Cgl012cut.hpp File Reference</a>   | 235 |
| 7.34.1 | <a href="#">Macro Definition Documentation</a>                 | 236 |
| 7.35   | <a href="#">src/CglZeroHalf/CglZeroHalf.hpp File Reference</a> | 236 |
| 7.35.1 | <a href="#">Function Documentation</a>                         | 237 |
| 7.36   | <a href="#">src/config_cgl_default.h File Reference</a>        | 237 |
| 7.36.1 | <a href="#">Macro Definition Documentation</a>                 | 237 |
| 7.37   | <a href="#">src/config_default.h File Reference</a>            | 238 |
| 7.37.1 | <a href="#">Macro Definition Documentation</a>                 | 238 |

## Index 239

### 1 Namespace Index

#### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

##### LAP

Performs one round of Lift & Project using [CglLandPSimplex](#) to build cuts 12

### 2 Hierarchical Index

#### 2.1 Class Hierarchy

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

```
std::allocator< T >
std::array< T >
std::auto_ptr< T >
```

##### auxiliary\_graph

14

```
std::basic_string< Char >
    std::string
    std::wstring
std::basic_string< char >
std::basic_string< wchar_t >
std::bitset< Bits >
```

##### Cgl012Cut

15

|                                       |     |
|---------------------------------------|-----|
| <code>cgl_arc</code>                  | 17  |
| <code>cgl_graph</code>                | 17  |
| <code>cgl_node</code>                 | 18  |
| <code>CglIBK</code>                   | 21  |
| <code>CglCutGenerator</code>          | 30  |
| <code>CglAllDifferent</code>          | 19  |
| <code>CglClique</code>                | 23  |
| <code>CglFakeClique</code>            | 40  |
| <code>CglDuplicateRow</code>          | 34  |
| <code>CglFlowCover</code>             | 42  |
| <code>CglGMI</code>                   | 47  |
| <code>CglGomory</code>                | 62  |
| <code>CglImplication</code>           | 68  |
| <code>CglKnapsackCover</code>         | 70  |
| <code>CglLandP</code>                 | 73  |
| <code>CglLiftAndProject</code>        | 85  |
| <code>CglMixedIntegerRounding</code>  | 88  |
| <code>CglMixedIntegerRounding2</code> | 91  |
| <code>CglOddHole</code>               | 97  |
| <code>CglProbing</code>               | 111 |
| <code>CglRedSplit</code>              | 118 |
| <code>CglRedSplit2</code>             | 124 |
| <code>CglResidualCapacity</code>      | 150 |
| <code>CglSimpleRounding</code>        | 153 |
| <code>CglStored</code>                | 155 |
| <code>CglTwomir</code>                | 167 |
| <code>CglZeroHalf</code>              | 173 |
| <code>CglFlowVUB</code>               | 45  |
| <code>CglHashLink</code>              | 68  |
| <code>LAP::CglLandPSimplex</code>     | 77  |

|   |            |
|---|------------|
| <b>CglMixIntRoundVUB</b>                        | <b>94</b>  |
| <b>CglMixIntRoundVUB2</b>                       | <b>96</b>  |
| <b>CglParam</b>                                 | <b>100</b> |
| <b>CglGMIParam</b>                              | <b>51</b>  |
| <b>CglLandP::Parameters</b>                     | <b>191</b> |
| <b>CglRedSplit2Param</b>                        | <b>127</b> |
| <b>CglRedSplitParam</b>                         | <b>142</b> |
| <b>CglPreProcess</b>                            | <b>103</b> |
| <b>CglTreeInfo</b>                              | <b>159</b> |
| <b>CglTreeProbingInfo</b>                       | <b>162</b> |
| <b>CglUniqueRowCuts</b>                         | <b>172</b> |
| <b>cliqueEntry</b>                              | <b>176</b> |
| CoinError                                       |            |
| <b>CglLandP::NoBasisError</b>                   | <b>190</b> |
| <b>CglLandP::SimplexInterfaceError</b>          | <b>200</b> |
| CoinIndexedVector                               |            |
| <b>LAP::TabRow</b>                              | <b>201</b> |
| CoinMessages                                    |            |
| <b>CglMessage</b>                               | <b>87</b>  |
| <b>LAP::LandPMessages</b>                       | <b>188</b> |
| <b>LAP::LapMessages</b>                         | <b>189</b> |
| std::complex                                    |            |
| std::deque< T >::const_iterator                 |            |
| std::list< T >::const_iterator                  |            |
| std::forward_list< T >::const_iterator          |            |
| std::map< K, T >::const_iterator                |            |
| std::unordered_map< K, T >::const_iterator      |            |
| std::basic_string< Char >::const_iterator       |            |
| std::multimap< K, T >::const_iterator           |            |
| std::unordered_multimap< K, T >::const_iterator |            |
| std::set< K >::const_iterator                   |            |
| std::string::const_iterator                     |            |
| std::unordered_set< K >::const_iterator         |            |
| std::multiset< K >::const_iterator              |            |
| std::wstring::const_iterator                    |            |
| std::unordered_multiset< K >::const_iterator    |            |
| std::vector< T >::const_iterator                |            |
| std::list< T >::const_reverse_iterator          |            |
| std::forward_list< T >::const_reverse_iterator  |            |
| std::map< K, T >::const_reverse_iterator        |            |

|   |            |
|---|------------|
| std::basic_string< Char >::const_reverse_iterator       |            |
| std::unordered_map< K, 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::string::const_reverse_iterator                     |            |
| std::unordered_set< K >::const_reverse_iterator         |            |
| std::multiset< K >::const_reverse_iterator              |            |
| std::unordered_multiset< K >::const_reverse_iterator    |            |
| std::wstring::const_reverse_iterator                    |            |
| std::vector< T >::const_reverse_iterator                |            |
| std::deque< T >::const_reverse_iterator                 |            |
| <b>cut</b>  | <b>176</b> |
| <b>cut_list</b>   | <b>177</b> |
| <b>cutParams</b>  | <b>178</b> |
| <b>LAP::Cuts</b>  | <b>179</b> |
| <b>cycle</b>  | <b>180</b> |
| <b>cycle_list</b>                                       | <b>181</b> |
| std::deque< T >   |            |
| <b>DGG_constraint_t</b>                                 | <b>181</b> |
| <b>DGG_data_t</b>                                       | <b>182</b> |
| <b>DGG_list_t</b>                                       | <b>184</b> |
| <b>disaggregationAction</b>                             | <b>185</b> |
| <b>edge</b>   | <b>185</b> |
| 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                                    |            |
| std::forward_list< T >                                  |            |
| <b>ilp</b>  | <b>186</b> |

**info\_weak**

187

```
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_ostreamstream< wchar_t >
std::basic_ostream
  std::basic_ofstream
  std::ofstream
  std::wofstream
  std::basic_ostreamstream
  std::ostreamstream
  std::wostringstream
std::ostream
  std::wostream
std::ios
  std::wios
std::list< T >::iterator
```



std::unordered\_set< K >::iterator  
 std::forward\_list< T >::iterator  
 std::map< K, T >::iterator  
 std::string::iterator  
 std::unordered\_map< K, T >::iterator  
 std::multimap< K, T >::iterator  
 std::basic\_string< Char >::iterator  
 std::unordered\_multimap< K, T >::iterator  
 std::set< K >::iterator  
 std::wstring::iterator  
 std::multiset< K >::iterator  
 std::unordered\_multiset< K >::iterator  
 std::deque< T >::iterator  
 std::vector< T >::iterator  
 std::list< T >

## **log\_var** 190

std::map< K, T >  
 std::multimap< K, T >  
 std::multiset< K >

## **parity\_ilp** 195

## **pool\_cut** 197

## **pool\_cut\_list** 198

std::priority\_queue< T >  
 std::queue< T >  
 std::forward\_list< T >::reverse\_iterator  
 std::map< K, T >::reverse\_iterator  
 std::basic\_string< Char >::reverse\_iterator  
 std::unordered\_multimap< K, T >::reverse\_iterator  
 std::string::reverse\_iterator  
 std::multimap< K, T >::reverse\_iterator  
 std::list< T >::reverse\_iterator  
 std::unordered\_set< K >::reverse\_iterator  
 std::vector< T >::reverse\_iterator  
 std::deque< T >::reverse\_iterator  
 std::set< K >::reverse\_iterator  
 std::wstring::reverse\_iterator  
 std::multiset< K >::reverse\_iterator  
 std::unordered\_multiset< K >::reverse\_iterator  
 std::unordered\_map< K, T >::reverse\_iterator

## **select\_cut** 198

## **separation\_graph** 199

std::set< K >

## **short\_path\_node** 200

std::smart\_ptr< T >  
 std::stack< T >  
 std::system\_error  
 std::thread  
 std::unique\_ptr< T >

```

std::unordered_map< K, T >
std::unordered_multimap< K, T >
std::unordered_multiset< K >
std::unordered_set< K >
std::valarray< T >

```

**LAP::Validator****203**

```

std::vector< T >
std::vector< bool >
std::vector< ColumnSelectionStrategy >
std::vector< double >
std::vector< int >
std::vector< OsiRowCut * >
std::vector< RowSelectionStrategy >
std::vector< std::string >
std::weak_ptr< T >
K
T

```

### 3 Class Index

#### 3.1 Class List

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

|  |    |
|--|----|
| <a href="#">auxiliary_graph</a>                            | 14 |
| <a href="#">Cgl012Cut</a>                                  |    |
| 012Cut Generator Class                                     | 15 |
| <a href="#">cgl_arc</a>                                    | 17 |
| <a href="#">cgl_graph</a>                                  | 17 |
| <a href="#">cgl_node</a>                                   | 18 |
| <a href="#">CglAllDifferent</a>                            |    |
| AllDifferent Cut Generator Class This has a number of sets | 19 |
| <a href="#">CglBK</a>                                      |    |
| For Bron-Kerbosch  | 21 |
| <a href="#">CglClique</a>                                  | 23 |
| <a href="#">CglCutGenerator</a>                            |    |
| Cut Generator Base Class                                   | 30 |
| <a href="#">CglDuplicateRow</a>                            |    |
| DuplicateRow Cut Generator Class                           | 34 |
| <a href="#">CglFakeClique</a>                              | 40 |
| <a href="#">CglFlowCover</a>                               |    |
| Lifed Simple Generalized Flow Cover Cut Generator Class    | 42 |

|  |     |
|--|-----|
| <b>CglFlowVUB</b>  |     |
| Variable upper bound class   | 45  |
| <b>CglGMI</b>  |     |
| Gomory cut generator with several cleaning procedures, used to test the numerical safety of the resulting cuts | 47  |
| <b>CglGMIParam</b>   |     |
| Class collecting parameters for the GMI cut generator  | 51  |
| <b>CglGomory</b>   |     |
| Gomory Cut Generator Class   | 62  |
| <b>CglHashLink</b>   |     |
| Only store unique row cuts   | 68  |
| <b>CglImplication</b>  |     |
| This just uses implication info  | 68  |
| <b>CglKnapsackCover</b>  |     |
| Knapsack Cover Cut Generator Class   | 70  |
| <b>CglLandP</b>  | 73  |
| <b>LAP::CglLandPSimplex</b>  | 77  |
| <b>CglLiftAndProject</b>   |     |
| Lift And Project Cut Generator Class   | 85  |
| <b>CglMessage</b>  |     |
| This deals with Cgl messages (as against Osi messages etc)   | 87  |
| <b>CglMixedIntegerRounding</b>   |     |
| Mixed Integer Rounding Cut Generator Class   | 88  |
| <b>CglMixedIntegerRounding2</b>  |     |
| Mixed Integer Rounding Cut Generator Class   | 91  |
| <b>CglMixIntRoundVUB</b>   | 94  |
| <b>CglMixIntRoundVUB2</b>  | 96  |
| <b>CglOddHole</b>  |     |
| Odd Hole Cut Generator Class   | 97  |
| <b>CglParam</b>  |     |
| Class collecting parameters for all cut generators   | 100 |
| <b>CglPreProcess</b>   |     |
| Class for preProcessing and postProcessing   | 103 |
| <b>CglProbing</b>  |     |
| Probing Cut Generator Class  | 111 |
| <b>CglRedSplit</b>   |     |
| Gomory Reduce-and-Split Cut Generator Class; See method <a href="#">generateCuts()</a>                         | 118 |

|                                      |  |     |
|--------------------------------------|--|-----|
| <a href="#">CglRedSplit2</a>         | Reduce-and-Split Cut Generator Class; See method <a href="#">generateCuts()</a>  | 124 |
| <a href="#">CglRedSplit2Param</a>    | Class collecting parameters the Reduced-and-split cut generator  | 127 |
| <a href="#">CglRedSplitParam</a>     | Class collecting parameters the Reduced-and-split cut generator  | 142 |
| <a href="#">CglResidualCapacity</a>  | Residual Capacity Inequalities Cut Generator Class   | 150 |
| <a href="#">CglSimpleRounding</a>    | Simple Rounding Cut Generator Class  | 153 |
| <a href="#">CglStored</a>            | Stored Cut Generator Class   | 155 |
| <a href="#">CglTreeInfo</a>          | Information about where the cut generator is invoked from  | 159 |
| <a href="#">CglTreeProbingInfo</a>   |  | 162 |
| <a href="#">CglTwomir</a>            | Twostep MIR Cut Generator Class  | 167 |
| <a href="#">CglUniqueRowCuts</a>     |  | 172 |
| <a href="#">CglZeroHalf</a>          | Zero Half Cut Generator Class  | 173 |
| <a href="#">cliqueEntry</a>          | Derived class to pick up probing info  | 176 |
| <a href="#">cut</a>                  |  | 176 |
| <a href="#">cut_list</a>             |  | 177 |
| <a href="#">cutParams</a>            |  | 178 |
| <a href="#">LAP::Cuts</a>            | To store extra cuts generated by columns from which they origin  | 179 |
| <a href="#">cycle</a>                |  | 180 |
| <a href="#">cycle_list</a>           |  | 181 |
| <a href="#">DGG_constraint_t</a>     |  | 181 |
| <a href="#">DGG_data_t</a>           |  | 182 |
| <a href="#">DGG_list_t</a>           |  | 184 |
| <a href="#">disaggregationAction</a> | Only useful type of disaggregation is most normal For now just done for 0-1 variables Can be used for building cliques | 185 |
| <a href="#">edge</a>                 |  | 185 |

|   |     |
|---|-----|
| <a href="#">ilp</a>                             | 186 |
| <a href="#">info_weak</a>                       | 187 |
| <a href="#">LAP::LandPMessages</a>              |     |
| Message handler for lift-and-project simplex    | 188 |
| <a href="#">LAP::LapMessages</a>                |     |
| Output messages for Cgl                         | 189 |
| <a href="#">log_var</a>                         | 190 |
| <a href="#">CglLandP::NoBasisError</a>          | 190 |
| <a href="#">CglLandP::Parameters</a>            |     |
| Class storing parameters                        | 191 |
| <a href="#">parity_ilp</a>                      | 195 |
| <a href="#">pool_cut</a>                        | 197 |
| <a href="#">pool_cut_list</a>                   | 198 |
| <a href="#">select_cut</a>                      | 198 |
| <a href="#">separation_graph</a>                | 199 |
| <a href="#">short_path_node</a>                 | 200 |
| <a href="#">CglLandP::SimplexInterfaceError</a> | 200 |
| <a href="#">LAP::TabRow</a>                     | 201 |
| <a href="#">LAP::Validator</a>                  |     |
| Class to validate or reject a cut               | 203 |

## 4 File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

|  |     |
|--|-----|
| <a href="#">src/CglConfig.h</a>          | 207 |
| <a href="#">src/CglCutGenerator.hpp</a>  | 207 |
| <a href="#">src/CglMessage.hpp</a>       | 216 |
| <a href="#">src/CglParam.hpp</a>         | 219 |
| <a href="#">src/CglStored.hpp</a>        | 224 |
| <a href="#">src/CglTreeInfo.hpp</a>      | 224 |
| <a href="#">src/config_cgl_default.h</a> | 237 |

|   |     |
|---|-----|
| src/config_default.h                                      | 238 |
| src/CglAllDifferent/CglAllDifferent.hpp                   | 206 |
| src/CglClique/CglClique.hpp                               | 206 |
| src/CglDuplicateRow/CglDuplicateRow.hpp                   | 207 |
| src/CglFlowCover/CglFlowCover.hpp                         | 207 |
| src/CglGMI/CglGMI.hpp                                     | 210 |
| src/CglGMI/CglGMIParam.hpp                                | 210 |
| src/CglGomory/CglGomory.hpp                               | 211 |
| src/CglKnapsackCover/CglKnapsackCover.hpp                 | 211 |
| src/CglLandP/CglLandP.hpp                                 | 212 |
| src/CglLandP/CglLandPMessages.hpp                         | 212 |
| src/CglLandP/CglLandPSimplex.hpp                          | 213 |
| src/CglLandP/CglLandPTabRow.hpp                           | 214 |
| src/CglLandP/CglLandPUtils.hpp                            | 214 |
| src/CglLandP/CglLandPValidator.hpp                        | 215 |
| src/CglLiftAndProject/CglLiftAndProject.hpp               | 215 |
| src/CglMixedIntegerRounding/CglMixedIntegerRounding.hpp   | 217 |
| src/CglMixedIntegerRounding2/CglMixedIntegerRounding2.hpp | 218 |
| src/CglOddHole/CglOddHole.hpp                             | 219 |
| src/CglPreProcess/CglPreProcess.hpp                       | 219 |
| src/CglProbing/CglProbing.hpp                             | 220 |
| src/CglRedSplit/CglRedSplit.hpp                           | 221 |
| src/CglRedSplit/CglRedSplitParam.hpp                      | 222 |
| src/CglRedSplit2/CglRedSplit2.hpp                         | 222 |
| src/CglRedSplit2/CglRedSplit2Param.hpp                    | 222 |
| src/CglResidualCapacity/CglResidualCapacity.hpp           | 223 |
| src/CglSimpleRounding/CglSimpleRounding.hpp               | 223 |
| src/CglTwomir/CglTwomir.hpp                               | 225 |
| src/CglZeroHalf/Cgl012cut.hpp                             | 235 |
| src/CglZeroHalf/CglZeroHalf.hpp                           | 236 |

## 5 Namespace Documentation

### 5.1 LAP Namespace Reference

Performs one round of Lift & Project using [CglLandPSimplex](#) to build cuts.

#### Classes

- class [LapMessages](#)  
*Output messages for Cgl.*
- class [LandPMessages](#)  
*Message handler for lift-and-project simplex.*
- class [CglLandPSimplex](#)
- struct [TabRow](#)
- struct [Cuts](#)  
*To store extra cuts generated by columns from which they origin.*
- class [Validator](#)  
*Class to validate or reject a cut.*

#### Enumerations

- enum [LapMessagesTypes](#) {  
[BEGIN\\_ROUND](#), [END\\_ROUND](#), [DURING\\_SEP](#), [CUT\\_REJECTED](#),  
[CUT\\_FAILED](#), [CUT\\_GAP](#), [LAP\\_CUT\\_FAILED\\_DO\\_MIG](#), [LAP\\_MESSAGES\\_DUMMY\\_END](#) }
- enum [LAP\\_messages](#) {  
[Separating](#), [FoundImprovingRow](#), [FoundBestImprovingCol](#), [WarnFailedBestImprovingCol](#),  
[LogHead](#), [PivotLog](#), [FinishedOptimal](#), [HitLimit](#),  
[NumberNegRc](#), [NumberZeroRc](#), [NumberPosRc](#), [WeightsStats](#),  
[WarnBadSigmaComputation](#), [WarnBadRowComputation](#), [WarnGiveUpRow](#), [PivotFailedSigmaUnchanged](#),  
[PivotFailedSigmaIncreased](#), [FailedSigmaIncreased](#), [WarnBadRhsComputation](#), [WarnFailedPivotTol](#),  
[WarnFailedPivotIf](#), [RoundStats](#), [CutStat](#), [DUMMY\\_END](#) }  
*Types of messages for lift-and-project simplex.*

#### Functions

- double [normCoef](#) ([TabRow](#) &row, int ncols, const int \*nonBasics)  
*Compute  $\sum_{j=1}^n |a_{ij}| |1 - a_{i0}|$  for row passed as argument.*
- void [scale](#) ([OsiRowCut](#) &cut)  
*scale the cut passed as argument*
- void [scale](#) ([OsiRowCut](#) &cut, double norma)  
*scale the cut passed as argument using provided normalization factor*
- void [modularizeRow](#) ([TabRow](#) &row, const bool \*integerVar)  
*Modularize row.*
- double [intersectionCutCoef](#) (double alpha\_i, double beta)  
*return the coefficients of the intersection cut*
- double [modularizedCoef](#) (double alpha, double beta)  
*compute the modularized row coefficient for an integer variable*
- bool [int\\_val](#) (double value, double tol)  
*Says is value is integer.*

## 5.1.1 Detailed Description

Performs one round of Lift & Project using [CglLandPSimplex](#) to build cuts. constants describing rejection codes

## 5.1.2 Enumeration Type Documentation

## 5.1.2.1 enum LAP::LapMessagesTypes

Enumerator

***BEGIN\_ROUND***  
***END\_ROUND***  
***DURING\_SEP***  
***CUT\_REJECTED***  
***CUT\_FAILED***  
***CUT\_GAP***  
***LAP\_CUT\_FAILED\_DO\_MIG***  
***LAP\_MESSAGES\_DUMMY\_END***

Definition at line 26 of file CglLandP.hpp.

## 5.1.2.2 enum LAP::LAP\_messages

Types of messages for lift-and-project simplex.

Enumerator

***Separating***  
***FoundImprovingRow***  
***FoundBestImprovingCol***  
***WarnFailedBestImprovingCol***  
***LogHead***  
***PivotLog***  
***FinishedOptimal***  
***HitLimit***  
***NumberNegRc***  
***NumberZeroRc***  
***NumberPosRc***  
***WeightsStats***  
***WarnBadSigmaComputation***  
***WarnBadRowComputation***  
***WarnGiveUpRow***  
***PivotFailedSigmaUnchanged***  
***PivotFailedSigmaIncreased***  
***FailedSigmaIncreased***  
***WarnBadRhsComputation***  
***WarnFailedPivotTol***



***WarnFailedPivotIf***

***RoundStats***

***CutStat***

***DUMMY\_END***

Definition at line 22 of file CglLandPMessages.hpp.

### 5.1.3 Function Documentation

5.1.3.1 `double LAP::normCoef ( TabRow & row, int ncols, const int * nonBasics )`

Compute  $\sum_{j=1}^n |a_{ij}| |1 - a_{i0}|$  for row passed as argument.

5.1.3.2 `void LAP::scale ( OsiRowCut & cut )`

scale the cut passed as argument

5.1.3.3 `void LAP::scale ( OsiRowCut & cut, double norma )`

scale the cut passed as argument using provided normalization factor

5.1.3.4 `void LAP::modularizeRow ( TabRow & row, const bool * integerVar )`

Modularize row.

5.1.3.5 `double LAP::intersectionCutCoef ( double alpha_i, double beta )` `[inline]`

return the coefficients of the intersection cut

Definition at line 35 of file CglLandPUtils.hpp.

5.1.3.6 `double LAP::modularizedCoef ( double alpha, double beta )` `[inline]`

compute the modularized row coefficient for an integer variable

Definition at line 42 of file CglLandPUtils.hpp.

5.1.3.7 `bool LAP::int_val ( double value, double tol )` `[inline]`

Says is value is integer.

Definition at line 52 of file CglLandPUtils.hpp.

## 6 Class Documentation

### 6.1 auxiliary\_graph Struct Reference

```
#include <Cgl012cut.hpp>
```

#### Public Attributes

- int `nnodes`
- int `narcs`

- [cgl\\_node](#) \* nodes
- [cgl\\_arc](#) \* arcs

### 6.1.1 Detailed Description

Definition at line 129 of file Cgl012cut.hpp.

### 6.1.2 Member Data Documentation

#### 6.1.2.1 int auxiliary\_graph::nnodes

Definition at line 130 of file Cgl012cut.hpp.

#### 6.1.2.2 int auxiliary\_graph::narcs

Definition at line 131 of file Cgl012cut.hpp.

#### 6.1.2.3 [cgl\\_node](#)\* auxiliary\_graph::nodes

Definition at line 132 of file Cgl012cut.hpp.

#### 6.1.2.4 [cgl\\_arc](#)\* auxiliary\_graph::arcs

Definition at line 133 of file Cgl012cut.hpp.

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

- src/CglZeroHalf/[Cgl012cut.hpp](#)

## 6.2 Cgl012Cut Class Reference

012Cut Generator Class

```
#include <Cgl012cut.hpp>
```

### Public Member Functions

#### Constructors and destructors

- [Cgl012Cut](#) ()  
*Default constructor.*
- [Cgl012Cut](#) (const [Cgl012Cut](#) &)  
*Copy constructor.*
- [Cgl012Cut](#) & [operator=](#) (const [Cgl012Cut](#) &rhs)  
*Assignment operator.*
- virtual [~Cgl012Cut](#) ()  
*Destructor.*

### Generate Cuts

- int [sep\\_012\\_cut](#) (int mr, int mc, int mnz, int \*mtbeg, int \*mtcnt, int \*mtind, int \*mtval, int \*vlb, int \*vub, int \*mrhs, char \*msense, const double \*xstar, bool aggressive, int \*cnum, int \*cnzcnt, int \*\*cbeg, int \*\*ccnt, int \*\*cind, int \*\*cval, int \*\*crhs, char \*\*csense)

- void [ilp\\_load](#) (int *mr*, int *mc*, int *mnz*, int \**mtbeg*, int \**mtcnt*, int \**mtind*, int \**mtval*, int \**vlb*, int \**vub*, int \**mrhs*, char \**msense*)
- void [free\\_ilp](#) ()
- void [alloc\\_parity\\_ilp](#) (int *mr*, int *mc*, int *mnz*)
- void [free\\_parity\\_ilp](#) ()
- void [initialize\\_log\\_var](#) ()
- void [free\\_log\\_var](#) ()

### 6.2.1 Detailed Description

#### 012Cut Generator Class

This class is to make Cgl01cut thread safe etc

Definition at line 207 of file Cgl012cut.hpp.

### 6.2.2 Constructor & Destructor Documentation

#### 6.2.2.1 Cgl012Cut::Cgl012Cut ( )

Default constructor.

#### 6.2.2.2 Cgl012Cut::Cgl012Cut ( const Cgl012Cut & )

Copy constructor.

#### 6.2.2.3 virtual Cgl012Cut::~~Cgl012Cut ( ) [virtual]

Destructor.

### 6.2.3 Member Function Documentation

#### 6.2.3.1 int Cgl012Cut::sep\_012\_cut ( int *mr*, int *mc*, int *mnz*, int \**mtbeg*, int \**mtcnt*, int \**mtind*, int \**mtval*, int \**vlb*, int \**vub*, int \**mrhs*, char \**msense*, const double \**xstar*, bool *aggressive*, int \**cnum*, int \**cnzcnt*, int \*\**cbeg*, int \*\**ccnt*, int \*\**cind*, int \*\**cval*, int \*\**crhs*, char \*\**csense* )

#### 6.2.3.2 void Cgl012Cut::ilp\_load ( int *mr*, int *mc*, int *mnz*, int \**mtbeg*, int \**mtcnt*, int \**mtind*, int \**mtval*, int \**vlb*, int \**vub*, int \**mrhs*, char \**msense* )

#### 6.2.3.3 void Cgl012Cut::free\_ilp ( )

#### 6.2.3.4 void Cgl012Cut::alloc\_parity\_ilp ( int *mr*, int *mc*, int *mnz* )

#### 6.2.3.5 void Cgl012Cut::free\_parity\_ilp ( )

#### 6.2.3.6 void Cgl012Cut::initialize\_log\_var ( )

#### 6.2.3.7 void Cgl012Cut::free\_log\_var ( )

#### 6.2.3.8 Cgl012Cut& Cgl012Cut::operator= ( const Cgl012Cut & *rhs* )

Assignment operator.

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

- [src/CglZeroHalf/Cgl012cut.hpp](#)

## 6.3 cgl\_arc Struct Reference

```
#include <Cgl012cut.hpp>
```

### Public Attributes

- int [length](#)
- int [to](#)

#### 6.3.1 Detailed Description

Definition at line 35 of file Cgl012cut.hpp.

#### 6.3.2 Member Data Documentation

##### 6.3.2.1 int cgl\_arc::length

Definition at line 37 of file Cgl012cut.hpp.

##### 6.3.2.2 int cgl\_arc::to

Definition at line 38 of file Cgl012cut.hpp.

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

- [src/CglZeroHalf/Cgl012cut.hpp](#)

## 6.4 cgl\_graph Struct Reference

```
#include <Cgl012cut.hpp>
```

### Public Attributes

- int [nnodes](#)
- int [narcs](#)
- [cgl\\_node](#) \* [nodes](#)
- [cgl\\_arc](#) \* [arcs](#)

#### 6.4.1 Detailed Description

Definition at line 49 of file Cgl012cut.hpp.

#### 6.4.2 Member Data Documentation

##### 6.4.2.1 int cgl\_graph::nnodes

Definition at line 51 of file Cgl012cut.hpp.

#### 6.4.2.2 int cgl\_graph::narcs

Definition at line 52 of file Cgl012cut.hpp.

#### 6.4.2.3 cgl\_node\* cgl\_graph::nodes

Definition at line 53 of file Cgl012cut.hpp.

#### 6.4.2.4 cgl\_arc\* cgl\_graph::arcs

Definition at line 54 of file Cgl012cut.hpp.

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

- [src/CglZeroHalf/Cgl012cut.hpp](#)

### 6.5 cgl\_node Struct Reference

```
#include <Cgl012cut.hpp>
```

#### Public Attributes

- [cgl\\_arc \\* firstArc](#)
- [int parentNode](#)
- [int index](#)
- [int distanceBack](#)

#### 6.5.1 Detailed Description

Definition at line 41 of file Cgl012cut.hpp.

#### 6.5.2 Member Data Documentation

##### 6.5.2.1 cgl\_arc\* cgl\_node::firstArc

Definition at line 43 of file Cgl012cut.hpp.

##### 6.5.2.2 int cgl\_node::parentNode

Definition at line 44 of file Cgl012cut.hpp.

##### 6.5.2.3 int cgl\_node::index

Definition at line 45 of file Cgl012cut.hpp.

##### 6.5.2.4 int cgl\_node::distanceBack

Definition at line 46 of file Cgl012cut.hpp.

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

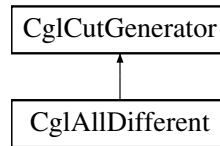
- [src/CglZeroHalf/Cgl012cut.hpp](#)

## 6.6 CglAllDifferent Class Reference

AllDifferent Cut Generator Class This has a number of sets.

```
#include <CglAllDifferent.hpp>
```

Inheritance diagram for CglAllDifferent:



### Public Member Functions

#### Generate Cuts

- virtual void [generateCuts](#) (const OsiSolverInterface &si, OsiCuts &cs, const [CglTreeInfo](#) info=[CglTreeInfo](#)())  
*This fixes (or reduces bounds) on sets of all different variables.*

#### Constructors and destructors

- [CglAllDifferent](#) ()  
*Default constructor.*
- [CglAllDifferent](#) (int numberSets, const int \*starts, const int \*which)  
*Useful constructot.*
- [CglAllDifferent](#) (const [CglAllDifferent](#) &)  
*Copy constructor.*
- virtual [CglCutGenerator](#) \* [clone](#) () const  
*Clone.*
- [CglAllDifferent](#) & [operator=](#) (const [CglAllDifferent](#) &rhs)  
*Assignment operator.*
- virtual [~CglAllDifferent](#) ()  
*Destructor.*
- virtual std::string [generateCpp](#) (FILE \*fp)  
*Create C++ lines to get to current state.*
- virtual void [refreshSolver](#) (OsiSolverInterface \*solver)  
*This can be used to refresh any inforamtion.*
- virtual bool [mayGenerateRowCutsInTree](#) () const  
*Returns true if may generate Row cuts in tree (rather than root node).*

#### Sets and Gets

- void [setLogLevel](#) (int value)  
*Set log level.*
- int [getLogLevel](#) () const  
*Get log level.*
- void [setMaxLook](#) (int value)  
*Set Maximum number of sets to look at at once.*
- int [getMaxLook](#) () const  
*Get Maximum number of sets to look at at once.*

## Additional Inherited Members

### 6.6.1 Detailed Description

AllDifferent Cut Generator Class This has a number of sets.

All the members in each set are general integer variables which have to be different from all others in the set.

At present this only generates column cuts

At present it is very primitive compared to proper CSP implementations

Definition at line 20 of file CglAllDifferent.hpp.

### 6.6.2 Constructor & Destructor Documentation

#### 6.6.2.1 CglAllDifferent::CglAllDifferent ( )

Default constructor.

#### 6.6.2.2 CglAllDifferent::CglAllDifferent ( int *numberSets*, const int \* *starts*, const int \* *which* )

Useful constructot.

#### 6.6.2.3 CglAllDifferent::CglAllDifferent ( const CglAllDifferent & )

Copy constructor.

#### 6.6.2.4 virtual CglAllDifferent::~~CglAllDifferent ( ) [virtual]

Destructor.

### 6.6.3 Member Function Documentation

#### 6.6.3.1 virtual void CglAllDifferent::generateCuts ( const OsiSolverInterface & *si*, OsiCuts & *cs*, const CglTreeInfo *info* = CglTreeInfo() ) [virtual]

This fixes (or reduces bounds) on sets of all different variables.

Implements [CglCutGenerator](#).

#### 6.6.3.2 virtual CglCutGenerator\* CglAllDifferent::clone ( ) const [virtual]

Clone.

Implements [CglCutGenerator](#).

#### 6.6.3.3 CglAllDifferent& CglAllDifferent::operator= ( const CglAllDifferent & *rhs* )

Assignment operator.

#### 6.6.3.4 virtual std::string CglAllDifferent::generateCpp ( FILE \* *fp* ) [virtual]

Create C++ lines to get to current state.

Reimplemented from [CglCutGenerator](#).

6.6.3.5 `virtual void CglAllDifferent::refreshSolver ( OsiSolverInterface * solver ) [virtual]`

This can be used to refresh any inforamtion.

Reimplemented from [CglCutGenerator](#).

6.6.3.6 `virtual bool CglAllDifferent::mayGenerateRowCutsInTree ( ) const [inline],[virtual]`

Returns true if may generate Row cuts in tree (rather than root node).

Used so know if matrix will change in tree. Really meant so column cut generators can still be active without worrying code. Default is true

Reimplemented from [CglCutGenerator](#).

Definition at line 69 of file `CglAllDifferent.hpp`.

6.6.3.7 `void CglAllDifferent::setLogLevel ( int value ) [inline]`

Set log level.

Definition at line 75 of file `CglAllDifferent.hpp`.

6.6.3.8 `int CglAllDifferent::getLogLevel ( ) const [inline]`

Get log level.

Definition at line 78 of file `CglAllDifferent.hpp`.

6.6.3.9 `void CglAllDifferent::setMaxLook ( int value ) [inline]`

Set Maximum number of sets to look at at once.

Definition at line 81 of file `CglAllDifferent.hpp`.

6.6.3.10 `int CglAllDifferent::getMaxLook ( ) const [inline]`

Get Maximum number of sets to look at at once.

Definition at line 84 of file `CglAllDifferent.hpp`.

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

- `src/CglAllDifferent/CglAllDifferent.hpp`

## 6.7 CglBK Class Reference

For Bron-Kerbosch.

```
#include <CglPreProcess.hpp>
```

### Public Member Functions

#### Main methods

- `void bronKerbosch ()`  
*For recursive Bron-Kerbosch.*
- `OsiSolverInterface * newSolver (const OsiSolverInterface &model)`  
*Creates strengthened smaller model.*



## Constructors and destructors etc

- [CglBK](#) ()  
*Default constructor.*
- [CglBK](#) (const OsiSolverInterface &model, const char \*rowType, int numberElements)  
*Useful constructor.*
- [CglBK](#) (const [CglBK](#) &rhs)  
*Copy constructor.*
- [CglBK](#) & [operator=](#) (const [CglBK](#) &rhs)  
*Assignment operator.*
- [~CglBK](#) ()  
*Destructor.*

### 6.7.1 Detailed Description

For Bron-Kerbosch.

Definition at line 364 of file CglPreProcess.hpp.

### 6.7.2 Constructor & Destructor Documentation

#### 6.7.2.1 [CglBK::CglBK](#) ( )

Default constructor.

#### 6.7.2.2 [CglBK::CglBK](#) ( const OsiSolverInterface & *model*, const char \* *rowType*, int *numberElements* )

Useful constructor.

#### 6.7.2.3 [CglBK::CglBK](#) ( const [CglBK](#) & *rhs* )

Copy constructor .

#### 6.7.2.4 [CglBK::~~CglBK](#) ( )

Destructor.

### 6.7.3 Member Function Documentation

#### 6.7.3.1 void [CglBK::bronKerbosch](#) ( )

For recursive Bron-Kerbosch.

#### 6.7.3.2 OsiSolverInterface\* [CglBK::newSolver](#) ( const OsiSolverInterface & *model* )

Creates strengthened smaller model.

#### 6.7.3.3 [CglBK& CglBK::operator=](#) ( const [CglBK](#) & *rhs* )

Assignment operator.

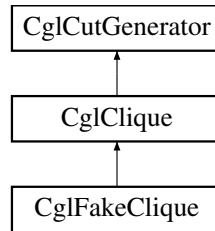
The documentation for this class was generated from the following file:

- [src/CglPreProcess/CglPreProcess.hpp](#)

## 6.8 CglClique Class Reference

```
#include <CglClique.hpp>
```

Inheritance diagram for CglClique:



### Public Member Functions

- [CglClique](#) (const [CglClique](#) &rhs)  
*Copy constructor.*
- virtual [CglCutGenerator](#) \* [clone](#) () const  
*Clone.*
- [CglClique](#) & [operator=](#) (const [CglClique](#) &rhs)  
*Assignment operator.*
- virtual void [generateCuts](#) (const OsiSolverInterface &si, OsiCuts &cs, const [CglTreeInfo](#) info=[CglTreeInfo](#)())  
*Generate cuts for the model data contained in si.*

### Protected Attributes

- const int \* [cl\\_perm\\_indices](#)  
*variables/arrays that are used across many methods*
- int [cl\\_perm\\_length](#)  
*The length of cl\_perm\_indices.*
- int \* [cl\\_indices](#)  
*List of indices that should be considered for extending the ones listed in cl\_perm\_indices.*
- int [cl\\_length](#)  
*The length of cl\_indices.*
- int \* [cl\\_del\\_indices](#)  
*An array of nodes discarded from the candidate list.*
- int [cl\\_del\\_length](#)  
*The length of cl\_del\_indices.*

### Friends

- void [CglCliqueUnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)  
*A function that tests the methods in the [CglClique](#) class.*

## Constructors and destructors

- enum `scl_next_node_method` { `SCL_MIN_DEGREE`, `SCL_MAX_DEGREE`, `SCL_MAX_XJ_MAX_DEG` }  
*possible choices for selecting the next node in the star clique search*
- struct `frac_graph`
- bool `setPacking_`  
*An indicator showing whether the whole matrix in the solverinterface is a set packing problem or not.*
- bool `justOriginalRows_`  
*True if just look at original rows.*
- int `sp_numrows`  
*pieces of the set packing part of the solverinterface*
- int \* `sp_orig_row_ind`
- int `sp_numcols`
- int \* `sp_orig_col_ind`
- double \* `sp_colsol`
- int \* `sp_col_start`
- int \* `sp_col_ind`
- int \* `sp_row_start`
- int \* `sp_row_ind`
- `frac_graph` `fgraph`  
*the intersection graph corresponding to the set packing problem*
- bool \* `node_node`  
*the node-node incidence matrix of the intersection graph.*
- double `petol`  
*The primal tolerance in the solverinterface.*
- bool `do_row_clique`  
*data for the star clique algorithm*
- bool `do_star_clique`  
*whether to do the star clique algorithm or not.*
- `scl_next_node_method` `scl_next_node_rule`  
*How the next node to be added to the star clique should be selected.*
- int `scl_candidate_length_threshold`  
*In the star clique method the maximal length of the candidate list (those nodes that are in a star, i.e., connected to the center of the star) to allow complete enumeration of maximal cliques.*
- bool `scl_report_result`  
*whether to give a detailed statistics on the star clique method*
- int `rcl_candidate_length_threshold`  
*In the row clique method the maximal length of the candidate list (those nodes that can extend the row clique, i.e., connected to all nodes in the row clique) to allow complete enumeration of maximal cliques.*
- bool `rcl_report_result`  
*whether to give a detailed statistics on the row clique method*
- `CglClique` (bool `setPacking=false`, bool `justOriginalRows=false`)  
*Default constructor.*
- virtual `~CglClique` ()  
*Destructor.*
- virtual std::string `generateCpp` (FILE \*fp)  
*Create C++ lines to get to current state.*
- void `considerRows` (const int numRows, const int \*rowInd)

- void [setStarCliqueNextNodeMethod](#) ([scl\\_next\\_node\\_method](#) method)
- void [setStarCliqueCandidateLengthThreshold](#) (int maxlen)
- void [setRowCliqueCandidateLengthThreshold](#) (int maxlen)
- void [setStarCliqueReport](#) (bool yesno=true)
- void [setRowCliqueReport](#) (bool yesno=true)
- void [setDoStarClique](#) (bool yesno=true)
- void [setDoRowClique](#) (bool yesno=true)
- void [setMinViolation](#) (double minviol)
- double [getMinViolation](#) () const

#### Additional Inherited Members

##### 6.8.1 Detailed Description

Definition at line 14 of file CglClique.hpp.

##### 6.8.2 Member Enumeration Documentation

###### 6.8.2.1 enum CglClique::scl\_next\_node\_method

possible choices for selecting the next node in the star clique search

Enumerator

***SCL\_MIN\_DEGREE***  
***SCL\_MAX\_DEGREE***  
***SCL\_MAX\_XJ\_MAX\_DEG***

Definition at line 64 of file CglClique.hpp.

##### 6.8.3 Constructor & Destructor Documentation

###### 6.8.3.1 CglClique::CglClique ( const CglClique & rhs )

Copy constructor.

###### 6.8.3.2 CglClique::CglClique ( bool setPacking = false, bool justOriginalRows = false )

Default constructor.

If the setPacking argument is set to true then [CglClique](#) will assume that the problem in the solverinterface passed to the [generateCuts\(\)](#) method describes a set packing problem, i.e.,

- all variables are binary
- the matrix is a 0-1 matrix
- all constraints are ' $= 1$ ' or ' $\leq 1$ '

Otherwise the user can use the [considerRows\(\)](#) method to set the list of clique rows, that is,

- all coeffs corresponding to binary variables at fractional level is 1

- all other coeffs are non-negative
- the constraint is ' $= 1$ ' or ' $\leq 1$ '.

If the user does not set the list of clique rows then [CglClique](#) will start the [generateCuts\(\)](#) methods by scanning the matrix for them. Also `justOriginalRows` can be set to true to limit clique creation

**6.8.3.3** `virtual CglClique::~CglClique ( ) [inline], [virtual]`

Destructor.

Definition at line 55 of file `CglClique.hpp`.

#### 6.8.4 Member Function Documentation

**6.8.4.1** `virtual CglCutGenerator* CglClique::clone ( ) const [virtual]`

Clone.

Implements [CglCutGenerator](#).

Reimplemented in [CglFakeClique](#).

**6.8.4.2** `CglClique& CglClique::operator= ( const CglClique & rhs )`

Assignment operator.

**6.8.4.3** `virtual void CglClique::generateCuts ( const OsiSolverInterface & si, OsiCuts & cs, const CglTreeInfo info = CglTreeInfo() ) [virtual]`

Generate cuts for the model data contained in `si`.

The generated cuts are inserted into and returned in the collection of cuts `cs`.

Implements [CglCutGenerator](#).

Reimplemented in [CglFakeClique](#).

**6.8.4.4** `virtual std::string CglClique::generateCpp ( FILE * fp ) [virtual]`

Create C++ lines to get to current state.

Reimplemented from [CglCutGenerator](#).

**6.8.4.5** `void CglClique::considerRows ( const int numRows, const int * rowInd )`

**6.8.4.6** `void CglClique::setStarCliqueNextNodeMethod ( scl_next_node_method method ) [inline]`

Definition at line 70 of file `CglClique.hpp`.

**6.8.4.7** `void CglClique::setStarCliqueCandidateLengthThreshold ( int maxlen ) [inline]`

Definition at line 74 of file `CglClique.hpp`.

**6.8.4.8** `void CglClique::setRowCliqueCandidateLengthThreshold ( int maxlen ) [inline]`

Definition at line 77 of file `CglClique.hpp`.

6.8.4.9 `void CglClique::setStarCliqueReport ( bool yesno =true ) [inline]`

Definition at line 81 of file CglClique.hpp.

6.8.4.10 `void CglClique::setRowCliqueReport ( bool yesno =true ) [inline]`

Definition at line 82 of file CglClique.hpp.

6.8.4.11 `void CglClique::setDoStarClique ( bool yesno =true ) [inline]`

Definition at line 84 of file CglClique.hpp.

6.8.4.12 `void CglClique::setDoRowClique ( bool yesno =true ) [inline]`

Definition at line 85 of file CglClique.hpp.

6.8.4.13 `void CglClique::setMinViolation ( double minviol ) [inline]`

Definition at line 87 of file CglClique.hpp.

6.8.4.14 `double CglClique::getMinViolation ( ) const [inline]`

Definition at line 88 of file CglClique.hpp.

## 6.8.5 Friends And Related Function Documentation

6.8.5.1 `friend struct frac_graph [friend]`

Definition at line 92 of file CglClique.hpp.

6.8.5.2 `void CglCliqueUnitTest ( const OsiSolverInterface * siP, const std::string mpdDir ) [friend]`

A function that tests the methods in the [CglClique](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

## 6.8.6 Member Data Documentation

6.8.6.1 `bool CglClique::setPacking_ [protected]`

An indicator showing whether the whole matrix in the solverinterface is a set packing problem or not.

Definition at line 139 of file CglClique.hpp.

6.8.6.2 `bool CglClique::justOriginalRows_ [protected]`

True if just look at original rows.

Definition at line 141 of file CglClique.hpp.

6.8.6.3 `int CglClique::sp_numrows [protected]`

pieces of the set packing part of the solverinterface

Definition at line 143 of file CglClique.hpp.

**6.8.6.4** `int* CgIClique::sp_orig_row_ind` [protected]

Definition at line 144 of file CgIClique.hpp.

**6.8.6.5** `int CgIClique::sp_numcols` [protected]

Definition at line 145 of file CgIClique.hpp.

**6.8.6.6** `int* CgIClique::sp_orig_col_ind` [protected]

Definition at line 146 of file CgIClique.hpp.

**6.8.6.7** `double* CgIClique::sp_colsol` [protected]

Definition at line 147 of file CgIClique.hpp.

**6.8.6.8** `int* CgIClique::sp_col_start` [protected]

Definition at line 148 of file CgIClique.hpp.

**6.8.6.9** `int* CgIClique::sp_col_ind` [protected]

Definition at line 149 of file CgIClique.hpp.

**6.8.6.10** `int* CgIClique::sp_row_start` [protected]

Definition at line 150 of file CgIClique.hpp.

**6.8.6.11** `int* CgIClique::sp_row_ind` [protected]

Definition at line 151 of file CgIClique.hpp.

**6.8.6.12** `frac_graph CgIClique::fgraph` [protected]

the intersection graph corresponding to the set packing problem

Definition at line 154 of file CgIClique.hpp.

**6.8.6.13** `bool* CgIClique::node_node` [protected]

the node-node incidence matrix of the intersection graph.

Definition at line 156 of file CgIClique.hpp.

**6.8.6.14** `double CgIClique::petol` [protected]

The primal tolerance in the solverinterface.

Definition at line 159 of file CgIClique.hpp.

**6.8.6.15** `bool CgIClique::do_row_clique` [protected]

data for the star clique algorithm

Parameters whether to do the row clique algorithm or not.

Definition at line 166 of file CgIClique.hpp.

**6.8.6.16** `bool CgIClique::do_star_clique` `[protected]`

whether to do the star clique algorithm or not.

Definition at line 168 of file CgIClique.hpp.

**6.8.6.17** `scl_next_node_method CgIClique::scl_next_node_rule` `[protected]`

How the next node to be added to the star clique should be selected.

Definition at line 171 of file CgIClique.hpp.

**6.8.6.18** `int CgIClique::scl_candidate_length_threshold` `[protected]`

In the star clique method the maximal length of the candidate list (those nodes that are in a star, i.e., connected to the center of the star) to allow complete enumeration of maximal cliques.

Otherwise a greedy algorithm is used.

Definition at line 176 of file CgIClique.hpp.

**6.8.6.19** `bool CgIClique::scl_report_result` `[protected]`

whether to give a detailed statistics on the star clique method

Definition at line 178 of file CgIClique.hpp.

**6.8.6.20** `int CgIClique::rcl_candidate_length_threshold` `[protected]`

In the row clique method the maximal length of the candidate list (those nodes that can extend the row clique, i.e., connected to all nodes in the row clique) to allow complete enumeration of maximal cliques.

Otherwise a greedy algorithm is used.

Definition at line 184 of file CgIClique.hpp.

**6.8.6.21** `bool CgIClique::rcl_report_result` `[protected]`

whether to give a detailed statistics on the row clique method

Definition at line 186 of file CgIClique.hpp.

**6.8.6.22** `const int* CgIClique::cl_perm_indices` `[protected]`

variables/arrays that are used across many methods

List of indices that must be in the to be created clique. This is just a pointer, it is never new'd and therefore does not need to be delete'd either.

Definition at line 194 of file CgIClique.hpp.

**6.8.6.23** `int CgIClique::cl_perm_length` `[protected]`

The length of cl\_perm\_indices.

Definition at line 196 of file CgIClique.hpp.

**6.8.6.24** `int* CgIClique::cl_indices` `[protected]`

List of indices that should be considered for extending the ones listed in cl\_perm\_indices.

Definition at line 200 of file CgIClique.hpp.



**6.8.6.25** `int CglClique::cl_length` `[protected]`

The length of `cl_indices`.

Definition at line 202 of file `CglClique.hpp`.

**6.8.6.26** `int* CglClique::cl_del_indices` `[protected]`

An array of nodes discarded from the candidate list.

These are rechecked when a maximal clique is found just to make sure that the clique is really maximal.

Definition at line 207 of file `CglClique.hpp`.

**6.8.6.27** `int CglClique::cl_del_length` `[protected]`

The length of `cl_del_indices`.

Definition at line 209 of file `CglClique.hpp`.

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

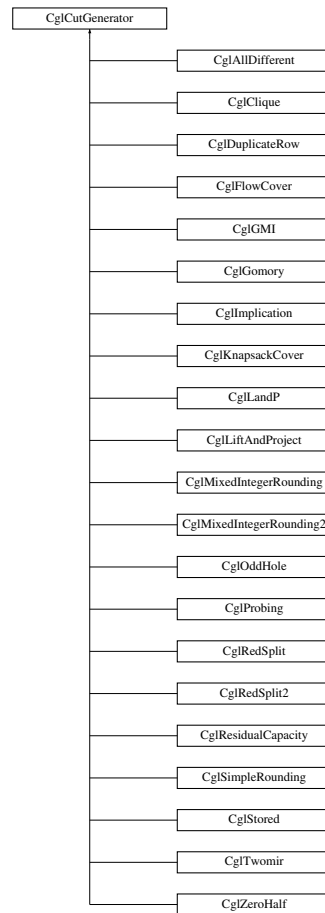
- `src/CglClique/CglClique.hpp`

## 6.9 CglCutGenerator Class Reference

Cut Generator Base Class.

```
#include <CglCutGenerator.hpp>
```

Inheritance diagram for `CglCutGenerator`:



## Public Member Functions

### Generate Cuts

- virtual void [generateCuts](#) (const OsiSolverInterface &si, OsiCuts &cs, const [CglTreeInfo](#) info=[CglTreeInfo](#)())=0  
*Generate cuts for the model data contained in si.*

### Constructors and destructors

- [CglCutGenerator](#) ()  
*Default constructor.*
- [CglCutGenerator](#) (const [CglCutGenerator](#) &)  
*Copy constructor.*
- virtual [CglCutGenerator](#) \* [clone](#) () const =0  
*Clone.*
- [CglCutGenerator](#) & [operator=](#) (const [CglCutGenerator](#) &rhs)  
*Assignment operator.*
- virtual [~CglCutGenerator](#) ()  
*Destructor.*
- virtual std::string [generateCpp](#) (FILE \*)  
*Create C++ lines to set the generator in the current state.*
- virtual void [refreshSolver](#) (OsiSolverInterface \*)  
*This can be used to refresh any information.*

## Gets and Sets

- int `getAggressiveness` () const  
*Get Aggressiveness - 0 = neutral, 100 is normal root node.*
- void `setAggressiveness` (int value)  
*Set Aggressiveness - 0 = neutral, 100 is normal root node.*
- void `setGlobalCuts` (bool trueOrFalse)  
*Set whether can do global cuts.*
- bool `canDoGlobalCuts` () const  
*Say whether can do global cuts.*
- virtual bool `mayGenerateRowCutsInTree` () const  
*Returns true if may generate Row cuts in tree (rather than root node).*
- virtual bool `needsOptimalBasis` () const  
*Return true if needs optimal basis to do cuts.*
- virtual int `maxLengthOfCutInTree` () const  
*Return maximum length of cut in tree.*

## Public Attributes

- int `aggressive_`  
*Aggressiveness - 0 = neutral, 100 is normal root node.*
- bool `canDoGlobalCuts_`  
*True if can do global cuts i.e. no general integers.*

### 6.9.1 Detailed Description

Cut Generator Base Class.

This is an abstract base class for generating cuts. A specific cut generator will inherit from this class.

Definition at line 23 of file `CglCutGenerator.hpp`.

### 6.9.2 Constructor & Destructor Documentation

#### 6.9.2.1 `CglCutGenerator::CglCutGenerator ( )`

Default constructor.

#### 6.9.2.2 `CglCutGenerator::CglCutGenerator ( const CglCutGenerator & )`

Copy constructor.

#### 6.9.2.3 `virtual CglCutGenerator::~~CglCutGenerator ( ) [virtual]`

Destructor.

### 6.9.3 Member Function Documentation

#### 6.9.3.1 `virtual void CglCutGenerator::generateCuts ( const OsiSolverInterface & si, OsiCuts & cs, const CglTreeInfo info = CglTreeInfo() ) [pure virtual]`

Generate cuts for the model data contained in si.

The generated cuts are inserted into and returned in the collection of cuts `cs`.

Implemented in [CglImplication](#), [CglFakeClique](#), [CglLandP](#), [CglFlowCover](#), [CglMixedIntegerRounding2](#), [CglMixedIntegerRounding](#), [CglTwomir](#), [CglProbing](#), [CglResidualCapacity](#), [CglRedSplit2](#), [CglGMI](#), [CglRedSplit](#), [CglOddHole](#), [CglDuplicateRow](#), [CglSimpleRounding](#), [CglZeroHalf](#), [CglGomory](#), [CglStored](#), [CglClique](#), [CglAllDifferent](#), [CglKnapsackCover](#), and [CglLiftAndProject](#).

**6.9.3.2** `virtual CglCutGenerator* CglCutGenerator::clone ( ) const` `[pure virtual]`

Clone.

Implemented in [CglImplication](#), [CglFakeClique](#), [CglProbing](#), [CglFlowCover](#), [CglLandP](#), [CglRedSplit](#), [CglTwomir](#), [CglRedSplit2](#), [CglGMI](#), [CglMixedIntegerRounding2](#), [CglMixedIntegerRounding](#), [CglGomory](#), [CglDuplicateRow](#), [CglResidualCapacity](#), [CglOddHole](#), [CglStored](#), [CglZeroHalf](#), [CglLiftAndProject](#), [CglSimpleRounding](#), [CglAllDifferent](#), [CglKnapsackCover](#), and [CglClique](#).

**6.9.3.3** `CglCutGenerator& CglCutGenerator::operator= ( const CglCutGenerator & rhs )`

Assignment operator.

**6.9.3.4** `virtual std::string CglCutGenerator::generateCpp ( FILE * )` `[inline],[virtual]`

Create C++ lines to set the generator in the current state.

The output must be parsed by the calling code, as each line starts with a key indicating the following:

0: must be kept (for #includes etc)

3: Set to changed (not default) values

4: Set to default values (redundant)

Keys 1, 2, 5, 6, 7, 8 are defined, but not applicable to cut generators.

Reimplemented in [CglImplication](#), [CglProbing](#), [CglFlowCover](#), [CglRedSplit](#), [CglTwomir](#), [CglMixedIntegerRounding2](#), [CglMixedIntegerRounding](#), [CglGMI](#), [CglGomory](#), [CglDuplicateRow](#), [CglZeroHalf](#), [CglLiftAndProject](#), [CglSimpleRounding](#), [CglAllDifferent](#), [CglClique](#), and [CglKnapsackCover](#).

Definition at line 65 of file `CglCutGenerator.hpp`.

**6.9.3.5** `virtual void CglCutGenerator::refreshSolver ( OsiSolverInterface * )` `[inline],[virtual]`

This can be used to refresh any information.

Reimplemented in [CglProbing](#), [CglTwomir](#), [CglMixedIntegerRounding2](#), [CglMixedIntegerRounding](#), [CglGomory](#), [CglDuplicateRow](#), [CglOddHole](#), [CglZeroHalf](#), [CglAllDifferent](#), and [CglKnapsackCover](#).

Definition at line 68 of file `CglCutGenerator.hpp`.

**6.9.3.6** `int CglCutGenerator::getAggressiveness ( ) const` `[inline]`

Get Aggressiveness - 0 = neutral, 100 is normal root node.

Really just a hint to cut generator

Definition at line 77 of file `CglCutGenerator.hpp`.

**6.9.3.7** `void CglCutGenerator::setAggressiveness ( int value )` `[inline]`

Set Aggressiveness - 0 = neutral, 100 is normal root node.

Really just a hint to cut generator

Definition at line 84 of file `CglCutGenerator.hpp`.

**6.9.3.8** `void CglCutGenerator::setGlobalCuts ( bool trueOrFalse ) [inline]`

Set whether can do global cuts.

Definition at line 87 of file CglCutGenerator.hpp.

**6.9.3.9** `bool CglCutGenerator::canDoGlobalCuts ( ) const [inline]`

Say whether can do global cuts.

Definition at line 90 of file CglCutGenerator.hpp.

**6.9.3.10** `virtual bool CglCutGenerator::mayGenerateRowCutsInTree ( ) const [virtual]`

Returns true if may generate Row cuts in tree (rather than root node).

Used so know if matrix will change in tree. Really meant so column cut generators can still be active without worrying code. Default is true

Reimplemented in [CglProbing](#), and [CglAllDifferent](#).

**6.9.3.11** `virtual bool CglCutGenerator::needsOptimalBasis ( ) const [virtual]`

Return true if needs optimal basis to do cuts.

Reimplemented in [CglLandP](#), [CglTwomir](#), [CglRedSplit2](#), [CglGMI](#), [CglRedSplit](#), and [CglGomory](#).

**6.9.3.12** `virtual int CglCutGenerator::maxLengthOfCutInTree ( ) const [inline],[virtual]`

Return maximum length of cut in tree.

Reimplemented in [CglTwomir](#), and [CglGomory](#).

Definition at line 103 of file CglCutGenerator.hpp.

## 6.9.4 Member Data Documentation

**6.9.4.1** `int CglCutGenerator::aggressive_`

Aggressiveness - 0 = neutral, 100 is normal root node.

Really just a hint to cut generator

Definition at line 116 of file CglCutGenerator.hpp.

**6.9.4.2** `bool CglCutGenerator::canDoGlobalCuts_`

True if can do global cuts i.e. no general integers.

Definition at line 118 of file CglCutGenerator.hpp.

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

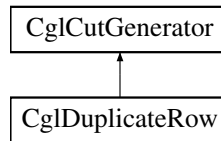
- [src/CglCutGenerator.hpp](#)

## 6.10 CglDuplicateRow Class Reference

DuplicateRow Cut Generator Class.

```
#include <CglDuplicateRow.hpp>
```

Inheritance diagram for CglDuplicateRow:



## Public Member Functions

### Get information on size of problem

- `const int * duplicate () const`  
*Get duplicate row list, -1 means still in, -2 means out (all fixed),  $k \geq$  means same as row  $k$ .*
- `int sizeDynamic () const`  
*Size of dynamic program.*
- `int numberOriginalRows () const`  
*Number of rows in original problem.*
- `int logLevel () const`  
*logLevel*
- `void setLogLevel (int value)`

### We only check for duplicates amongst rows with effective rhs $\leq$ this

- `int maximumRhs () const`  
*Get.*
- `void setMaximumRhs (int value)`  
*Set.*

### We only check for dominated amongst groups of columns whose size $\leq$ this

- `int maximumDominated () const`  
*Get.*
- `void setMaximumDominated (int value)`  
*Set.*

### gets and sets

- `int mode () const`  
*Get mode.*
- `void setMode (int value)`  
*Set mode.*

### Constructors and destructors

- `CglDuplicateRow ()`  
*Default constructor.*
- `CglDuplicateRow (OsiSolverInterface *solver)`  
*Useful constructor.*
- `CglDuplicateRow (const CglDuplicateRow &rhs)`  
*Copy constructor.*
- `virtual CglCutGenerator * clone () const`  
*Clone.*

- `CglDuplicateRow` & `operator=` (const `CglDuplicateRow` &rhs)  
*Assignment operator.*
- virtual `~CglDuplicateRow` ()  
*Destructor.*
- virtual `std::string generateCpp` (FILE \*fp)  
*Create C++ lines to get to current state.*
- virtual void `refreshSolver` (OsiSolverInterface \*solver)  
*This can be used to refresh any information.*

## Protected Attributes

### Protected member data

- CoinPackedMatrix `matrix_`  
*Matrix.*
- CoinPackedMatrix `matrixByRow_`  
*Matrix by row.*
- int \* `rhs_`  
*Possible rhs (if 0 then not possible)*
- int \* `duplicate_`  
*Marks duplicate rows.*
- int \* `lower_`  
*To allow for  $\leq$  rows.*
- `CglStored` \* `storedCuts_`  
*Stored cuts if we found dominance cuts.*
- int `maximumDominated_`  
*Check dominated columns if less than this number of candidates.*
- int `maximumRhs_`  
*Check duplicates if effective rhs  $\leq$  this.*
- int `sizeDynamic_`  
*Size of dynamic program.*
- int `mode_`  
*1 do rows, 2 do columns, 3 do both*
- int `logLevel_`  
*Controls print out.*

## Generate Cuts

- virtual void `generateCuts` (const OsiSolverInterface &si, OsiCuts &cs, const `CglTreeInfo` info=`CglTreeInfo`())  
*Fix variables and find duplicate/dominated rows for the model of the solver interface, si.*
- `CglStored` \* `outDuplicates` (OsiSolverInterface \*solver)  
*Fix variables and find duplicate/dominated rows for the model of the solver interface, si.*

## Additional Inherited Members

### 6.10.1 Detailed Description

DuplicateRow Cut Generator Class.

Definition at line 15 of file `CglDuplicateRow.hpp`.

### 6.10.2 Constructor & Destructor Documentation

#### 6.10.2.1 CglDuplicateRow::CglDuplicateRow ( )

Default constructor.

#### 6.10.2.2 CglDuplicateRow::CglDuplicateRow ( OsiSolverInterface \* *solver* )

Useful constructor.

#### 6.10.2.3 CglDuplicateRow::CglDuplicateRow ( const CglDuplicateRow & *rhs* )

Copy constructor.

#### 6.10.2.4 virtual CglDuplicateRow::~CglDuplicateRow ( ) [virtual]

Destructor.

### 6.10.3 Member Function Documentation

#### 6.10.3.1 virtual void CglDuplicateRow::generateCuts ( const OsiSolverInterface & *si*, OsiCuts & *cs*, const CglTreeInfo *info* = CglTreeInfo ( ) ) [virtual]

Fix variables and find duplicate/dominated rows for the model of the solver interface, *si*.

This is a very simple minded idea but I (JJF) am using it in a project so thought I might as well add it. It should really be called before first solve and I may modify CBC to allow for that.

This is designed for problems with few rows and many integer variables where the rhs are  $\leq$  or  $=$  and all coefficients and rhs are small integers.

If effective rhs is *K* then we can fix all variables with coefficients  $> K$  to their lower bounds (effective rhs just means original with variables with nonzero lower bounds subtracted out).

If one row is a subset of another and the effective rhs are same we can fix some variables and then the two rows are identical.

The generator marks identical rows so can be taken out in solve

Implements [CglCutGenerator](#).

#### 6.10.3.2 CglStored\* CglDuplicateRow::outDuplicates ( OsiSolverInterface \* *solver* )

Fix variables and find duplicate/dominated rows for the model of the solver interface, *si*.

This is a very simple minded idea but I (JJF) am using it in a project so thought I might as well add it. It should really be called before first solve and I may modify CBC to allow for that.

This is designed for problems with few rows and many integer variables where the rhs are  $\leq$  or  $=$  and all coefficients and rhs are small integers.

If effective rhs is *K* then we can fix all variables with coefficients  $> K$  to their lower bounds (effective rhs just means original with variables with nonzero lower bounds subtracted out).

If one row is a subset of another and the effective rhs are same we can fix some variables and then the two rows are identical.

This version does deletions and fixings and may return stored cuts for dominated columns



### 6.10.3.3 `const int* CglDuplicateRow::duplicate ( ) const [inline]`

Get duplicate row list, -1 means still in, -2 means out (all fixed),  $k \geq$  means same as row  $k$ .

Definition at line 79 of file CglDuplicateRow.hpp.

### 6.10.3.4 `int CglDuplicateRow::sizeDynamic ( ) const [inline]`

Size of dynamic program.

Definition at line 82 of file CglDuplicateRow.hpp.

### 6.10.3.5 `int CglDuplicateRow::numberOriginalRows ( ) const [inline]`

Number of rows in original problem.

Definition at line 85 of file CglDuplicateRow.hpp.

### 6.10.3.6 `int CglDuplicateRow::logLevel ( ) const [inline]`

logLevel

Definition at line 92 of file CglDuplicateRow.hpp.

### 6.10.3.7 `void CglDuplicateRow::setLogLevel ( int value ) [inline]`

Definition at line 94 of file CglDuplicateRow.hpp.

### 6.10.3.8 `int CglDuplicateRow::maximumRhs ( ) const [inline]`

Get.

Definition at line 102 of file CglDuplicateRow.hpp.

### 6.10.3.9 `void CglDuplicateRow::setMaximumRhs ( int value ) [inline]`

Set.

Definition at line 105 of file CglDuplicateRow.hpp.

### 6.10.3.10 `int CglDuplicateRow::maximumDominated ( ) const [inline]`

Get.

Definition at line 112 of file CglDuplicateRow.hpp.

### 6.10.3.11 `void CglDuplicateRow::setMaximumDominated ( int value ) [inline]`

Set.

Definition at line 115 of file CglDuplicateRow.hpp.

### 6.10.3.12 `int CglDuplicateRow::mode ( ) const [inline]`

Get mode.

Definition at line 121 of file CglDuplicateRow.hpp.

### 6.10.3.13 `void CglDuplicateRow::setMode ( int value ) [inline]`

Set mode.

Definition at line 124 of file CglDuplicateRow.hpp.

6.10.3.14 `virtual CglCutGenerator* CglDuplicateRow::clone ( ) const` [virtual]

Clone.

Implements [CglCutGenerator](#).

6.10.3.15 `CglDuplicateRow& CglDuplicateRow::operator= ( const CglDuplicateRow & rhs )`

Assignment operator.

6.10.3.16 `virtual std::string CglDuplicateRow::generateCpp ( FILE * fp )` [virtual]

Create C++ lines to get to current state.

Reimplemented from [CglCutGenerator](#).

6.10.3.17 `virtual void CglDuplicateRow::refreshSolver ( OsiSolverInterface * solver )` [virtual]

This can be used to refresh any information.

Reimplemented from [CglCutGenerator](#).

#### 6.10.4 Member Data Documentation

6.10.4.1 `CoinPackedMatrix CglDuplicateRow::matrix_` [protected]

Matrix.

Definition at line 166 of file CglDuplicateRow.hpp.

6.10.4.2 `CoinPackedMatrix CglDuplicateRow::matrixByRow_` [protected]

Matrix by row.

Definition at line 168 of file CglDuplicateRow.hpp.

6.10.4.3 `int* CglDuplicateRow::rhs_` [protected]

Possible rhs (if 0 then not possible)

Definition at line 170 of file CglDuplicateRow.hpp.

6.10.4.4 `int* CglDuplicateRow::duplicate_` [protected]

Marks duplicate rows.

Definition at line 172 of file CglDuplicateRow.hpp.

6.10.4.5 `int* CglDuplicateRow::lower_` [protected]

To allow for <= rows.

Definition at line 174 of file CglDuplicateRow.hpp.

6.10.4.6 `CglStored* CglDuplicateRow::storedCuts_` [protected]

Stored cuts if we found dominance cuts.

Definition at line 176 of file CglDuplicateRow.hpp.

#### 6.10.4.7 `int CglDuplicateRow::maximumDominated_ [protected]`

Check dominated columns if less than this number of candidates.

Definition at line 178 of file `CglDuplicateRow.hpp`.

#### 6.10.4.8 `int CglDuplicateRow::maximumRhs_ [protected]`

Check duplicates if effective rhs  $\leq$  this.

Definition at line 180 of file `CglDuplicateRow.hpp`.

#### 6.10.4.9 `int CglDuplicateRow::sizeDynamic_ [protected]`

Size of dynamic program.

Definition at line 182 of file `CglDuplicateRow.hpp`.

#### 6.10.4.10 `int CglDuplicateRow::mode_ [protected]`

1 do rows, 2 do columns, 3 do both

Definition at line 184 of file `CglDuplicateRow.hpp`.

#### 6.10.4.11 `int CglDuplicateRow::logLevel_ [protected]`

Controls print out.

Definition at line 186 of file `CglDuplicateRow.hpp`.

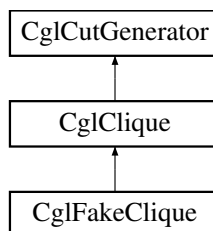
The documentation for this class was generated from the following file:

- [src/CglDuplicateRow/CglDuplicateRow.hpp](#)

## 6.11 CglFakeClique Class Reference

```
#include <CglClique.hpp>
```

Inheritance diagram for CglFakeClique:



### Public Member Functions

- [CglFakeClique](#) (const [CglFakeClique](#) &rhs)  
*Copy constructor.*
- virtual [CglCutGenerator](#) \* [clone](#) () const  
*Clone.*
- [CglFakeClique](#) & [operator=](#) (const [CglFakeClique](#) &rhs)

*Assignment operator.*

- virtual void [generateCuts](#) (const OsiSolverInterface &si, OsiCuts &cs, const [CglTreeInfo](#) info=[CglTreeInfo](#)())

*Generate cuts for the model data contained in si.*

#### Constructors and destructors

- OsiSolverInterface \* [fakeSolver\\_](#)  
*fake solver to use*
- [CglProbing](#) \* [probing\\_](#)  
*Probing object.*
- [CglFakeClique](#) (OsiSolverInterface \*solver=NULL, bool setPacking=false)  
*Default constructor.*
- virtual [~CglFakeClique](#) ()  
*Destructor.*
- void [assignSolver](#) (OsiSolverInterface \*fakeSolver)  
*Assign solver (generator takes over ownership)*

#### Additional Inherited Members

##### 6.11.1 Detailed Description

Definition at line 262 of file CglClique.hpp.

##### 6.11.2 Constructor & Destructor Documentation

###### 6.11.2.1 CglFakeClique::CglFakeClique ( const CglFakeClique & rhs )

Copy constructor.

###### 6.11.2.2 CglFakeClique::CglFakeClique ( OsiSolverInterface \* solver = NULL, bool setPacking = false )

Default constructor.

If the setPacking argument is set to true then [CglFakeClique](#) will assume that the problem in the solverinterface passed to the [generateCuts\(\)](#) method describes a set packing problem, i.e.,

- all variables are binary
- the matrix is a 0-1 matrix
- all constraints are ' $= 1$ ' or ' $\leq 1$ '

Otherwise the user can use the [considerRows\(\)](#) method to set the list of clique rows, that is,

- all coeffs corresponding to binary variables at fractional level is 1
- all other coeffs are non-negative
- the constraint is ' $= 1$ ' or ' $\leq 1$ '.

If the user does not set the list of clique rows then [CglFakeClique](#) will start the [generateCuts\(\)](#) methods by scanning the matrix for them.

6.11.2.3 `virtual CglFakeClique::~~CglFakeClique ( ) [virtual]`

Destructor.

### 6.11.3 Member Function Documentation

6.11.3.1 `virtual CglCutGenerator* CglFakeClique::clone ( ) const [virtual]`

Clone.

Reimplemented from [CglClique](#).

6.11.3.2 `CglFakeClique& CglFakeClique::operator= ( const CglFakeClique & rhs )`

Assignment operator.

6.11.3.3 `virtual void CglFakeClique::generateCuts ( const OsiSolverInterface & si, OsiCuts & cs, const CglTreeInfo info = CglTreeInfo ( ) ) [virtual]`

Generate cuts for the model data contained in si.

The generated cuts are inserted into and returned in the collection of cuts cs.

Reimplemented from [CglClique](#).

6.11.3.4 `void CglFakeClique::assignSolver ( OsiSolverInterface * fakeSolver )`

Assign solver (generator takes over ownership)

### 6.11.4 Member Data Documentation

6.11.4.1 `OsiSolverInterface* CglFakeClique::fakeSolver_ [protected]`

fake solver to use

Definition at line 303 of file [CglClique.hpp](#).

6.11.4.2 `CglProbing* CglFakeClique::probing_ [protected]`

Probing object.

Definition at line 305 of file [CglClique.hpp](#).

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

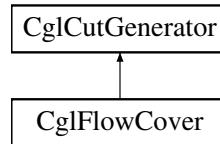
- [src/CglClique/CglClique.hpp](#)

## 6.12 CglFlowCover Class Reference

Lifed Simple Generalized Flow Cover Cut Generator Class.

```
#include <CglFlowCover.hpp>
```

Inheritance diagram for CglFlowCover:



### Public Member Functions

- void [flowPreprocess](#) (const OsiSolverInterface &si)

*Do the following tasks:*

### Generate Cuts

- virtual void [generateCuts](#) (const OsiSolverInterface &si, OsiCuts &cs, const [CglTreeInfo](#) info=[CglTreeInfo](#)())  
*Generate Lifed Simple Generalized flow cover cuts for the model data contained in si.*

### Functions to query and set maximum number of cuts can be generated.

- int [getMaxNumCuts](#) () const
- void [setMaxNumCuts](#) (int mc)

### Constructors and destructors

- [CglFlowCover](#) ()  
*Default constructor.*
- [CglFlowCover](#) (const [CglFlowCover](#) &)  
*Copy constructor.*
- virtual [CglCutGenerator](#) \* [clone](#) () const  
*Clone.*
- [CglFlowCover](#) & [operator=](#) (const [CglFlowCover](#) &rhs)  
*Assignment operator.*
- virtual [~CglFlowCover](#) ()  
*Destructor.*
- virtual std::string [generateCpp](#) (FILE \*fp)  
*Create C++ lines to get to current state.*

### Static Public Member Functions

### Functions to query and set the number of cuts have been generated.

- static int [getNumFlowCuts](#) ()
- static void [setNumFlowCuts](#) (int fc)
- static void [incNumFlowCuts](#) (int fc=1)

### Friends

- void [CglFlowCoverUnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)  
*A function that tests the methods in the [CglFlowCover](#) class.*

## Additional Inherited Members

### 6.12.1 Detailed Description

Lifed Simple Generalized Flow Cover Cut Generator Class.

Definition at line 148 of file CglFlowCover.hpp.

### 6.12.2 Constructor & Destructor Documentation

#### 6.12.2.1 CglFlowCover::CglFlowCover ( )

Default constructor.

#### 6.12.2.2 CglFlowCover::CglFlowCover ( const CglFlowCover & )

Copy constructor.

#### 6.12.2.3 virtual CglFlowCover::~~CglFlowCover ( ) [virtual]

Destructor.

### 6.12.3 Member Function Documentation

#### 6.12.3.1 void CglFlowCover::flowPreprocess ( const OsiSolverInterface & si )

Do the following tasks:

- classify row types
- indentify vubs and vlbs

This function is called by `generateCuts(const OsiSolverInterface & si, OsiCuts & cs).`

#### 6.12.3.2 virtual void CglFlowCover::generateCuts ( const OsiSolverInterface & si, OsiCuts & cs, const CglTreeInfo info = CglTreeInfo() ) [virtual]

Generate Lifed Simple Generalized flow cover cuts for the model data contained in si.

The generated cuts are inserted into and returned in the collection of cuts cs.

Implements [CglCutGenerator](#).

#### 6.12.3.3 int CglFlowCover::getNumCuts ( ) const [inline]

Definition at line 178 of file CglFlowCover.hpp.

#### 6.12.3.4 void CglFlowCover::setMaxNumCuts ( int mc ) [inline]

Definition at line 179 of file CglFlowCover.hpp.

#### 6.12.3.5 static int CglFlowCover::getNumFlowCuts ( ) [inline],[static]

Definition at line 185 of file CglFlowCover.hpp.

6.12.3.6 `static void CglFlowCover::setNumFlowCuts ( int fc ) [inline],[static]`

Definition at line 186 of file CglFlowCover.hpp.

6.12.3.7 `static void CglFlowCover::incNumFlowCuts ( int fc = 1 ) [inline],[static]`

Definition at line 187 of file CglFlowCover.hpp.

6.12.3.8 `virtual CglCutGenerator* CglFlowCover::clone ( ) const [virtual]`

Clone.

Implements [CglCutGenerator](#).

6.12.3.9 `CglFlowCover& CglFlowCover::operator= ( const CglFlowCover & rhs )`

Assignment operator.

6.12.3.10 `virtual std::string CglFlowCover::generateCpp ( FILE * fp ) [virtual]`

Create C++ lines to get to current state.

Reimplemented from [CglCutGenerator](#).

## 6.12.4 Friends And Related Function Documentation

6.12.4.1 `void CglFlowCoverUnitTest ( const OsiSolverInterface * siP, const std::string mpdDir ) [friend]`

A function that tests the methods in the [CglFlowCover](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

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

- `src/CglFlowCover/CglFlowCover.hpp`

## 6.13 CglFlowVUB Class Reference

Variable upper bound class.

```
#include <CglFlowCover.hpp>
```

### Public Member Functions

- [CglFlowVUB](#) ()  
*The Value of the associated upper bound.*
- [CglFlowVUB](#) (const [CglFlowVUB](#) &source)
- [CglFlowVUB](#) & `operator=` (const [CglFlowVUB](#) &rhs)

**Query and set functions for associated 0-1 variable index and value.**

- int [getVar](#) () const



- double `getVal` () const
- void `setVar` (const int v)
- void `setVal` (const double v)

#### Protected Attributes

- int `varInd_`
- double `upper_`

*The index of the associated 0-1 variable.*

#### 6.13.1 Detailed Description

Variable upper bound class.

Definition at line 102 of file CglFlowCover.hpp.

#### 6.13.2 Constructor & Destructor Documentation

##### 6.13.2.1 `CglFlowVUB::CglFlowVUB ( )` [inline]

The Value of the associated upper bound.

Definition at line 109 of file CglFlowCover.hpp.

##### 6.13.2.2 `CglFlowVUB::CglFlowVUB ( const CglFlowVUB & source )` [inline]

Definition at line 111 of file CglFlowCover.hpp.

#### 6.13.3 Member Function Documentation

##### 6.13.3.1 `CglFlowVUB& CglFlowVUB::operator= ( const CglFlowVUB & rhs )` [inline]

Definition at line 116 of file CglFlowCover.hpp.

##### 6.13.3.2 `int CglFlowVUB::getVar ( )` const [inline]

Definition at line 128 of file CglFlowCover.hpp.

##### 6.13.3.3 `double CglFlowVUB::getVal ( )` const [inline]

Definition at line 129 of file CglFlowCover.hpp.

##### 6.13.3.4 `void CglFlowVUB::setVar ( const int v )` [inline]

Definition at line 130 of file CglFlowCover.hpp.

##### 6.13.3.5 `void CglFlowVUB::setVal ( const double v )` [inline]

Definition at line 131 of file CglFlowCover.hpp.

#### 6.13.4 Member Data Documentation

#### 6.13.4.1 int CglFlowVUB::varInd\_ [protected]

Definition at line 105 of file CglFlowCover.hpp.

#### 6.13.4.2 double CglFlowVUB::upper\_ [protected]

The index of the associated 0-1 variable.

Definition at line 106 of file CglFlowCover.hpp.

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

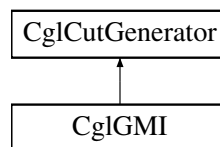
- src/CglFlowCover/CglFlowCover.hpp

## 6.14 CglGMI Class Reference

Gomory cut generator with several cleaning procedures, used to test the numerical safety of the resulting cuts.

```
#include <CglGMI.hpp>
```

Inheritance diagram for CglGMI:



### Public Types

- enum [RejectionType](#) {  
[failureFractionality](#), [failureDynamism](#), [failureViolation](#), [failureSupport](#),  
[failureScale](#) }

*Public enum: all possible reasons for cut rejection.*

### Public Member Functions

#### generateCuts

- virtual void [generateCuts](#) (const OsiSolverInterface &si, OsiCuts &cs, const [CglTreeInfo](#) info=[CglTreeInfo](#)())  
*Generate Gomory Mixed-Integer cuts for the model of the solver interface si.*
- virtual bool [needsOptimalBasis](#) () const  
*Return true if needs optimal basis to do cuts (will return true)*

#### Common Methods

- bool [areEqual](#) (double x, double y, double epsAbs=1e-12, double epsRel=1e-12)
- bool [isZero](#) (double x, double epsZero=1e-20)
- bool [isIntegerValue](#) (double x, double intEpsAbs=1e-9, double intEpsRel=1e-15)

#### Public Methods

- void [setParam](#) (const [CglGMIParam](#) &source)
- [CglGMIParam](#) [getParam](#) () const

- [CglGMIParam](#) & [getParam](#) ()
- void [computeInteger](#) ()
- void [printOptTab](#) (OsiSolverInterface \*solver) const  
*Print the current simplex tableau.*
- void [setTrackRejection](#) (bool value)  
*Set/get tracking of the rejection of cutting planes.*
- bool [getTrackRejection](#) ()
- int [getNumberRejectedCuts](#) ([RejectionType](#) reason)  
*Get number of cuts rejected for given reason; see above.*
- void [resetRejectionCounters](#) ()  
*Reset counters for cut rejection tracking; see above.*
- int [getNumberGeneratedCuts](#) ()  
*Get total number of generated cuts since last [resetRejectionCounters\(\)](#)*

### Constructors and destructors

- [CglGMI](#) ()  
*Default constructor.*
- [CglGMI](#) (const [CglGMIParam](#) &param)  
*Constructor with specified parameters.*
- [CglGMI](#) (const [CglGMI](#) &)  
*Copy constructor.*
- virtual [CglCutGenerator](#) \* [clone](#) () const  
*Clone.*
- [CglGMI](#) & [operator=](#) (const [CglGMI](#) &rhs)  
*Assignment operator.*
- virtual [~CglGMI](#) ()  
*Destructor.*
- virtual std::string [generateCpp](#) (FILE \*fp)  
*Create C++ lines to get to current state.*

### Friends

- void [CglGMIUnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)  
*A function that tests the methods in the [CglGMI](#) class.*

### Additional Inherited Members

#### 6.14.1 Detailed Description

Gomory cut generator with several cleaning procedures, used to test the numerical safety of the resulting cuts.

Definition at line 37 of file [CglGMI.hpp](#).

#### 6.14.2 Member Enumeration Documentation

##### 6.14.2.1 enum [CglGMI::RejectionType](#)

Public enum: all possible reasons for cut rejection.

### Enumerator

***failureFractionality***

*failureDynamism**failureViolation**failureSupport**failureScale*

Definition at line 44 of file CglGMI.hpp.

### 6.14.3 Constructor & Destructor Documentation

#### 6.14.3.1 CglGMI::CglGMI ( )

Default constructor.

#### 6.14.3.2 CglGMI::CglGMI ( const CglGMIParam & param )

Constructor with specified parameters.

#### 6.14.3.3 CglGMI::CglGMI ( const CglGMI & )

Copy constructor.

#### 6.14.3.4 virtual CglGMI::~CglGMI ( ) [virtual]

Destructor.

### 6.14.4 Member Function Documentation

#### 6.14.4.1 virtual void CglGMI::generateCuts ( const OsiSolverInterface & si, OsiCuts & cs, const CglTreeInfo info = CglTreeInfo() ) [virtual]

Generate Gomory Mixed-Integer cuts for the model of the solver interface si.

Insert the generated cuts into OsiCuts cs.

Warning: This generator currently works only with the Lp solvers Clp or Cplex9.0 or higher. It requires access to the optimal tableau and optimal basis inverse and makes assumptions on the way slack variables are added by the solver. The Osi implementations for Clp and Cplex verify these assumptions.

When calling the generator, the solver interface si must contain an optimized problem and information related to the optimal basis must be available through the OsiSolverInterface methods (si->optimalBasisIsAvailable() must return 'true'). It is also essential that the integrality of structural variable i can be obtained using si->isInteger(i).

Implements [CglCutGenerator](#).

#### 6.14.4.2 virtual bool CglGMI::needsOptimalBasis ( ) const [inline], [virtual]

Return true if needs optimal basis to do cuts (will return true)

Reimplemented from [CglCutGenerator](#).

Definition at line 77 of file CglGMI.hpp.

#### 6.14.4.3 bool CglGMI::areEqual ( double x, double y, double epsAbs = 1e-12, double epsRel = 1e-12 ) [inline]

Definition at line 83 of file CglGMI.hpp.

6.14.4.4 `bool CglGMI::isZero ( double x, double epsZero = 1e-20 ) [inline]`

Definition at line 91 of file CglGMI.hpp.

6.14.4.5 `bool CglGMI::isIntegerValue ( double x, double intEpsAbs = 1e-9, double intEpsRel = 1e-15 ) [inline]`

Definition at line 97 of file CglGMI.hpp.

6.14.4.6 `void CglGMI::setParam ( const CglGMIParam & source )`

6.14.4.7 `CglGMIParam CglGMI::getParam ( ) const [inline]`

Definition at line 114 of file CglGMI.hpp.

6.14.4.8 `CglGMIParam& CglGMI::getParam ( ) [inline]`

Definition at line 115 of file CglGMI.hpp.

6.14.4.9 `void CglGMI::computeInteger ( )`

6.14.4.10 `void CglGMI::printOptTab ( OsiSolverInterface * solver ) const`

Print the current simplex tableau.

6.14.4.11 `void CglGMI::setTrackRejection ( bool value )`

Set/get tracking of the rejection of cutting planes.

Note that all rejection related functions will not do anything unless the generator is compiled with the define GMI\_TRACK\_REJECTION

6.14.4.12 `bool CglGMI::getTrackRejection ( )`

6.14.4.13 `int CglGMI::getNumberRejectedCuts ( RejectionType reason )`

Get number of cuts rejected for given reason; see above.

6.14.4.14 `void CglGMI::resetRejectionCounters ( )`

Reset counters for cut rejection tracking; see above.

6.14.4.15 `int CglGMI::getNumberGeneratedCuts ( )`

Get total number of generated cuts since last [resetRejectionCounters\(\)](#)

6.14.4.16 `virtual CglCutGenerator* CglGMI::clone ( ) const [virtual]`

Clone.

Implements [CglCutGenerator](#).

6.14.4.17 `CglGMI& CglGMI::operator= ( const CglGMI & rhs )`

Assignment operator.

6.14.4.18 `virtual std::string CglGMI::generateCpp ( FILE * fp ) [virtual]`

Create C++ lines to get to current state.

Reimplemented from [CglCutGenerator](#).

#### 6.14.5 Friends And Related Function Documentation

6.14.5.1 `void CglGMIUnitTest ( const OsiSolverInterface * siP, const std::string mpdDir )` [*friend*]

A function that tests the methods in the [CglGMI](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

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

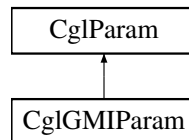
- [src/CglGMI/CglGMI.hpp](#)

## 6.15 CglGMIParam Class Reference

Class collecting parameters for the GMI cut generator.

```
#include <CglGMIParam.hpp>
```

Inheritance diagram for CglGMIParam:



### Public Types

#### Enumerations

- enum [CleaningProcedure](#) {  
[CP\\_CGLLANDP1](#), [CP\\_CGLLANDP2](#), [CP\\_CGLREDSPLIT](#), [CP\\_INTEGRAL\\_CUTS](#),  
[CP\\_CGLLANDP1\\_INT](#), [CP\\_CGLLANDP1\\_SCALEMAX](#), [CP\\_CGLLANDP1\\_SCALERHS](#) }

### Public Member Functions

#### Set/get methods

- void [setInfinity](#) (double value)  
*Aliases for parameter get/set method in the base class [CglParam](#).*
- double [getInfinity](#) () const
- void [setEps](#) (double value)  
*Epsilon for comparing numbers.*
- double [getEps](#) () const
- void [setEpsCoeff](#) (double value)  
*Epsilon for zeroing out coefficients.*
- double [getEpsCoeff](#) () const
- void [setMaxSupport](#) (int value)  
*Maximum support of the cutting planes.*

- int [getMaxSupport](#) () const
- void [setMaxSupportAbs](#) (int value)
  - Alias for consistency with our naming scheme.*
- int [getMaxSupportAbs](#) () const
- int [getMAX\\_SUPPORT\\_ABS](#) () const
- virtual void [setAway](#) (double value)
  - Set AWAY, the minimum distance from being integer used for selecting rows for cut generation; all rows whose pivot variable should be integer but is more than away from integrality will be selected; Default: 0.005.*
- double [getAway](#) () const
  - Get value of away.*
- void [setAWAY](#) (double value)
  - Aliases.*
- double [getAWAY](#) () const
- virtual void [setEPS\\_ELIM](#) (double value)
  - Set the value of EPS\_ELIM, epsilon for values of coefficients when eliminating slack variables; Default: 0.*
- double [getEPS\\_ELIM](#) () const
  - Get the value of EPS\_ELIM.*
- void [setEpsElim](#) (double value)
  - Aliases.*
- double [getEpsElim](#) () const
- virtual void [setEPS\\_RELAX\\_ABS](#) (double value)
  - Set EPS\_RELAX\_ABS.*
- double [getEPS\\_RELAX\\_ABS](#) () const
  - Get value of EPS\_RELAX\_ABS.*
- void [setEpsRelaxAbs](#) (double value)
  - Aliases.*
- double [getEpsRelaxAbs](#) () const
- virtual void [setEPS\\_RELAX\\_REL](#) (double value)
  - Set EPS\_RELAX\_REL.*
- double [getEPS\\_RELAX\\_REL](#) () const
  - Get value of EPS\_RELAX\_REL.*
- void [setEpsRelaxRel](#) (double value)
  - Aliases.*
- double [getEpsRelaxRel](#) () const
- virtual void [setMAXDYN](#) (double value)
- double [getMAXDYN](#) () const
  - Get the value of MAXDYN.*
- void [setMaxDyn](#) (double value)
  - Aliases.*
- double [getMaxDyn](#) () const
- virtual void [setMINVIOL](#) (double value)
  - Set the value of MINVIOL, the minimum violation for the current basic solution in a generated cut.*
- double [getMINVIOL](#) () const
  - Get the value of MINVIOL.*
- void [setMinViol](#) (double value)
  - Aliases.*
- double [getMinViol](#) () const
- virtual void [setMAX\\_SUPPORT\\_REL](#) (double value)
  - Set the value of MAX\_SUPPORT\_REL, the factor contributing to the maximum support relative to the number of columns.*
- double [getMAX\\_SUPPORT\\_REL](#) () const
  - Get the value of MINVIOL.*
- void [setMaxSupportRel](#) (double value)
  - Aliases.*
- double [getMaxSupportRel](#) () const
- virtual void [setUSE\\_INTSLACKS](#) (bool value)

- Set the value of USE\_INTSLACKS.*
  - bool [getUSE\\_INTSLACKS](#) () const
  - Get the value of USE\_INTSLACKS.*
  - void [setUseIntSlacks](#) (bool value)
  - Aliases.*
  - int [getUseIntSlacks](#) () const
  - virtual void [setCHECK\\_DUPLICATES](#) (bool value)
  - Set the value of CHECK\_DUPLICATES.*
  - bool [getCHECK\\_DUPLICATES](#) () const
  - Get the value of CHECK\_DUPLICATES.*
  - void [setCheckDuplicates](#) (bool value)
  - Aliases.*
  - bool [getCheckDuplicates](#) () const
  - virtual void [setCLEAN\\_PROC](#) (CleaningProcedure value)
  - Set the value of CLEAN\_PROC.*
  - CleaningProcedure [getCLEAN\\_PROC](#) () const
  - Get the value of CLEAN\_PROC.*
  - void [setCleanProc](#) (CleaningProcedure value)
  - Aliases.*
  - CleaningProcedure [getCleaningProcedure](#) () const
  - virtual void [setINTEGRAL\\_SCALE\\_CONT](#) (bool value)
  - Set the value of INTEGRAL\_SCALE\_CONT.*
  - bool [getINTEGRAL\\_SCALE\\_CONT](#) () const
  - Get the value of INTEGRAL\_SCALE\_CONT.*
  - void [setIntegralScaleCont](#) (bool value)
  - Aliases.*
  - bool [getIntegralScaleCont](#) () const
  - virtual void [setENFORCE\\_SCALING](#) (bool value)
  - Set the value of ENFORCE\_SCALING.*
  - bool [getENFORCE\\_SCALING](#) () const
  - Get the value of ENFORCE\_SCALING.*
  - void [setEnforceScaling](#) (bool value)
  - Aliases.*
  - bool [getEnforcescalting](#) () const

### Constructors and destructors

- [CglGMIPParam](#) (double eps=1e-12, double away=0.005, double eps\_coeff=1e-11, double eps\_elim=0, double eps\_relax\_abs=1e-11, double eps\_relax\_rel=1e-13, double max\_dyn=1e6, double min\_viol=1e-4, int max\_supp\_abs=1000, double max\_supp\_rel=0.1, CleaningProcedure clean\_proc=CP\_CGLLANDP1, bool use\_int\_slacks=false, bool check\_duplicates=false, bool integral\_scale\_cont=false, bool enforce\_scaling=true)
- Default constructor.*
- [CglGMIPParam](#) ([CglParam](#) &source, double away=0.005, double eps\_elim=1e-12, double eps\_relax\_abs=1e-11, double eps\_relax\_rel=1e-13, double max\_dyn=1e6, double min\_viol=1e-4, double max\_supp\_rel=0.1, CleaningProcedure clean\_proc=CP\_CGLLANDP1, bool use\_int\_slacks=false, bool check\_duplicates=false, bool integral\_scale\_cont=false, bool enforce\_scaling=true)
- Constructor from [CglParam](#).*
- [CglGMIPParam](#) (const [CglGMIPParam](#) &source)
- Copy constructor.*
- virtual [CglGMIPParam](#) \* [clone](#) () const
- Clone.*
- virtual [CglGMIPParam](#) & [operator=](#) (const [CglGMIPParam](#) &rhs)
- Assignment operator.*
- virtual [~CglGMIPParam](#) ()
- Destructor.*



## Protected Attributes

## Parameters

- double [AWAY](#)  
*Use row only if pivot variable should be integer but is more than AWAY from being integer.*
- double [EPS\\_ELIM](#)  
*Epsilon for value of coefficients when eliminating slack variables.*
- double [EPS\\_RELAX\\_ABS](#)  
*Value added to the right hand side of each generated cut to relax it.*
- double [EPS\\_RELAX\\_REL](#)  
*For a generated cut with right hand side  $rhs\_val$ ,  $EPS\_RELAX\_EPS * fabs(rhs\_val)$  is used to relax the constraint.*
- double [MAXDYN](#)  
*Maximum ratio between largest and smallest non zero coefficients in a cut.*
- double [MINVIOL](#)  
*Minimum violation for the current basic solution in a generated cut.*
- double [MAX\\_SUPPORT\\_REL](#)  
*Maximum support relative to number of columns.*
- [CleaningProcedure](#) [CLEAN\\_PROC](#)  
*Which cleaning procedure should be used?*
- bool [USE\\_INTSLACKS](#)  
*Use integer slacks to generate cuts if  $USE\_INTSLACKS = 1$ .*
- bool [CHECK\\_DUPLICATES](#)  
*Check for duplicates when adding the cut to the collection?*
- bool [INTEGRAL\\_SCALE\\_CONT](#)  
*Should we try to rescale cut coefficients on continuous variables so that they become integral, or do we only rescale coefficients on integral variables? Used only by cleaning procedure that try the integral scaling.*
- bool [ENFORCE\\_SCALING](#)  
*Should we discard badly scaled cuts (according to the scaling procedure in use)? If false, `CglGMI::scaleCut` always returns true, even though it still scales cuts whenever possible, but not cut is rejected for scaling.*

## 6.15.1 Detailed Description

Class collecting parameters for the GMI cut generator.

Parameters of the generator are listed below. Modifying the default values for parameters other than the last four might result in invalid cuts.

- **MAXDYN**: Maximum ratio between largest and smallest non zero coefficients in a cut. See method [setMAXDYN\(\)](#).
- **EPS\_ELIM**: Precision for deciding if a coefficient is zero when eliminating slack variables. See method [setEPS\\_ELIM\(\)](#).
- **MINVIOL**: Minimum violation for the current basic solution in a generated cut. See method [setMINVIOL\(\)](#).
- **USE\_INTSLACKS**: Use integer slacks to generate cuts. (not implemented yet, will be in the future). See method [setUSE\\_INTSLACKS\(\)](#).
- **AWAY**: Look only at basic integer variables whose current value is at least this value away from being integer. See method [setAway\(\)](#).
- **CHECK\_DUPLICATES**: Should we check for duplicates when adding a cut to the collection? Can be slow. Default 0 - do not check, add cuts anyway.
- **CLEAN\_PROC**: Cleaning procedure that should be used. Look below at the enumeration `CleaningProcedure` for possible values.

- **INTEGRAL\_SCALE\_CONT**: If we try to scale cut coefficients so that they become integral, do we also scale on continuous variables? Default 0 - do not scale continuous vars. Used only if **CLEAN\_PROC** does integral scaling.
- **ENFORCE\_SCALING**: Discard badly scaled cuts, or keep them (unscaled). Default 1 - yes.

Definition at line 52 of file CglGMIPParam.hpp.

## 6.15.2 Member Enumeration Documentation

### 6.15.2.1 enum CglGMIPParam::CleaningProcedure

Enumerator

**CP\_CGLLANDP1**  
**CP\_CGLLANDP2**  
**CP\_CGLREDSPLIT**  
**CP\_INTEGRAL\_CUTS**  
**CP\_CGLLANDP1\_INT**  
**CP\_CGLLANDP1\_SCALEMAX**  
**CP\_CGLLANDP1\_SCALERHS**

Definition at line 57 of file CglGMIPParam.hpp.

## 6.15.3 Constructor & Destructor Documentation

**6.15.3.1** CglGMIPParam::CglGMIPParam ( double *eps* = 1e-12, double *away* = 0.005, double *eps\_coeff* = 1e-11, double *eps\_elim* = 0, double *eps\_relax\_abs* = 1e-11, double *eps\_relax\_rel* = 1e-13, double *max\_dyn* = 1e6, double *min\_viol* = 1e-4, int *max\_supp\_abs* = 1000, double *max\_supp\_rel* = 0.1, CleaningProcedure *clean\_proc* = CP\_CGLLANDP1, bool *use\_int\_slacks* = false, bool *check\_duplicates* = false, bool *integral\_scale\_cont* = false, bool *enforce\_scaling* = true )

Default constructor.

**6.15.3.2** CglGMIPParam::CglGMIPParam ( CglParam & *source*, double *away* = 0.005, double *eps\_elim* = 1e-12, double *eps\_relax\_abs* = 1e-11, double *eps\_relax\_rel* = 1e-13, double *max\_dyn* = 1e6, double *min\_viol* = 1e-4, double *max\_supp\_rel* = 0.1, CleaningProcedure *clean\_proc* = CP\_CGLLANDP1, bool *use\_int\_slacks* = false, bool *check\_duplicates* = false, bool *integral\_scale\_cont* = false, bool *enforce\_scaling* = true )

Constructor from [CglParam](#).

**6.15.3.3** CglGMIPParam::CglGMIPParam ( const CglGMIPParam & *source* )

Copy constructor.

**6.15.3.4** virtual CglGMIPParam::~CglGMIPParam ( ) [virtual]

Destructor.

## 6.15.4 Member Function Documentation

**6.15.4.1** void CglGMIPParam::setInfinity ( double *value* ) [inline]

Aliases for parameter get/set method in the base class [CglParam](#).

Value for Infinity. Default: DBL\_MAX

Definition at line 80 of file CglGMIParam.hpp.

**6.15.4.2** `double CglGMIParam::getInfinity ( ) const [inline]`

Definition at line 81 of file CglGMIParam.hpp.

**6.15.4.3** `void CglGMIParam::setEps ( double value ) [inline]`

Epsilon for comparing numbers.

Default: 1.0e-6

Definition at line 84 of file CglGMIParam.hpp.

**6.15.4.4** `double CglGMIParam::getEps ( ) const [inline]`

Definition at line 85 of file CglGMIParam.hpp.

**6.15.4.5** `void CglGMIParam::setEpsCoeff ( double value ) [inline]`

Epsilon for zeroing out coefficients.

Default: 1.0e-5

Definition at line 88 of file CglGMIParam.hpp.

**6.15.4.6** `double CglGMIParam::getEpsCoeff ( ) const [inline]`

Definition at line 89 of file CglGMIParam.hpp.

**6.15.4.7** `void CglGMIParam::setMaxSupport ( int value ) [inline]`

Maximum support of the cutting planes.

Default: INT\_MAX

Definition at line 92 of file CglGMIParam.hpp.

**6.15.4.8** `int CglGMIParam::getMaxSupport ( ) const [inline]`

Definition at line 93 of file CglGMIParam.hpp.

**6.15.4.9** `void CglGMIParam::setMaxSupportAbs ( int value ) [inline]`

Alias for consistency with our naming scheme.

Definition at line 95 of file CglGMIParam.hpp.

**6.15.4.10** `int CglGMIParam::getMaxSupportAbs ( ) const [inline]`

Definition at line 96 of file CglGMIParam.hpp.

**6.15.4.11** `int CglGMIParam::getMAX_SUPPORT_ABS ( ) const [inline]`

Definition at line 97 of file CglGMIParam.hpp.

6.15.4.12 `virtual void CglGMIPParam::setAway ( double value ) [virtual]`

Set AWAY, the minimum distance from being integer used for selecting rows for cut generation; all rows whose pivot variable should be integer but is more than away from integrality will be selected; Default: 0.005.

6.15.4.13 `double CglGMIPParam::getAway ( ) const [inline]`

Get value of away.

Definition at line 105 of file CglGMIPParam.hpp.

6.15.4.14 `void CglGMIPParam::setAWAY ( double value ) [inline]`

Aliases.

Definition at line 107 of file CglGMIPParam.hpp.

6.15.4.15 `double CglGMIPParam::getAWAY ( ) const [inline]`

Definition at line 108 of file CglGMIPParam.hpp.

6.15.4.16 `virtual void CglGMIPParam::setEPS_ELIM ( double value ) [virtual]`

Set the value of EPS\_ELIM, epsilon for values of coefficients when eliminating slack variables; Default: 0.

6.15.4.17 `double CglGMIPParam::getEPS_ELIM ( ) const [inline]`

Get the value of EPS\_ELIM.

Definition at line 115 of file CglGMIPParam.hpp.

6.15.4.18 `void CglGMIPParam::setEpsElim ( double value ) [inline]`

Aliases.

Definition at line 117 of file CglGMIPParam.hpp.

6.15.4.19 `double CglGMIPParam::getEpsElim ( ) const [inline]`

Definition at line 118 of file CglGMIPParam.hpp.

6.15.4.20 `virtual void CglGMIPParam::setEPS_RELAX_ABS ( double value ) [virtual]`

Set EPS\_RELAX\_ABS.

6.15.4.21 `double CglGMIPParam::getEPS_RELAX_ABS ( ) const [inline]`

Get value of EPS\_RELAX\_ABS.

Definition at line 123 of file CglGMIPParam.hpp.

6.15.4.22 `void CglGMIPParam::setEpsRelaxAbs ( double value ) [inline]`

Aliases.

Definition at line 125 of file CglGMIPParam.hpp.

6.15.4.23 `double CglGMIPParam::getEpsRelaxAbs ( ) const [inline]`

Definition at line 126 of file CglGMIPParam.hpp.

6.15.4.24 `virtual void CglGMIParam::setEPS_RELAX_REL ( double value ) [virtual]`

Set EPS\_RELAX\_REL.

6.15.4.25 `double CglGMIParam::getEPS_RELAX_REL ( ) const [inline]`

Get value of EPS\_RELAX\_REL.

Definition at line 131 of file CglGMIParam.hpp.

6.15.4.26 `void CglGMIParam::setEpsRelaxRel ( double value ) [inline]`

Aliases.

Definition at line 133 of file CglGMIParam.hpp.

6.15.4.27 `double CglGMIParam::getEpsRelaxRel ( ) const [inline]`

Definition at line 134 of file CglGMIParam.hpp.

6.15.4.28 `virtual void CglGMIParam::setMaxDYN ( double value ) [virtual]`

6.15.4.29 `double CglGMIParam::getMAXDYN ( ) const [inline]`

Get the value of MAXDYN.

Definition at line 140 of file CglGMIParam.hpp.

6.15.4.30 `void CglGMIParam::setMaxDyn ( double value ) [inline]`

Aliases.

Definition at line 142 of file CglGMIParam.hpp.

6.15.4.31 `double CglGMIParam::getMaxDyn ( ) const [inline]`

Definition at line 143 of file CglGMIParam.hpp.

6.15.4.32 `virtual void CglGMIParam::setMINVIOL ( double value ) [virtual]`

Set the value of MINVIOL, the minimum violation for the current basic solution in a generated cut.

Default: 1e-7

6.15.4.33 `double CglGMIParam::getMINVIOL ( ) const [inline]`

Get the value of MINVIOL.

Definition at line 149 of file CglGMIParam.hpp.

6.15.4.34 `void CglGMIParam::setMinViol ( double value ) [inline]`

Aliases.

Definition at line 151 of file CglGMIParam.hpp.

6.15.4.35 `double CglGMIParam::getMinViol ( ) const [inline]`

Definition at line 152 of file CglGMIParam.hpp.

**6.15.4.36** `virtual void CglGMIPParam::setMax_SUPPORT_REL ( double value ) [virtual]`

Set the value of MAX\_SUPPORT\_REL, the factor contributing to the maximum support relative to the number of columns.

Maximum allowed support is: MAX\_SUPPORT\_ABS + MAX\_SUPPORT\_REL\*ncols. Default: 0.1

**6.15.4.37** `double CglGMIPParam::getMAX_SUPPORT_REL ( ) const [inline]`

Get the value of MINVIOL.

Definition at line 160 of file CglGMIPParam.hpp.

**6.15.4.38** `void CglGMIPParam::setMaxSupportRel ( double value ) [inline]`

Aliases.

Definition at line 162 of file CglGMIPParam.hpp.

**6.15.4.39** `double CglGMIPParam::getMaxSupportRel ( ) const [inline]`

Definition at line 163 of file CglGMIPParam.hpp.

**6.15.4.40** `virtual void CglGMIPParam::setUSE_INTSLACKS ( bool value ) [virtual]`

Set the value of USE\_INTSLACKS.

Default: 0

**6.15.4.41** `bool CglGMIPParam::getUSE_INTSLACKS ( ) const [inline]`

Get the value of USE\_INTSLACKS.

Definition at line 168 of file CglGMIPParam.hpp.

**6.15.4.42** `void CglGMIPParam::setUseIntSlacks ( bool value ) [inline]`

Aliases.

Definition at line 170 of file CglGMIPParam.hpp.

**6.15.4.43** `int CglGMIPParam::getUseIntSlacks ( ) const [inline]`

Definition at line 171 of file CglGMIPParam.hpp.

**6.15.4.44** `virtual void CglGMIPParam::setCHECK_DUPLICATES ( bool value ) [virtual]`

Set the value of CHECK\_DUPLICATES.

Default: 0

**6.15.4.45** `bool CglGMIPParam::getCHECK_DUPLICATES ( ) const [inline]`

Get the value of CHECK\_DUPLICATES.

Definition at line 176 of file CglGMIPParam.hpp.

**6.15.4.46** `void CglGMIPParam::setCheckDuplicates ( bool value ) [inline]`

Aliases.

Definition at line 178 of file CglGMIPParam.hpp.

**6.15.4.47** `bool CglGMIParam::getCheckDuplicates ( ) const` `[inline]`

Definition at line 179 of file CglGMIParam.hpp.

**6.15.4.48** `virtual void CglGMIParam::setCLEAN_PROC ( CleaningProcedure value )` `[virtual]`

Set the value of CLEAN\_PROC.

Default: CP\_CGLLANDP1

**6.15.4.49** `CleaningProcedure CglGMIParam::getCLEAN_PROC ( ) const` `[inline]`

Get the value of CLEAN\_PROC.

Definition at line 184 of file CglGMIParam.hpp.

**6.15.4.50** `void CglGMIParam::setCleanProc ( CleaningProcedure value )` `[inline]`

Aliases.

Definition at line 186 of file CglGMIParam.hpp.

**6.15.4.51** `CleaningProcedure CglGMIParam::getCleaningProcedure ( ) const` `[inline]`

Definition at line 187 of file CglGMIParam.hpp.

**6.15.4.52** `virtual void CglGMIParam::setINTEGRAL_SCALE_CONT ( bool value )` `[virtual]`

Set the value of INTEGRAL\_SCALE\_CONT.

Default: 0

**6.15.4.53** `bool CglGMIParam::getINTEGRAL_SCALE_CONT ( ) const` `[inline]`

Get the value of INTEGRAL\_SCALE\_CONT.

Definition at line 192 of file CglGMIParam.hpp.

**6.15.4.54** `void CglGMIParam::setIntegralScaleCont ( bool value )` `[inline]`

Aliases.

Definition at line 194 of file CglGMIParam.hpp.

**6.15.4.55** `bool CglGMIParam::getIntegralScaleCont ( ) const` `[inline]`

Definition at line 195 of file CglGMIParam.hpp.

**6.15.4.56** `virtual void CglGMIParam::setENFORCE_SCALING ( bool value )` `[virtual]`

Set the value of ENFORCE\_SCALING.

Default: 1

**6.15.4.57** `bool CglGMIParam::getENFORCE_SCALING ( ) const` `[inline]`

Get the value of ENFORCE\_SCALING.

Definition at line 200 of file CglGMIParam.hpp.

6.15.4.58 `void CglGMIPParam::setEnforceScaling ( bool value ) [inline]`

Aliases.

Definition at line 202 of file CglGMIPParam.hpp.

6.15.4.59 `bool CglGMIPParam::getEnforcescalting ( ) const [inline]`

Definition at line 203 of file CglGMIPParam.hpp.

6.15.4.60 `virtual CglGMIPParam* CglGMIPParam::clone ( ) const [virtual]`

Clone.

Reimplemented from [CglParam](#).

6.15.4.61 `virtual CglGMIPParam& CglGMIPParam::operator= ( const CglGMIPParam & rhs ) [virtual]`

Assignment operator.

## 6.15.5 Member Data Documentation

6.15.5.1 `double CglGMIPParam::AWAY [protected]`

Use row only if pivot variable should be integer but is more than AWAY from being integer.

Definition at line 261 of file CglGMIPParam.hpp.

6.15.5.2 `double CglGMIPParam::EPS_ELIM [protected]`

Epsilon for value of coefficients when eliminating slack variables.

Default: 0.

Definition at line 265 of file CglGMIPParam.hpp.

6.15.5.3 `double CglGMIPParam::EPS_RELAX_ABS [protected]`

Value added to the right hand side of each generated cut to relax it.

Default: 1e-11

Definition at line 269 of file CglGMIPParam.hpp.

6.15.5.4 `double CglGMIPParam::EPS_RELAX_REL [protected]`

For a generated cut with right hand side *rhs\_val*, `EPS_RELAX_EPS * fabs(rhs_val)` is used to relax the constraint.

Default: 1.e-13

Definition at line 274 of file CglGMIPParam.hpp.

6.15.5.5 `double CglGMIPParam::MAXDYN [protected]`

Maximum ratio between largest and smallest non zero coefficients in a cut.

Default: 1e6.

Definition at line 278 of file CglGMIPParam.hpp.



**6.15.5.6 double CglGMIParam::MINVIOL** [protected]

Minimum violation for the current basic solution in a generated cut.

Default: 1e-4.

Definition at line 282 of file CglGMIParam.hpp.

**6.15.5.7 double CglGMIParam::MAX\_SUPPORT\_REL** [protected]

Maximum support relative to number of columns.

Must be between 0 and 1. Default: 0.

Definition at line 286 of file CglGMIParam.hpp.

**6.15.5.8 CleaningProcedure CglGMIParam::CLEAN\_PROC** [protected]

Which cleaning procedure should be used?

Definition at line 289 of file CglGMIParam.hpp.

**6.15.5.9 bool CglGMIParam::USE\_INTSLACKS** [protected]

Use integer slacks to generate cuts if USE\_INTSLACKS = 1.

Default: 0.

Definition at line 292 of file CglGMIParam.hpp.

**6.15.5.10 bool CglGMIParam::CHECK\_DUPLICATES** [protected]

Check for duplicates when adding the cut to the collection?

Definition at line 295 of file CglGMIParam.hpp.

**6.15.5.11 bool CglGMIParam::INTEGRAL\_SCALE\_CONT** [protected]

Should we try to rescale cut coefficients on continuous variables so that they become integral, or do we only rescale coefficients on integral variables? Used only by cleaning procedure that try the integral scaling.

Definition at line 301 of file CglGMIParam.hpp.

**6.15.5.12 bool CglGMIParam::ENFORCE\_SCALING** [protected]

Should we discard badly scaled cuts (according to the scaling procedure in use)? If false, CglGMI::scaleCut always returns true, even though it still scales cuts whenever possible, but not cut is rejected for scaling.

Default true. Used only by cleaning procedure that try to scale.

Definition at line 308 of file CglGMIParam.hpp.

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

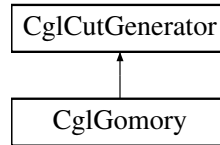
- src/CglGMI/[CglGMIParam.hpp](#)

**6.16 CglGomory Class Reference**

Gomory Cut Generator Class.

```
#include <CglGomory.hpp>
```

Inheritance diagram for CglGomory:



## Public Member Functions

### Generate Cuts

- virtual void [generateCuts](#) (const OsiSolverInterface &si, OsiCuts &cs, const [CglTreeInfo](#) info=[CglTreeInfo](#)())  
*Generate Mixed Integer Gomory cuts for the model of the solver interface, si.*
- int [generateCuts](#) (const OsiRowCutDebugger \*debugger, OsiCuts &cs, const CoinPackedMatrix &columnCopy, const CoinPackedMatrix &rowCopy, const double \*colsol, const double \*colLower, const double \*colUpper, const double \*rowLower, const double \*rowUpper, const char \*intVar, const CoinWarmStartBasis \*warm, const [CglTreeInfo](#) info=[CglTreeInfo](#)())  
*Generates cuts given matrix and solution etc, returns number of cuts generated.*
- int [generateCuts](#) (const OsiRowCutDebugger \*debugger, OsiCuts &cs, const CoinPackedMatrix &columnCopy, const double \*colsol, const double \*colLower, const double \*colUpper, const double \*rowLower, const double \*rowUpper, const char \*intVar, const CoinWarmStartBasis \*warm, const [CglTreeInfo](#) info=[CglTreeInfo](#)())  
*Generates cuts given matrix and solution etc, returns number of cuts generated (no row copy passed in)*
- virtual bool [needsOptimalBasis](#) () const  
*Return true if needs optimal basis to do cuts (will return true)*

### Change way Gomory works

- void [passInOriginalSolver](#) (OsiSolverInterface \*solver)  
*Pass in a copy of original solver (clone it)*
- OsiSolverInterface \* [originalSolver](#) () const  
*Returns original solver.*
- void [setGomoryType](#) (int type)  
*Set type - 0 normal, 1 add original matrix one, 2 replace.*
- int [gomoryType](#) () const  
*Return type.*

### Change limit on how many variables in cut (default 50)

- void [setLimit](#) (int limit)  
*Set.*
- int [getLimit](#) () const  
*Get.*
- void [setLimitAtRoot](#) (int limit)  
*Set at root (if < normal then use normal)*
- int [getLimitAtRoot](#) () const  
*Get at root.*
- virtual int [maximumLengthOfCutInTree](#) () const  
*Return maximum length of cut in tree.*

### Change criterion on which variables to look at. All ones

*more than "away" away from integrality will be investigated (default 0.05)*

- void [setAway](#) (double value)  
*Set away.*
- double [getAway](#) () const  
*Get away.*
- void [setAwayAtRoot](#) (double value)  
*Set away at root.*
- double [getAwayAtRoot](#) () const  
*Get away at root.*

### Change criterion on which the cut id relaxed if the code

*thinks the factorization has inaccuracies.*

*The relaxation to RHS is smallest of - 1) 1.0e-4 2) conditionNumberMultiplier \* condition number of factorization 3) largestFactorMultiplier \* largest (dual\*element) forming tableau row*

- void [setConditionNumberMultiplier](#) (double value)  
*Set ConditionNumberMultiplier.*
- double [getConditionNumberMultiplier](#) () const  
*Get ConditionNumberMultiplier.*
- void [setLargestFactorMultiplier](#) (double value)  
*Set LargestFactorMultiplier.*
- double [getLargestFactorMultiplier](#) () const  
*Get LargestFactorMultiplier.*

### change factorization

- void [useAlternativeFactorization](#) (bool yes=true)  
*Set/unset alternative factorization.*
- bool [alternativeFactorization](#) () const  
*Get whether alternative factorization being used.*

### Constructors and destructors

- [CglGomory](#) ()  
*Default constructor.*
- [CglGomory](#) (const [CglGomory](#) &)  
*Copy constructor.*
- virtual [CglCutGenerator](#) \* [clone](#) () const  
*Clone.*
- [CglGomory](#) & [operator=](#) (const [CglGomory](#) &rhs)  
*Assignment operator.*
- virtual [~CglGomory](#) ()  
*Destructor.*
- virtual std::string [generateCpp](#) (FILE \*fp)  
*Create C++ lines to get to current state.*
- virtual void [refreshSolver](#) (OsiSolverInterface \*solver)  
*This can be used to refresh any inforamtion.*

### Friends

- void [CglGomoryUnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)  
*A function that tests the methods in the [CglGomory](#) class.*

## Additional Inherited Members

## 6.16.1 Detailed Description

Gomory Cut Generator Class.

Definition at line 14 of file CglGomory.hpp.

## 6.16.2 Constructor &amp; Destructor Documentation

## 6.16.2.1 CglGomory::CglGomory ( )

Default constructor.

## 6.16.2.2 CglGomory::CglGomory ( const CglGomory &amp; )

Copy constructor.

## 6.16.2.3 virtual CglGomory::~~CglGomory ( ) [virtual]

Destructor.

## 6.16.3 Member Function Documentation

## 6.16.3.1 virtual void CglGomory::generateCuts ( const OsiSolverInterface &amp; si, OsiCuts &amp; cs, const CglTreeInfo info = CglTreeInfo ( ) ) [virtual]

Generate Mixed Integer Gomory cuts for the model of the solver interface, si.

Insert the generated cuts into OsiCut, cs.

There is a limit option, which will only generate cuts with less than this number of entries.

We can also only look at 0-1 variables a certain distance from integer.

Implements [CglCutGenerator](#).

## 6.16.3.2 int CglGomory::generateCuts ( const OsiRowCutDebugger \* debugger, OsiCuts &amp; cs, const CoinPackedMatrix &amp; columnCopy, const CoinPackedMatrix &amp; rowCopy, const double \* colSol, const double \* colLower, const double \* colUpper, const double \* rowLower, const double \* rowUpper, const char \* intVar, const CoinWarmStartBasis \* warm, const CglTreeInfo info = CglTreeInfo ( ) )

Generates cuts given matrix and solution etc, returns number of cuts generated.

## 6.16.3.3 int CglGomory::generateCuts ( const OsiRowCutDebugger \* debugger, OsiCuts &amp; cs, const CoinPackedMatrix &amp; columnCopy, const double \* colSol, const double \* colLower, const double \* colUpper, const double \* rowLower, const double \* rowUpper, const char \* intVar, const CoinWarmStartBasis \* warm, const CglTreeInfo info = CglTreeInfo ( ) )

Generates cuts given matrix and solution etc, returns number of cuts generated (no row copy passed in)

## 6.16.3.4 virtual bool CglGomory::needsOptimalBasis ( ) const [inline], [virtual]

Return true if needs optimal basis to do cuts (will return true)

Reimplemented from [CglCutGenerator](#).

Definition at line 61 of file CglGomory.hpp.

#### 6.16.3.5 void CglGomory::passInOriginalSolver ( OsiSolverInterface \* *solver* )

Pass in a copy of original solver (clone it)

#### 6.16.3.6 OsiSolverInterface\* CglGomory::originalSolver ( ) const [inline]

Returns original solver.

Definition at line 69 of file CglGomory.hpp.

#### 6.16.3.7 void CglGomory::setGomoryType ( int *type* ) [inline]

Set type - 0 normal, 1 add original matrix one, 2 replace.

Definition at line 72 of file CglGomory.hpp.

#### 6.16.3.8 int CglGomory::gomoryType ( ) const [inline]

Return type.

Definition at line 75 of file CglGomory.hpp.

#### 6.16.3.9 void CglGomory::setLimit ( int *limit* )

Set.

#### 6.16.3.10 int CglGomory::getLimit ( ) const

Get.

#### 6.16.3.11 void CglGomory::setLimitAtRoot ( int *limit* )

Set at root (if <normal then use normal)

#### 6.16.3.12 int CglGomory::getLimitAtRoot ( ) const

Get at root.

#### 6.16.3.13 virtual int CglGomory::maxLengthOfCutInTree ( ) const [virtual]

Return maximum length of cut in tree.

Reimplemented from [CglCutGenerator](#).

#### 6.16.3.14 void CglGomory::setAway ( double *value* )

Set away.

#### 6.16.3.15 double CglGomory::getAway ( ) const

Get away.

#### 6.16.3.16 void CglGomory::setAwayAtRoot ( double *value* )

Set away at root.

#### 6.16.3.17 double CglGomory::getAwayAtRoot ( ) const

Get away at root.

6.16.3.18 void CglGomory::setConditionNumberMultiplier ( double *value* )

Set ConditionNumberMultiplier.

6.16.3.19 double CglGomory::getConditionNumberMultiplier ( ) const

Get ConditionNumberMultiplier.

6.16.3.20 void CglGomory::setLargestFactorMultiplier ( double *value* )

Set LargestFactorMultiplier.

6.16.3.21 double CglGomory::getLargestFactorMultiplier ( ) const

Get LargestFactorMultiplier.

6.16.3.22 void CglGomory::useAlternativeFactorization ( bool *yes* =true ) [inline]

Set/unset alternative factorization.

Definition at line 129 of file CglGomory.hpp.

6.16.3.23 bool CglGomory::alternativeFactorization ( ) const [inline]

Get whether alternative factorization being used.

Definition at line 132 of file CglGomory.hpp.

6.16.3.24 virtual CglCutGenerator\* CglGomory::clone ( ) const [virtual]

Clone.

Implements [CglCutGenerator](#).

6.16.3.25 CglGomory& CglGomory::operator= ( const CglGomory & *rhs* )

Assignment operator.

6.16.3.26 virtual std::string CglGomory::generateCpp ( FILE \* *fp* ) [virtual]

Create C++ lines to get to current state.

Reimplemented from [CglCutGenerator](#).

6.16.3.27 virtual void CglGomory::refreshSolver ( OsiSolverInterface \* *solver* ) [virtual]

This can be used to refresh any inforamtion.

Reimplemented from [CglCutGenerator](#).

## 6.16.4 Friends And Related Function Documentation

6.16.4.1 void CglGomoryUnitTest ( const OsiSolverInterface \* *siP*, const std::string *mpdDir* ) [friend]

A function that tests the methods in the [CglGomory](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

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

- [src/CglGomory/CglGomory.hpp](#)

## 6.17 CglHashLink Struct Reference

Only store unique row cuts.

```
#include <CglPreProcess.hpp>
```

### Public Attributes

- int [index](#)
- int [next](#)

#### 6.17.1 Detailed Description

Only store unique row cuts.

Definition at line 454 of file CglPreProcess.hpp.

#### 6.17.2 Member Data Documentation

##### 6.17.2.1 int CglHashLink::index

Definition at line 455 of file CglPreProcess.hpp.

##### 6.17.2.2 int CglHashLink::next

Definition at line 455 of file CglPreProcess.hpp.

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

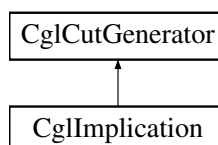
- [src/CglPreProcess/CglPreProcess.hpp](#)

## 6.18 CglImplication Class Reference

This just uses implication info.

```
#include <CglProbing.hpp>
```

Inheritance diagram for CglImplication:



## Public Member Functions

## Generate Cuts

- virtual void [generateCuts](#) (const OsiSolverInterface &si, OsiCuts &cs, const [CglTreeInfo](#) info=[CglTreeInfo](#)())  
*Generate cuts from implication table*  
*Insert generated cuts into the cut set cs.*

## Constructors and destructors

- [CgIImplication](#) ()  
*Default constructor.*
- [CgIImplication](#) ([CglTreeProbingInfo](#) \*info)  
*Constructor with info.*
- [CgIImplication](#) (const [CgIImplication](#) &)  
*Copy constructor.*
- virtual [CglCutGenerator](#) \* [clone](#) () const  
*Clone.*
- [CgIImplication](#) & [operator=](#) (const [CgIImplication](#) &rhs)  
*Assignment operator.*
- virtual [~CgIImplication](#) ()  
*Destructor.*
- virtual std::string [generateCpp](#) (FILE \*fp)  
*Create C++ lines to get to current state.*

## Set implication

- void [setProbingInfo](#) ([CglTreeProbingInfo](#) \*info)  
*Set implication.*

## Additional Inherited Members

## 6.18.1 Detailed Description

This just uses implication info.

Definition at line 468 of file CglProbing.hpp.

## 6.18.2 Constructor &amp; Destructor Documentation

6.18.2.1 [CgIImplication::CgIImplication](#) ( )

Default constructor.

6.18.2.2 [CgIImplication::CgIImplication](#) ( [CglTreeProbingInfo](#) \* *info* )

Constructor with info.

6.18.2.3 [CgIImplication::CgIImplication](#) ( const [CgIImplication](#) & )

Copy constructor.

6.18.2.4 virtual [CgIImplication::~~CgIImplication](#) ( ) [virtual]

Destructor.



### 6.18.3 Member Function Documentation

**6.18.3.1** `virtual void CgIImplication::generateCuts ( const OsiSolverInterface & si, OsiCuts & cs, const CgITreeInfo info = CgITreeInfo() ) [virtual]`

Generate cuts from implication table

Insert generated cuts into the cut set cs.

Implements [CgICutGenerator](#).

**6.18.3.2** `virtual CgICutGenerator* CgIImplication::clone ( ) const [virtual]`

Clone.

Implements [CgICutGenerator](#).

**6.18.3.3** `CgIImplication& CgIImplication::operator= ( const CgIImplication & rhs )`

Assignment operator.

**6.18.3.4** `virtual std::string CgIImplication::generateCpp ( FILE * fp ) [virtual]`

Create C++ lines to get to current state.

Reimplemented from [CgICutGenerator](#).

**6.18.3.5** `void CgIImplication::setProbingInfo ( CgITreeProbingInfo * info ) [inline]`

Set implication.

Definition at line 510 of file CgIProbing.hpp.

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

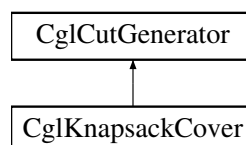
- [src/CgIProbing/CgIProbing.hpp](#)

## 6.19 CgIKnapsackCover Class Reference

Knapsack Cover Cut Generator Class.

`#include <CgIKnapsackCover.hpp>`

Inheritance diagram for CgIKnapsackCover:



### Public Member Functions

- `void setTestedRowIndices (int num, const int *ind)`  
*A method to set which rows should be tested for knapsack covers.*

### Generate Cuts

- virtual void [generateCuts](#) (const OsiSolverInterface &si, OsiCuts &cs, const [CglTreeInfo](#) info=[CglTreeInfo](#)())  
*Generate knapsack cover cuts for the model of the solver interface, si.*

### Constructors and destructors

- [CglKnapsackCover](#) ()  
*Default constructor.*
- [CglKnapsackCover](#) (const [CglKnapsackCover](#) &)  
*Copy constructor.*
- virtual [CglCutGenerator](#) \* [clone](#) () const  
*Clone.*
- [CglKnapsackCover](#) & [operator=](#) (const [CglKnapsackCover](#) &rhs)  
*Assignment operator.*
- virtual [~CglKnapsackCover](#) ()  
*Destructor.*
- virtual std::string [generateCpp](#) (FILE \*fp)  
*Create C++ lines to get to current state.*
- virtual void [refreshSolver](#) (OsiSolverInterface \*solver)  
*This can be used to refresh any information.*

### Sets and gets

- void [setMaxInKnapsack](#) (int value)  
*Set limit on number in knapsack.*
- int [getMaxInKnapsack](#) () const  
*get limit on number in knapsack*
- void [switchOffExpensive](#) ()  
*Switch off expensive cuts.*
- void [switchOnExpensive](#) ()  
*Switch on expensive cuts.*

### Friends

- void [CglKnapsackCoverUnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)  
*A function that tests the methods in the [CglKnapsackCover](#) class.*

### Additional Inherited Members

#### 6.19.1 Detailed Description

Knapsack Cover Cut Generator Class.

Definition at line 15 of file [CglKnapsackCover.hpp](#).

#### 6.19.2 Constructor & Destructor Documentation

##### 6.19.2.1 [CglKnapsackCover::CglKnapsackCover](#) ( )

Default constructor.

##### 6.19.2.2 [CglKnapsackCover::CglKnapsackCover](#) ( const [CglKnapsackCover](#) & )

Copy constructor.

6.19.2.3 `virtual CglKnapsackCover::~~CglKnapsackCover ( ) [virtual]`

Destructor.

### 6.19.3 Member Function Documentation

6.19.3.1 `void CglKnapsackCover::setTestedRowIndices ( int num, const int * ind )`

A method to set which rows should be tested for knapsack covers.

6.19.3.2 `virtual void CglKnapsackCover::generateCuts ( const OsiSolverInterface & si, OsiCuts & cs, const CglTreeInfo info = CglTreeInfo ( ) ) [virtual]`

Generate knapsack cover cuts for the model of the solver interface, si.

Insert the generated cuts into OsiCut, cs.

Implements [CglCutGenerator](#).

6.19.3.3 `virtual CglCutGenerator* CglKnapsackCover::clone ( ) const [virtual]`

Clone.

Implements [CglCutGenerator](#).

6.19.3.4 `CglKnapsackCover& CglKnapsackCover::operator= ( const CglKnapsackCover & rhs )`

Assignment operator.

6.19.3.5 `virtual std::string CglKnapsackCover::generateCpp ( FILE * fp ) [virtual]`

Create C++ lines to get to current state.

Reimplemented from [CglCutGenerator](#).

6.19.3.6 `virtual void CglKnapsackCover::refreshSolver ( OsiSolverInterface * solver ) [virtual]`

This can be used to refresh any information.

Reimplemented from [CglCutGenerator](#).

6.19.3.7 `void CglKnapsackCover::setMaxInKnapsack ( int value ) [inline]`

Set limit on number in knapsack.

Definition at line 62 of file CglKnapsackCover.hpp.

6.19.3.8 `int CglKnapsackCover::getMaxInKnapsack ( ) const [inline]`

get limit on number in knapsack

Definition at line 65 of file CglKnapsackCover.hpp.

6.19.3.9 `void CglKnapsackCover::switchOffExpensive ( ) [inline]`

Switch off expensive cuts.

Definition at line 68 of file CglKnapsackCover.hpp.

#### 6.19.3.10 void CglKnapsackCover::switchOnExpensive ( ) [inline]

Switch on expensive cuts.

Definition at line 71 of file CglKnapsackCover.hpp.

### 6.19.4 Friends And Related Function Documentation

#### 6.19.4.1 void CglKnapsackCoverUnitTest ( const OsiSolverInterface \* *siP*, const std::string *mpdDir* ) [friend]

A function that tests the methods in the [CglKnapsackCover](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

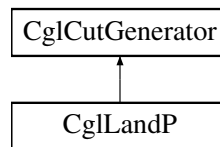
The documentation for this class was generated from the following file:

- [src/CglKnapsackCover/CglKnapsackCover.hpp](#)

## 6.20 CglLandP Class Reference

```
#include <CglLandP.hpp>
```

Inheritance diagram for CglLandP:



### Classes

- class [NoBasisError](#)
- class [Parameters](#)  
*Class storing parameters.*
- class [SimplexInterfaceError](#)

### Public Types

- enum [SelectionRules](#) { [mostNegativeRc](#), [bestPivot](#), [initialReducedCosts](#) }
- enum [ExtraCutsMode](#) { [none](#), [AtOptimalBasis](#), [WhenEnteringBasis](#), [AllViolatedMigs](#) }
- enum [SeparationSpaces](#) { [Fractional](#) =0, [Fractional\\_rc](#), [Full](#) }  
*Space where cuts are optimized.*
- enum [Normalization](#) { [Unweighted](#) = 0, [WeightRHS](#), [WeightLHS](#), [WeightBoth](#) }  
*Normalization.*
- enum [LHSnorm](#) { [L1](#) = 0, [L2](#), [SupportSize](#), [Infinity](#), [Average](#), [Uniform](#) }
- enum [RhsWeightType](#) { [Fixed](#) = 0, [Dynamic](#) }  
*RHS weight in normalization.*

### Public Member Functions

- `CglLandP` (const `CglLandP::Parameters` &params=`CglLandP::Parameters`(), const `LAP::Validator` &validator=`LAP::Validator`())  
*Constructor for the class.*
- `~CglLandP` ()  
*Destructor.*
- `CglLandP` (const `CglLandP` &source)  
*Copy constructor.*
- `CglLandP` & `operator=` (const `CglLandP` &rhs)  
*Assignment operator.*
- `CglCutGenerator` \* `clone` () const  
*Clone function.*
- virtual bool `needsOptimalBasis` () const  
*Return true if needs optimal basis to do cuts.*
- `LAP::Validator` & `validator` ()
- void `setLogLevel` (int level)  
*set level of log for cut generation procedure :*
- `Parameters` & `parameter` ()

### Generate Cuts

- virtual void `generateCuts` (const `OsiSolverInterface` &si, `OsiCuts` &cs, const `CglTreeInfo` info=`CglTreeInfo`())  
*Generate cuts for the model data contained in si.*

### Friends

- class `LAP::CglLandPSimplex`
- class `CftCglp`
- void `CglLandPUnitTest` (`OsiSolverInterface` \*si, const std::string &mpsDir)

### Additional Inherited Members

#### 6.20.1 Detailed Description

Definition at line 49 of file `CglLandP.hpp`.

#### 6.20.2 Member Enumeration Documentation

##### 6.20.2.1 enum `CglLandP::SelectionRules`

#### Enumerator

- mostNegativeRc*** select most negative reduced cost
- bestPivot*** select best possible pivot.
- initialReducedCosts*** Select only those rows which had initially a 0 reduced cost.

Definition at line 58 of file `CglLandP.hpp`.

## 6.20.2.2 enum CgILandP::ExtraCutsMode

Enumerator

***none*** Generate no extra cuts.

***AtOptimalBasis*** Generate cuts from the optimal basis.

***WhenEnteringBasis*** Generate cuts as soon as a structural enters the basis.

***AllViolatedMigs*** Generate all violated Mixed integer Gomory cuts in the course of the optimization.

Definition at line 65 of file CgILandP.hpp.

## 6.20.2.3 enum CgILandP::SeparationSpaces

Space where cuts are optimized.

Enumerator

***Fractional***

***Fractional\_rc*** Use fractional space only for computing reduced costs.

***Full*** Work in full space.

Definition at line 74 of file CgILandP.hpp.

## 6.20.2.4 enum CgILandP::Normalization

Normalization.

Enumerator

***Unweighted***

***WeightRHS***

***WeightLHS***

***WeightBoth***

Definition at line 82 of file CgILandP.hpp.

## 6.20.2.5 enum CgILandP::LHSnorm

Enumerator

***L1***

***L2***

***SupportSize***

***Infinity***

***Average***

***Uniform***

Definition at line 90 of file CgILandP.hpp.



6.20.4.6 void CgILandP::setLogLevel ( int *level* ) [inline]

set level of log for cut generation procedure :

1. for none
2. for log at begin and end of procedure + at some time interval
3. for log at every cut generated

Definition at line 213 of file CgILandP.hpp.

6.20.4.7 Parameters& CgILandP::parameter ( ) [inline]

Definition at line 229 of file CgILandP.hpp.

## 6.20.5 Friends And Related Function Documentation

6.20.5.1 friend class LAP::CgILandPSimplex [friend]

Definition at line 53 of file CgILandP.hpp.

6.20.5.2 friend class CftCgIp [friend]

Definition at line 54 of file CgILandP.hpp.

6.20.5.3 void CgILandPUnitTest ( OsiSolverInterface \* *si*, const std::string & *mpsDir* ) [friend]

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

- src/CgILandP/CgILandP.hpp

## 6.21 LAP::CgILandPSimplex Class Reference

```
#include <CgILandPSimplex.hpp>
```

### Public Member Functions

- [CgILandPSimplex](#) (const OsiSolverInterface &si, const CgILandP::CachedData &cached, const [CgILandP::Parameters](#) &params, [Validator](#) &validator)  
*Usefull onstructor.*
- [~CgILandPSimplex](#) ()  
*Destructor.*
- void [cacheUpdate](#) (const CgILandP::CachedData &cached, bool reducedSpace=0)  
*Update cached information in case of basis change in a round.*
- bool [resetSolver](#) (const CoinWarmStartBasis \*basis)  
*reset the solver to optimal basis*
- bool [optimize](#) (int var, OsiRowCut &[cut](#), const CgILandP::CachedData &cached, const [CgILandP::Parameters](#) &params)  
*Perform pivots to find the best cuts.*
- bool [generateMig](#) (int row, OsiRowCut &[cut](#), const [CgILandP::Parameters](#) &params)



- *Find Gomory cut (i.e.*  
 • `int generateExtraCuts` (const CglLandP::CachedData &cached, const CglLandP::Parameters &params)  
*Find extra constraints in current tableau.*
- `int generateExtraCut` (int i, const CglLandP::CachedData &cached, const CglLandP::Parameters &params)  
*Generate a constraint for a row of the tableau different from the source row.*
- `void genThisBasisMigs` (const CglLandP::CachedData &cached, const CglLandP::Parameters &params)
- `int insertAllExtr` (OsiCuts &cs, CoinRelFltEq eq)  
*insert all extra cuts in cs.*
- `void setLogLevel` (int level)
- `void setSi` (OsiSolverInterface \*si)
- `void freeSi` ()
- `Cuts & extraCuts` ()
- `void loadBasis` (const OsiSolverInterface &si, std::vector< int > &M1, std::vector< int > &M2, int k)
- `int getNumCols` () const
- `int getNumRows` () const
- `const CoinWarmStartBasis * getBasis` () const
- `const int * getNonBasics` () const
- `const int * getBasics` () const
- `void outPivInfo` (int ncuts)

#### Protected Member Functions

- `bool changeBasis` (int incoming, int leaving, int direction, bool modularize)  
*Perform a change in the basis (direction is 1 if leaving variable is going to ub, 0 otherwise)*
- `int fastFindCutImprovingPivotRow` (int &direction, int &gammaSign, double tolerance, bool flagPositiveRows)  
*Find a row which can be used to perform an improving pivot the fast way (i.e., find the leaving variable).*
- `int rescanReducedCosts` (int &direction, int &gammaSign, double tolerance)  
*Rescan reduced costs tables.*
- `int fastFindBestPivotColumn` (int direction, int gammaSign, double pivotTol, double rhsTol, bool reducedSpace, bool allowNonImproving, double &bestSigma, bool modularize)  
*Find the column which leads to the best cut (i.e., find incoming variable).*
- `int findBestPivot` (int &leaving, int &direction, const CglLandP::Parameters &params)  
*Find incoming and leaving variables which lead to the most violated adjacent normalized lift-and-project cut.*
- `double computeCglpObjective` (const TabRow &row, bool modularize=false) const  
*Compute the objective value of the Cglp for given row and rhs (if strengthening shall be applied row should have been modularized).*
- `double strengthenedIntersectionCutCoef` (int i, double alpha\_i, double beta) const  
*return the coefficients of the strengthened intersection cut takes one extra argument seems needs to consider variable type.*
- `double newRowCoefficient` (int j, double gamma) const  
*return the coefficient of the new row (combining row\_k + gamma row\_i).*
- `void createIntersectionCut` (TabRow &row, OsiRowCut &cut) const  
*Create the intersection cut of row k.*
- `double normalizationFactor` (const TabRow &row) const  
*Compute the normalization factor of the cut.*
- `void scaleCut` (OsiRowCut &cut, double factor) const  
*Scale the cut by factor.*
- `void createMIG` (TabRow &row, OsiRowCut &cut) const

- Create strenghtened row.*
- void [pullTableauRow](#) ([TabRow](#) &row) const
- Get the row i of the tableau.*
- void [adjustTableauRow](#) (int var, [TabRow](#) &row, int direction)
- Adjust the row of the tableau to reflect leaving variable direction.*
- void [resetOriginalTableauRow](#) (int var, [TabRow](#) &row, int direction)
- reset the tableau row after a call to adjustTableauRow*
- double [getLoBound](#) (int index) const
- Get lower bound for variable or constraint.*
- double [getUpBound](#) (int index) const
- Get upper bound for variable or constraint.*
- double [getColsolToCut](#) (int index) const
- Access to value in solution to cut (indexed in reduced problem)*
- bool [isGtConst](#) (int index) const
- void [setColsolToCut](#) (int index, double value)
- Access to value in solution to cut (indexed in reduced problem)*
- CoinWarmStartBasis::Status [getStatus](#) (int index) const
- Get the basic status of a variable (structural or slack).*
- bool [isInteger](#) (int index) const
- Say if variable index by i in current tableau is integer.*
- void [computeWeights](#) ([CglLandP::LHSnorm](#) norm, [CglLandP::Normalization](#) type, [CglLandP::RhsWeightType](#) rhs)
- Compute normalization weights.*
- double [normedCoef](#) (double a, int ii) const
- Evenutaly multiply a by w if normed\_weights\_ is not empty.*
- void [printTableau](#) (std::ostream &os)
- print the tableau of current basis.*
- void [printEverything](#) ()
- Print everything .*
- void [printTableauLateX](#) (std::ostream &os)
- print the tableau of current basis.*
- void [printRowLateX](#) (std::ostream &os, int i)
- void [printCutLateX](#) (std::ostream &os, int i)
- void [printCglpBasis](#) (std::ostream &os=std::cout)
- Print CGLP basis corresponding to current tableau and source row.*
- void [get\\_M1\\_M2\\_M3](#) (const [TabRow](#) &row, std::vector< int > &M1, std::vector< int > &M2, std::vector< int > &M3)
- Put variables in M1 M2 and M3 according to their sign.*
- void [eliminate\\_slacks](#) (double \*vec) const
- Put a vector in structural sapce.*

#### Slow versions of the function (old versions do not work).

- double [computeCglpRedCost](#) (int direction, int gammaSign, double tau)
- Compute the reduced cost of Cglp.*
- double [computeRedCostConstantsInRow](#) ()
- Compute the value of sigma and thau (which are constants for a row i as defined in Mike Perregaard thesis.*
- double [computeCglpObjective](#) (double gamma, bool strengthen, [TabRow](#) &row)
- Compute the objective value of the Cglp with linear combination of the two rows by gamma.*

- double [computeCglpObjective](#) (double gamma, bool strengthen)  
*Compute the objective value of the Cglp with linear combination of the row\_k\_ and gamma row\_i\_.*
- int [findCutImprovingPivotRow](#) (int &direction, int &gammaSign, double tolerance)  
*Find a row which can be used to perform an improving pivot return index of the cut or -1 if none exists (i.e., find the leaving variable).*
- int [findBestPivotColumn](#) (int direction, double pivotTol, bool reducedSpace, bool allowDegeneratePivot, bool modularize)  
*Find the column which leads to the best cut (i.e., find incoming variable).*
- int [plotCGLPobj](#) (int direction, double gammaTolerance, double pivotTol, bool reducedSpace, bool allowDegenerate, bool modularize)

### 6.21.1 Detailed Description

Definition at line 42 of file CglLandPSimplex.hpp.

### 6.21.2 Constructor & Destructor Documentation

6.21.2.1 **LAP::CglLandPSimplex::CglLandPSimplex ( const OsiSolverInterface & si, const CglLandP::CachedData & cached, const CglLandP::Parameters & params, Validator & validator )**

Usefull onstructor.

6.21.2.2 **LAP::CglLandPSimplex::~~CglLandPSimplex ( )**

Destructor.

### 6.21.3 Member Function Documentation

6.21.3.1 **void LAP::CglLandPSimplex::cacheUpdate ( const CglLandP::CachedData & cached, bool reducedSpace = 0 )**

Update cached information in case of basis change in a round.

6.21.3.2 **bool LAP::CglLandPSimplex::resetSolver ( const CoinWarmStartBasis \* basis )**

reset the solver to optimal basis

6.21.3.3 **bool LAP::CglLandPSimplex::optimize ( int var, OsiRowCut & cut, const CglLandP::CachedData & cached, const CglLandP::Parameters & params )**

Perform pivots to find the best cuts.

6.21.3.4 **bool LAP::CglLandPSimplex::generateMig ( int row, OsiRowCut & cut, const CglLandP::Parameters & params )**

Find Gomory cut (i.e.

don't do extra setup required for pivots).

6.21.3.5 **int LAP::CglLandPSimplex::generateExtraCuts ( const CglLandP::CachedData & cached, const CglLandP::Parameters & params )**

Find extra constraints in current tableau.

6.21.3.6 `int LAP::CgILandPSimplex::generateExtraCut ( int i, const CgILandP::CachedData & cached, const CgILandP::Parameters & params )`

Generate a constraint for a row of the tableau different from the source row.

6.21.3.7 `void LAP::CgILandPSimplex::genThisBasisMigs ( const CgILandP::CachedData & cached, const CgILandP::Parameters & params )`

6.21.3.8 `int LAP::CgILandPSimplex::insertAllExtr ( OsiCuts & cs, CoinRelFitEq eq )`

insert all extra cuts in *cs*.

6.21.3.9 `void LAP::CgILandPSimplex::setLogLevel ( int level )` `[inline]`

Definition at line 73 of file CgILandPSimplex.hpp.

6.21.3.10 `void LAP::CgILandPSimplex::setSi ( OsiSolverInterface * si )` `[inline]`

Definition at line 79 of file CgILandPSimplex.hpp.

6.21.3.11 `void LAP::CgILandPSimplex::freeSi ( )` `[inline]`

Definition at line 90 of file CgILandPSimplex.hpp.

6.21.3.12 `Cuts& LAP::CgILandPSimplex::extraCuts ( )` `[inline]`

Definition at line 100 of file CgILandPSimplex.hpp.

6.21.3.13 `void LAP::CgILandPSimplex::loadBasis ( const OsiSolverInterface & si, std::vector< int > & M1, std::vector< int > & M2, int k )`

6.21.3.14 `int LAP::CgILandPSimplex::getNumCols ( ) const` `[inline]`

Definition at line 109 of file CgILandPSimplex.hpp.

6.21.3.15 `int LAP::CgILandPSimplex::getNumRows ( ) const` `[inline]`

Definition at line 114 of file CgILandPSimplex.hpp.

6.21.3.16 `const CoinWarmStartBasis* LAP::CgILandPSimplex::getBasis ( ) const` `[inline]`

Definition at line 119 of file CgILandPSimplex.hpp.

6.21.3.17 `const int* LAP::CgILandPSimplex::getNonBasics ( ) const` `[inline]`

Definition at line 123 of file CgILandPSimplex.hpp.

6.21.3.18 `const int* LAP::CgILandPSimplex::getBasics ( ) const` `[inline]`

Definition at line 128 of file CgILandPSimplex.hpp.

6.21.3.19 `void LAP::CgILandPSimplex::outPivInfo ( int ncuts )` `[inline]`

Definition at line 133 of file CgILandPSimplex.hpp.

6.21.3.20 `bool LAP::CglLandPSimplex::changeBasis ( int incoming, int leaving, int direction, bool modularize )`  
`[protected]`

Perform a change in the basis (direction is 1 if leaving variable is going to ub, 0 otherwise)

6.21.3.21 `int LAP::CglLandPSimplex::fastFindCutImprovingPivotRow ( int & direction, int & gammaSign, double tolerance, bool flagPositiveRows )`  
`[protected]`

Find a row which can be used to perform an improving pivot the fast way (i.e., find the leaving variable).

#### Returns

index of the row on which to pivot or -1 if none exists.

6.21.3.22 `int LAP::CglLandPSimplex::rescanReducedCosts ( int & direction, int & gammaSign, double tolerance )`  
`[protected]`

Rescan reduced costs tables.

6.21.3.23 `int LAP::CglLandPSimplex::fastFindBestPivotColumn ( int direction, int gammaSign, double pivotTol, double rhsTol, bool reducedSpace, bool allowNonImproving, double & bestSigma, bool modularize )`  
`[protected]`

Find the column which leads to the best cut (i.e., find incoming variable).

6.21.3.24 `int LAP::CglLandPSimplex::findBestPivot ( int & leaving, int & direction, const CglLandP::Parameters & params )`  
`[protected]`

Find incoming and leaving variables which lead to the most violated adjacent normalized lift-and-project cut.

#### Remarks

At this point reduced costs should be already computed.

#### Returns

incoming variable variable,

#### Parameters

|                  |                   |
|------------------|-------------------|
| <i>leaving</i>   | variable          |
| <i>direction</i> | leaving direction |

6.21.3.25 `double LAP::CglLandPSimplex::computeCglpObjective ( const TabRow & row, bool modularize = false ) const`  
`[protected]`

Compute the objective value of the Cglp for given row and rhs (if strengthening shall be applied row should have been modularized).

6.21.3.26 `double LAP::CglLandPSimplex::strengthenedIntersectionCutCoef ( int i, double alpha_i, double beta ) const`  
`[inline], [protected]`

return the coefficients of the strengthened intersection cut takes one extra argument seems needs to consider variable type.

return the coefficients of the strengthened intersection cut

Definition at line 426 of file CglLandPSimplex.hpp.

6.21.3.27 `double LAP::CglLandPSimplex::newRowCoefficient ( int j, double gamma ) const` [inline],[protected]

return the coefficient of the new row (combining row\_k + gamma row\_i).

Definition at line 444 of file CglLandPSimplex.hpp.

6.21.3.28 `void LAP::CglLandPSimplex::createIntersectionCut ( TabRow & row, OsiRowCut & cut ) const` [protected]

Create the intersection cut of row k.

6.21.3.29 `double LAP::CglLandPSimplex::normalizationFactor ( const TabRow & row ) const` [protected]

Compute the normalization factor of the cut.

6.21.3.30 `void LAP::CglLandPSimplex::scaleCut ( OsiRowCut & cut, double factor ) const` [protected]

Scale the cut by factor.

6.21.3.31 `void LAP::CglLandPSimplex::createMIG ( TabRow & row, OsiRowCut & cut ) const` [protected]

Create strenghtened row.

Create MIG cut from row k

6.21.3.32 `void LAP::CglLandPSimplex::pullTableauRow ( TabRow & row ) const` [protected]

Get the row i of the tableau.

6.21.3.33 `void LAP::CglLandPSimplex::adjustTableauRow ( int var, TabRow & row, int direction )` [protected]

Adjust the row of the tableau to reflect leaving variable direction.

6.21.3.34 `void LAP::CglLandPSimplex::resetOriginalTableauRow ( int var, TabRow & row, int direction )` [protected]

reset the tableau row after a call to adjustTableauRow

6.21.3.35 `double LAP::CglLandPSimplex::getLoBound ( int index ) const` [inline],[protected]

Get lower bound for variable or constraint.

Definition at line 205 of file CglLandPSimplex.hpp.

6.21.3.36 `double LAP::CglLandPSimplex::getUpBound ( int index ) const` [inline],[protected]

Get upper bound for variable or constraint.

Definition at line 210 of file CglLandPSimplex.hpp.

6.21.3.37 `double LAP::CglLandPSimplex::getColSolToCut ( int index ) const` [inline],[protected]

Access to value in solution to cut (indexed in reduced problem)

Definition at line 215 of file CglLandPSimplex.hpp.

6.21.3.38 `bool LAP::CglLandPSimplex::isGtConst ( int index ) const` [inline],[protected]

Definition at line 219 of file CglLandPSimplex.hpp.

6.21.3.39 void LAP::CgILandPSimplex::setColsolToCut ( int *index*, double *value* ) [inline], [protected]

Access to value in solution to cut (indexed in reduced problem)

Definition at line 224 of file CgILandPSimplex.hpp.

6.21.3.40 CoinWarmStartBasis::Status LAP::CgILandPSimplex::getStatus ( int *index* ) const [inline], [protected]

Get the basic status of a variable (structural or slack).

Definition at line 229 of file CgILandPSimplex.hpp.

6.21.3.41 bool LAP::CgILandPSimplex::isInteger ( int *index* ) const [inline], [protected]

Say if variable index by *i* in current tableau is integer.

Definition at line 235 of file CgILandPSimplex.hpp.

6.21.3.42 void LAP::CgILandPSimplex::computeWeights ( CgILandP::LHSnorm *norm*, CgILandP::Normalization *type*, CgILandP::RhsWeightType *rhs* ) [protected]

Compute normalization weights.

6.21.3.43 double LAP::CgILandPSimplex::normedCoef ( double *a*, int *ii* ) const [inline], [protected]

Evenutally multiply a by *w* if `normed_weights_` is not empty.

Definition at line 243 of file CgILandPSimplex.hpp.

6.21.3.44 void LAP::CgILandPSimplex::printTableau ( std::ostream & *os* ) [protected]

print the tableau of current basis.

6.21.3.45 void LAP::CgILandPSimplex::printEverything ( ) [protected]

Print everything .

6.21.3.46 void LAP::CgILandPSimplex::printTableauLateX ( std::ostream & *os* ) [protected]

print the tableau of current basis.

6.21.3.47 void LAP::CgILandPSimplex::printRowLateX ( std::ostream & *os*, int *i* ) [protected]

6.21.3.48 void LAP::CgILandPSimplex::printCutLateX ( std::ostream & *os*, int *i* ) [protected]

6.21.3.49 void LAP::CgILandPSimplex::printCglpBasis ( std::ostream & *os* = std::cout ) [protected]

Print CGLP basis corresponding to current tableau and source row.

6.21.3.50 void LAP::CgILandPSimplex::get\_M1\_M2\_M3 ( const TabRow & *row*, std::vector< int > & *M1*, std::vector< int > & *M2*, std::vector< int > & *M3* ) [protected]

Put variables in *M1* *M2* and *M3* according to their sign.

6.21.3.51 void LAP::CgILandPSimplex::eliminate\_slacks ( double \* *vec* ) const [protected]

Put a vector in structural sapce.

6.21.3.52 `double LAP::CglLandPSimplex::computeCglpRedCost ( int direction, int gammaSign, double tau )` [protected]

Compute the reduced cost of Cglp.

6.21.3.53 `double LAP::CglLandPSimplex::computeRedCostConstantsInRow ( )` [protected]

Compute the value of sigma and thau (which are constants for a row i as defined in Mike Perregaard thesis).

6.21.3.54 `double LAP::CglLandPSimplex::computeCglpObjective ( double gamma, bool strengthen, TabRow & row )` [protected]

Compute the objective value of the Cglp with linear combination of the two rows by gamma.

6.21.3.55 `double LAP::CglLandPSimplex::computeCglpObjective ( double gamma, bool strengthen )` [protected]

Compute the objective value of the Cglp with linear combination of the row\_k\_ and gamma row\_i\_.

6.21.3.56 `int LAP::CglLandPSimplex::findCutImprovingPivotRow ( int & direction, int & gammaSign, double tolerance )` [protected]

Find a row which can be used to perform an improving pivot return index of the cut or -1 if none exists (i.e., find the leaving variable).

6.21.3.57 `int LAP::CglLandPSimplex::findBestPivotColumn ( int direction, double pivotTol, bool reducedSpace, bool allowDegeneratePivot, bool modularize )` [protected]

Find the column which leads to the best cut (i.e., find incoming variable).

6.21.3.58 `int LAP::CglLandPSimplex::plotCGLPobj ( int direction, double gammaTolerance, double pivotTol, bool reducedSpace, bool allowDegenerate, bool modularize )` [protected]

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

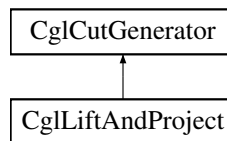
- [src/CglLandP/CglLandPSimplex.hpp](#)

## 6.22 CglLiftAndProject Class Reference

Lift And Project Cut Generator Class.

```
#include <CglLiftAndProject.hpp>
```

Inheritance diagram for CglLiftAndProject:



### Public Member Functions

#### Generate Cuts

- virtual void [generateCuts](#) (const OsiSolverInterface &si, OsiCuts &cs, const [CglTreeInfo](#) info=[CglTreeInfo](#)())  
Generate lift-and-project cuts for the model of the solver interface, si.



- double `getBeta` () const  
*Get the normalization : Either beta=+1 or beta=-1.*
- void `setBeta` (int oneOrMinusOne)  
*Set the normalization : Either beta=+1 or beta=-1.*

### Constructors and destructors

- `CglLiftAndProject` ()  
*Default constructor.*
- `CglLiftAndProject` (const `CglLiftAndProject` &)  
*Copy constructor.*
- virtual `CglCutGenerator * clone` () const  
*Clone.*
- `CglLiftAndProject & operator=` (const `CglLiftAndProject` &rhs)  
*Assignment operator.*
- virtual `~CglLiftAndProject` ()  
*Destructor.*
- virtual std::string `generateCpp` (FILE \*fp)  
*Create C++ lines to get to current state.*

### Friends

- void `CglLiftAndProjectUnitTest` (const `OsiSolverInterface *siP`, const std::string mpdDir)  
*A function that tests the methods in the `CglLiftAndProject` class.*

### Additional Inherited Members

#### 6.22.1 Detailed Description

Lift And Project Cut Generator Class.

Definition at line 13 of file `CglLiftAndProject.hpp`.

#### 6.22.2 Constructor & Destructor Documentation

##### 6.22.2.1 `CglLiftAndProject::CglLiftAndProject` ( )

Default constructor.

##### 6.22.2.2 `CglLiftAndProject::CglLiftAndProject` ( const `CglLiftAndProject` & )

Copy constructor.

##### 6.22.2.3 virtual `CglLiftAndProject::~~CglLiftAndProject` ( ) [virtual]

Destructor.

#### 6.22.3 Member Function Documentation

##### 6.22.3.1 virtual void `CglLiftAndProject::generateCuts` ( const `OsiSolverInterface & si`, `OsiCuts & cs`, const `CglTreeInfo info = CglTreeInfo` () ) [virtual]

Generate lift-and-project cuts for the model of the solver interface, si.

Insert the generated cuts into OsiCut, cs.

Implements [CglCutGenerator](#).

**6.22.3.2** `double CglLiftAndProject::getBeta ( ) const [inline]`

Get the normalization : Either beta=+1 or beta=-1.

Definition at line 30 of file CglLiftAndProject.hpp.

**6.22.3.3** `void CglLiftAndProject::setBeta ( int oneOrMinusOne ) [inline]`

Set the normalization : Either beta=+1 or beta=-1.

Default value is 1.

Definition at line 37 of file CglLiftAndProject.hpp.

**6.22.3.4** `virtual CglCutGenerator* CglLiftAndProject::clone ( ) const [virtual]`

Clone.

Implements [CglCutGenerator](#).

**6.22.3.5** `CglLiftAndProject& CglLiftAndProject::operator= ( const CglLiftAndProject & rhs )`

Assignment operator.

**6.22.3.6** `virtual std::string CglLiftAndProject::generateCpp ( FILE * fp ) [virtual]`

Create C++ lines to get to current state.

Reimplemented from [CglCutGenerator](#).

## 6.22.4 Friends And Related Function Documentation

**6.22.4.1** `void CglLiftAndProjectUnitTest ( const OsiSolverInterface * siP, const std::string mpdDir ) [friend]`

A function that tests the methods in the [CglLiftAndProject](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

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

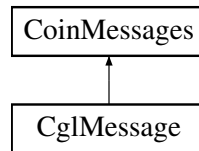
- `src/CglLiftAndProject/CglLiftAndProject.hpp`

## 6.23 CglMessage Class Reference

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

```
#include <CglMessage.hpp>
```

Inheritance diagram for CglMessage:



## Public Member Functions

### Constructors etc

- [CglMessage](#) (Language language=us\_en)  
*Constructor.*

#### 6.23.1 Detailed Description

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

Definition at line 38 of file CglMessage.hpp.

#### 6.23.2 Constructor & Destructor Documentation

##### 6.23.2.1 CglMessage::CglMessage ( Language *language* = us\_en )

Constructor.

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

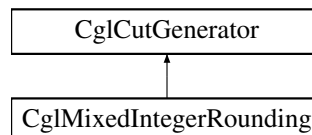
- [src/CglMessage.hpp](#)

## 6.24 CglMixedIntegerRounding Class Reference

Mixed Integer Rounding Cut Generator Class.

```
#include <CglMixedIntegerRounding.hpp>
```

Inheritance diagram for CglMixedIntegerRounding:



## Public Member Functions

### Generate Cuts

- virtual void [generateCuts](#) (const OsiSolverInterface &si, OsiCuts &cs, const [CglTreeInfo](#) info=[CglTreeInfo](#)())  
*Generate Mixed Integer Rounding cuts for the model data contained in si.*

### Constructors and destructors

- [CglMixedIntegerRounding](#) ()  
*Default constructor.*
- [CglMixedIntegerRounding](#) (const int maxaggr, const bool multiply, const int criterion, const int preproc=-1)  
*Alternate Constructor.*
- [CglMixedIntegerRounding](#) (const [CglMixedIntegerRounding](#) &)  
*Copy constructor.*
- virtual [CglCutGenerator](#) \* [clone](#) () const  
*Clone.*
- [CglMixedIntegerRounding](#) & [operator=](#) (const [CglMixedIntegerRounding](#) &rhs)  
*Assignment operator.*
- virtual [~CglMixedIntegerRounding](#) ()  
*Destructor.*
- virtual void [refreshSolver](#) (OsiSolverInterface \*solver)  
*This can be used to refresh any information.*
- virtual std::string [generateCpp](#) (FILE \*fp)  
*Create C++ lines to get to current state.*

### Set and get methods

- void [setMAXAGGR\\_](#) (int maxaggr)  
*Set MAXAGGR\_.*
- int [getMAXAGGR\\_](#) () const  
*Get MAXAGGR\_.*
- void [setMULTIPLY\\_](#) (bool multiply)  
*Set MULTIPLY\_.*
- bool [getMULTIPLY\\_](#) () const  
*Get MULTIPLY\_.*
- void [setCRITERION\\_](#) (int criterion)  
*Set CRITERION\_.*
- int [getCRITERION\\_](#) () const  
*Get CRITERION\_.*
- void [setDoPreproc](#) (int value)  
*Set doPreproc.*
- bool [getDoPreproc](#) () const  
*Get doPreproc.*

### Friends

- void [CglMixedIntegerRoundingUnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)

### Additional Inherited Members

#### 6.24.1 Detailed Description

Mixed Integer Rounding Cut Generator Class.

Definition at line 86 of file [CglMixedIntegerRounding.hpp](#).

#### 6.24.2 Constructor & Destructor Documentation

##### 6.24.2.1 [CglMixedIntegerRounding::CglMixedIntegerRounding](#) ( )

Default constructor.

6.24.2.2 `CglMixedIntegerRounding::CglMixedIntegerRounding ( const int maxaggr, const bool multiply, const int criterion, const int preproc = -1 )`

Alternate Constructor.

6.24.2.3 `CglMixedIntegerRounding::CglMixedIntegerRounding ( const CglMixedIntegerRounding & )`

Copy constructor.

6.24.2.4 `virtual CglMixedIntegerRounding::~~CglMixedIntegerRounding ( ) [virtual]`

Destructor.

### 6.24.3 Member Function Documentation

6.24.3.1 `virtual void CglMixedIntegerRounding::generateCuts ( const OsiSolverInterface & si, OsiCuts & cs, const CglTreeInfo info = CglTreeInfo ( ) ) [virtual]`

Generate Mixed Integer Rounding cuts for the model data contained in *si*.

The generated cuts are inserted in the collection of cuts *cs*.

Implements [CglCutGenerator](#).

6.24.3.2 `virtual CglCutGenerator* CglMixedIntegerRounding::clone ( ) const [virtual]`

Clone.

Implements [CglCutGenerator](#).

6.24.3.3 `CglMixedIntegerRounding& CglMixedIntegerRounding::operator= ( const CglMixedIntegerRounding & rhs )`

Assignment operator.

6.24.3.4 `virtual void CglMixedIntegerRounding::refreshSolver ( OsiSolverInterface * solver ) [virtual]`

This can be used to refresh any information.

Reimplemented from [CglCutGenerator](#).

6.24.3.5 `virtual std::string CglMixedIntegerRounding::generateCpp ( FILE * fp ) [virtual]`

Create C++ lines to get to current state.

Reimplemented from [CglCutGenerator](#).

6.24.3.6 `void CglMixedIntegerRounding::setMaxAGGR_ ( int maxaggr ) [inline]`

Set MAXAGGR\_.

Definition at line 170 of file `CglMixedIntegerRounding.hpp`.

6.24.3.7 `int CglMixedIntegerRounding::getMAXAGGR_ ( ) const [inline]`

Get MAXAGGR\_.

Definition at line 181 of file `CglMixedIntegerRounding.hpp`.

6.24.3.8 void CglMixedIntegerRounding::setMULTIPLY\_( bool *multiply* ) [inline]

Set MULTIPLY\_.

Definition at line 184 of file CglMixedIntegerRounding.hpp.

6.24.3.9 bool CglMixedIntegerRounding::getMULTIPLY\_( ) const [inline]

Get MULTIPLY\_.

Definition at line 187 of file CglMixedIntegerRounding.hpp.

6.24.3.10 void CglMixedIntegerRounding::setCRITERION\_( int *criterion* ) [inline]

Set CRITERION\_.

Definition at line 190 of file CglMixedIntegerRounding.hpp.

6.24.3.11 int CglMixedIntegerRounding::getCRITERION\_( ) const [inline]

Get CRITERION\_.

Definition at line 201 of file CglMixedIntegerRounding.hpp.

6.24.3.12 void CglMixedIntegerRounding::setDoPreproc ( int *value* )

Set doPreproc.

6.24.3.13 bool CglMixedIntegerRounding::getDoPreproc ( ) const

Get doPreproc.

## 6.24.4 Friends And Related Function Documentation

6.24.4.1 void CglMixedIntegerRoundingUnitTest ( const OsiSolverInterface \* *siP*, const std::string *mpdDir* ) [friend]

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

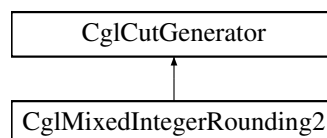
- [src/CglMixedIntegerRounding/CglMixedIntegerRounding.hpp](#)

## 6.25 CglMixedIntegerRounding2 Class Reference

Mixed Integer Rounding Cut Generator Class.

```
#include <CglMixedIntegerRounding2.hpp>
```

Inheritance diagram for CglMixedIntegerRounding2:



## Public Member Functions

### Generate Cuts

- virtual void [generateCuts](#) (const OsiSolverInterface &si, OsiCuts &cs, const [CglTreeInfo](#) info=[CglTreeInfo](#)())  
*Generate Mixed Integer Rounding cuts for the model data contained in si.*

### Constructors and destructors

- [CglMixedIntegerRounding2](#) ()  
*Default constructor.*
- [CglMixedIntegerRounding2](#) (const int maxaggr, const bool multiply, const int criterion, const int preproc=-1)  
*Alternate Constructor.*
- [CglMixedIntegerRounding2](#) (const [CglMixedIntegerRounding2](#) &)  
*Copy constructor.*
- virtual [CglCutGenerator](#) \* [clone](#) () const  
*Clone.*
- [CglMixedIntegerRounding2](#) & [operator=](#) (const [CglMixedIntegerRounding2](#) &rhs)  
*Assignment operator.*
- virtual [~CglMixedIntegerRounding2](#) ()  
*Destructor.*
- virtual void [refreshSolver](#) (OsiSolverInterface \*solver)  
*This can be used to refresh any informtion.*
- virtual std::string [generateCpp](#) (FILE \*fp)  
*Create C++ lines to get to current state.*

### Set and get methods

- void [setMAXAGGR\\_](#) (int maxaggr)  
*Set MAXAGGR\_.*
- int [getMAXAGGR\\_](#) () const  
*Get MAXAGGR\_.*
- void [setMULTIPLY\\_](#) (bool multiply)  
*Set MULTIPLY\_.*
- bool [getMULTIPLY\\_](#) () const  
*Get MULTIPLY\_.*
- void [setCRITERION\\_](#) (int criterion)  
*Set CRITERION\_.*
- int [getCRITERION\\_](#) () const  
*Get CRITERION\_.*
- void [setDoPreproc](#) (int value)  
*Set doPreproc.*
- bool [getDoPreproc](#) () const  
*Get doPreproc.*

## Friends

- void [CglMixedIntegerRounding2UnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)

## Additional Inherited Members

### 6.25.1 Detailed Description

Mixed Integer Rounding Cut Generator Class.

Definition at line 87 of file [CglMixedIntegerRounding2.hpp](#).

## 6.25.2 Constructor &amp; Destructor Documentation

## 6.25.2.1 CglMixedIntegerRounding2::CglMixedIntegerRounding2 ( )

Default constructor.

6.25.2.2 CglMixedIntegerRounding2::CglMixedIntegerRounding2 ( const int *maxaggr*, const bool *multiply*, const int *criterion*, const int *preproc* = -1 )

Alternate Constructor.

## 6.25.2.3 CglMixedIntegerRounding2::CglMixedIntegerRounding2 ( const CglMixedIntegerRounding2 &amp; )

Copy constructor.

## 6.25.2.4 virtual CglMixedIntegerRounding2::~CglMixedIntegerRounding2 ( ) [virtual]

Destructor.

## 6.25.3 Member Function Documentation

6.25.3.1 virtual void CglMixedIntegerRounding2::generateCuts ( const OsiSolverInterface & *si*, OsiCuts & *cs*, const CglTreeInfo *info* = CglTreeInfo ( ) ) [virtual]

Generate Mixed Integer Rounding cuts for the model data contained in *si*.

The generated cuts are inserted in the collection of cuts *cs*.

Implements [CglCutGenerator](#).

## 6.25.3.2 virtual CglCutGenerator\* CglMixedIntegerRounding2::clone ( ) const [virtual]

Clone.

Implements [CglCutGenerator](#).

6.25.3.3 CglMixedIntegerRounding2& CglMixedIntegerRounding2::operator= ( const CglMixedIntegerRounding2 & *rhs* )

Assignment operator.

6.25.3.4 virtual void CglMixedIntegerRounding2::refreshSolver ( OsiSolverInterface \* *solver* ) [virtual]

This can be used to refresh any information.

Reimplemented from [CglCutGenerator](#).

6.25.3.5 virtual std::string CglMixedIntegerRounding2::generateCpp ( FILE \* *fp* ) [virtual]

Create C++ lines to get to current state.

Reimplemented from [CglCutGenerator](#).

6.25.3.6 void CglMixedIntegerRounding2::setMAXAGGR\_ ( int *maxaggr* ) [inline]

Set MAXAGGR\_.

Definition at line 171 of file CglMixedIntegerRounding2.hpp.



6.25.3.7 `int CglMixedIntegerRounding2::getMAXAGGR_ ( ) const [inline]`

Get MAXAGGR\_.

Definition at line 182 of file CglMixedIntegerRounding2.hpp.

6.25.3.8 `void CglMixedIntegerRounding2::setMULTIPLY_ ( bool multiply ) [inline]`

Set MULTIPLY\_.

Definition at line 185 of file CglMixedIntegerRounding2.hpp.

6.25.3.9 `bool CglMixedIntegerRounding2::getMULTIPLY_ ( ) const [inline]`

Get MULTIPLY\_.

Definition at line 188 of file CglMixedIntegerRounding2.hpp.

6.25.3.10 `void CglMixedIntegerRounding2::setCRITERION_ ( int criterion ) [inline]`

Set CRITERION\_.

Definition at line 191 of file CglMixedIntegerRounding2.hpp.

6.25.3.11 `int CglMixedIntegerRounding2::getCRITERION_ ( ) const [inline]`

Get CRITERION\_.

Definition at line 202 of file CglMixedIntegerRounding2.hpp.

6.25.3.12 `void CglMixedIntegerRounding2::setDoPreproc ( int value )`

Set doPreproc.

6.25.3.13 `bool CglMixedIntegerRounding2::getDoPreproc ( ) const`

Get doPreproc.

## 6.25.4 Friends And Related Function Documentation

6.25.4.1 `void CglMixedIntegerRounding2UnitTest ( const OsiSolverInterface * siP, const std::string mpdDir ) [friend]`

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

- [src/CglMixedIntegerRounding2/CglMixedIntegerRounding2.hpp](#)

## 6.26 CglMixIntRoundVUB Class Reference

```
#include <CglMixedIntegerRounding.hpp>
```

### Public Member Functions

- [CglMixIntRoundVUB \(\)](#)
- [CglMixIntRoundVUB \(const \[CglMixIntRoundVUB\]\(#\) &source\)](#)
- [CglMixIntRoundVUB & operator= \(const \[CglMixIntRoundVUB\]\(#\) &rhs\)](#)
- [~CglMixIntRoundVUB \(\)](#)

- int `getVar` () const
- double `getVal` () const
- void `setVar` (const int v)
- void `setVal` (const double v)

#### Protected Attributes

- int `var_`
- double `val_`

#### 6.26.1 Detailed Description

Definition at line 32 of file CgIMixedIntegerRounding.hpp.

#### 6.26.2 Constructor & Destructor Documentation

##### 6.26.2.1 CgIMixIntRoundVUB::CgIMixIntRoundVUB ( ) [inline]

Definition at line 43 of file CgIMixedIntegerRounding.hpp.

##### 6.26.2.2 CgIMixIntRoundVUB::CgIMixIntRoundVUB ( const CgIMixIntRoundVUB & source ) [inline]

Definition at line 46 of file CgIMixedIntegerRounding.hpp.

##### 6.26.2.3 CgIMixIntRoundVUB::~~CgIMixIntRoundVUB ( ) [inline]

Definition at line 61 of file CgIMixedIntegerRounding.hpp.

#### 6.26.3 Member Function Documentation

##### 6.26.3.1 CgIMixIntRoundVUB& CgIMixIntRoundVUB::operator= ( const CgIMixIntRoundVUB & rhs ) [inline]

Definition at line 52 of file CgIMixedIntegerRounding.hpp.

##### 6.26.3.2 int CgIMixIntRoundVUB::getVar ( ) const [inline]

Definition at line 64 of file CgIMixedIntegerRounding.hpp.

##### 6.26.3.3 double CgIMixIntRoundVUB::getVal ( ) const [inline]

Definition at line 65 of file CgIMixedIntegerRounding.hpp.

##### 6.26.3.4 void CgIMixIntRoundVUB::setVar ( const int v ) [inline]

Definition at line 66 of file CgIMixedIntegerRounding.hpp.

##### 6.26.3.5 void CgIMixIntRoundVUB::setVal ( const double v ) [inline]

Definition at line 67 of file CgIMixedIntegerRounding.hpp.

#### 6.26.4 Member Data Documentation

#### 6.26.4.1 `int CglMixIntRoundVUB::var_ [protected]`

Definition at line 38 of file `CglMixedIntegerRounding.hpp`.

#### 6.26.4.2 `double CglMixIntRoundVUB::val_ [protected]`

Definition at line 39 of file `CglMixedIntegerRounding.hpp`.

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

- `src/CglMixedIntegerRounding/CglMixedIntegerRounding.hpp`

## 6.27 CglMixIntRoundVUB2 Class Reference

```
#include <CglMixedIntegerRounding2.hpp>
```

### Public Member Functions

- `CglMixIntRoundVUB2 ()`
- `CglMixIntRoundVUB2 (const CglMixIntRoundVUB2 &source)`
- `CglMixIntRoundVUB2 & operator= (const CglMixIntRoundVUB2 &rhs)`
- `~CglMixIntRoundVUB2 ()`
- `int getVar () const`
- `double getVal () const`
- `void setVar (const int v)`
- `void setVal (const double v)`

### Protected Attributes

- `int var_`
- `double val_`

#### 6.27.1 Detailed Description

Definition at line 33 of file `CglMixedIntegerRounding2.hpp`.

#### 6.27.2 Constructor & Destructor Documentation

##### 6.27.2.1 `CglMixIntRoundVUB2::CglMixIntRoundVUB2 ( ) [inline]`

Definition at line 44 of file `CglMixedIntegerRounding2.hpp`.

##### 6.27.2.2 `CglMixIntRoundVUB2::CglMixIntRoundVUB2 ( const CglMixIntRoundVUB2 & source ) [inline]`

Definition at line 47 of file `CglMixedIntegerRounding2.hpp`.

##### 6.27.2.3 `CglMixIntRoundVUB2::~~CglMixIntRoundVUB2 ( ) [inline]`

Definition at line 62 of file `CglMixedIntegerRounding2.hpp`.

### 6.27.3 Member Function Documentation

#### 6.27.3.1 CglMixIntRoundVUB2 & CglMixIntRoundVUB2::operator= ( const CglMixIntRoundVUB2 & rhs ) [inline]

Definition at line 53 of file CglMixedIntegerRounding2.hpp.

#### 6.27.3.2 int CglMixIntRoundVUB2::getVar ( ) const [inline]

Definition at line 65 of file CglMixedIntegerRounding2.hpp.

#### 6.27.3.3 double CglMixIntRoundVUB2::getVal ( ) const [inline]

Definition at line 66 of file CglMixedIntegerRounding2.hpp.

#### 6.27.3.4 void CglMixIntRoundVUB2::setVar ( const int v ) [inline]

Definition at line 67 of file CglMixedIntegerRounding2.hpp.

#### 6.27.3.5 void CglMixIntRoundVUB2::setVal ( const double v ) [inline]

Definition at line 68 of file CglMixedIntegerRounding2.hpp.

### 6.27.4 Member Data Documentation

#### 6.27.4.1 int CglMixIntRoundVUB2::var\_ [protected]

Definition at line 39 of file CglMixedIntegerRounding2.hpp.

#### 6.27.4.2 double CglMixIntRoundVUB2::val\_ [protected]

Definition at line 40 of file CglMixedIntegerRounding2.hpp.

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

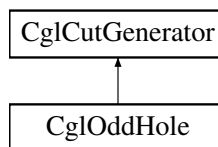
- src/CglMixedIntegerRounding2/CglMixedIntegerRounding2.hpp

## 6.28 CglOddHole Class Reference

Odd Hole Cut Generator Class.

```
#include <CglOddHole.hpp>
```

Inheritance diagram for CglOddHole:



### Public Member Functions

#### Generate Cuts

- virtual void [generateCuts](#) (const OsiSolverInterface &si, OsiCuts &cs, const [CglTreeInfo](#) info=[CglTreeInfo](#)())

*Generate odd hole cuts for the model of the solver interface, si.*

### Create Row List

- void `createRowList` (const OsiSolverInterface &si, const int \*possible=NULL)  
*Create a list of rows which might yield cuts this is to speed up process The possible parameter is a list to cut down search.*
- void `createRowList` (int numberOfRows, const int \*whichRow)  
*This version passes in a list - 1 marks possible.*

### Create Clique List

- void `createCliqueList` (int numberCliques, const int \*cliqueStart, const int \*cliqueMember)  
*Create a list of extra row cliques which may not be in matrix At present these are classical cliques.*

### Number Possibilities

- int `numberPossible` ()  
*Returns how many rows might give odd hole cuts.*

### Gets and Sets

- double `getMinimumViolation` () const  
*Minimum violation.*
- void `setMinimumViolation` (double value)
- double `getMinimumViolationPer` () const  
*Minimum violation per entry.*
- void `setMinimumViolationPer` (double value)
- int `getMaximumEntries` () const  
*Maximum number of entries in a cut.*
- void `setMaximumEntries` (int value)

### Constructors and destructors

- `CglOddHole` ()  
*Default constructor.*
- `CglOddHole` (const `CglOddHole` &)  
*Copy constructor.*
- virtual `CglCutGenerator` \* `clone` () const  
*Clone.*
- `CglOddHole` & `operator=` (const `CglOddHole` &rhs)  
*Assignment operator.*
- virtual `~CglOddHole` ()  
*Destructor.*
- virtual void `refreshSolver` (OsiSolverInterface \*solver)  
*This can be used to refresh any inforamtion.*

### Friends

- void `CglOddHoleUnitTest` (const OsiSolverInterface \*siP, const std::string mpdDir)  
*A function that tests the methods in the `CglOddHole` class.*

## Additional Inherited Members

## 6.28.1 Detailed Description

Odd Hole Cut Generator Class.

Definition at line 14 of file CglOddHole.hpp.

## 6.28.2 Constructor &amp; Destructor Documentation

## 6.28.2.1 CglOddHole::CglOddHole ( )

Default constructor.

## 6.28.2.2 CglOddHole::CglOddHole ( const CglOddHole &amp; )

Copy constructor.

## 6.28.2.3 virtual CglOddHole::~CglOddHole ( ) [virtual]

Destructor.

## 6.28.3 Member Function Documentation

6.28.3.1 virtual void CglOddHole::generateCuts ( const OsiSolverInterface & *si*, OsiCuts & *cs*, const CglTreeInfo *info* = CglTreeInfo ( ) ) [virtual]

Generate odd hole cuts for the model of the solver interface, *si*.

This looks at all rows of type  $\sum x(i) \leq 1$  (or  $= 1$ ) ( $x$  0-1) and sees if there is an odd cycle cut. See Grotschel, Lovasz and Schrijver (1988) for method. This is then lifted by using the corresponding Chvatal cut i.e. Take all rows in cycle and add them together. RHS will be odd so weaken all odd coefficients so 1.0 goes to 0.0 etc - then constraint is  $\sum \text{even}(j) * x(j) \leq \text{odd}$  which can be replaced by  $\sum (\text{even}(j)/2) * x(j) \leq (\text{odd}-1.0)/2$ . A similar cut can be generated for  $\sum x(i) \geq 1$ .

Insert the generated cuts into OsiCut, *cs*.

This is only done for rows with unsatisfied 0-1 variables. If there are many of these it will be slow. Improvements would do a randomized subset and also speed up shortest path algorithm used.

Implements [CglCutGenerator](#).

6.28.3.2 void CglOddHole::createRowList ( const OsiSolverInterface & *si*, const int \* *possible* = NULL )

Create a list of rows which might yield cuts this is to speed up process The possible parameter is a list to cut down search.

6.28.3.3 void CglOddHole::createRowList ( int *numberOfRows*, const int \* *whichRow* )

This version passes in a list - 1 marks possible.

6.28.3.4 void CglOddHole::createCliqueList ( int *numberOfCliques*, const int \* *cliqueStart*, const int \* *cliqueMember* )

Create a list of extra row cliques which may not be in matrix At present these are classical cliques.

6.28.3.5 `int CglOddHole::numberPossible ( )`

Returns how many rows might give odd hole cuts.

6.28.3.6 `double CglOddHole::getMinimumViolation ( ) const`

Minimum violation.

6.28.3.7 `void CglOddHole::setMinimumViolation ( double value )`

6.28.3.8 `double CglOddHole::getMinimumViolationPer ( ) const`

Minimum violation per entry.

6.28.3.9 `void CglOddHole::setMinimumViolationPer ( double value )`

6.28.3.10 `int CglOddHole::getMaximumEntries ( ) const`

Maximum number of entries in a cut.

6.28.3.11 `void CglOddHole::setMaximumEntries ( int value )`

6.28.3.12 `virtual CglCutGenerator* CglOddHole::clone ( ) const` [virtual]

Clone.

Implements [CglCutGenerator](#).

6.28.3.13 `CglOddHole& CglOddHole::operator= ( const CglOddHole & rhs )`

Assignment operator.

6.28.3.14 `virtual void CglOddHole::refreshSolver ( OsiSolverInterface * solver )` [virtual]

This can be used to refresh any information.

Reimplemented from [CglCutGenerator](#).

## 6.28.4 Friends And Related Function Documentation

6.28.4.1 `void CglOddHoleUnitTest ( const OsiSolverInterface * siP, const std::string mpdDir )` [friend]

A function that tests the methods in the [CglOddHole](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

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

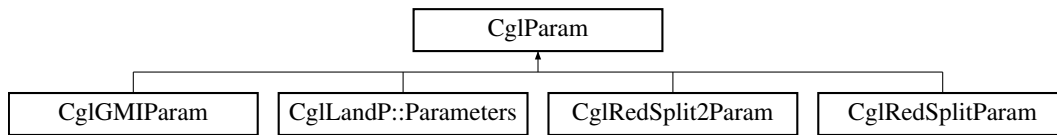
- `src/CglOddHole/CglOddHole.hpp`

## 6.29 CglParam Class Reference

Class collecting parameters for all cut generators.

```
#include <CglParam.hpp>
```

Inheritance diagram for CglParam:



## Public Member Functions

### Public Set/get methods

- virtual void **setINFINIT** (const double inf)  
*Set INFINIT.*
- double **getINFINIT** () const  
*Get value of INFINIT.*
- virtual void **setEPS** (const double eps)  
*Set EPS.*
- double **getEPS** () const  
*Get value of EPS.*
- virtual void **setEPS\_COEFF** (const double eps\_c)  
*Set EPS\_COEFF.*
- double **getEPS\_COEFF** () const  
*Get value of EPS\_COEFF.*
- virtual void **setMax\_SUPPORT** (const int max\_s)  
*Set MAX\_SUPPORT.*
- int **getMAX\_SUPPORT** () const  
*Get value of MAX\_SUPPORT.*

### Constructors and destructors

- **CglParam** (const double inf=COIN\_DBL\_MAX, const double eps=1e-6, const double eps\_c=1e-5, const int max\_s=COIN\_INT\_MAX)  
*Default constructor.*
- **CglParam** (const **CglParam** &)  
*Copy constructor.*
- virtual **CglParam** \* **clone** () const  
*Clone.*
- **CglParam** & **operator=** (const **CglParam** &rhs)  
*Assignment operator.*
- virtual **~CglParam** ()  
*Destructor.*

## Protected Attributes

### Protected member data

- double **INFINIT**
- double **EPS**
- double **EPS\_COEFF**
- int **MAX\_SUPPORT**  
*Maximum number of non zero coefficients in a generated cut; Default: COIN\_INT\_MAX.*



### 6.29.1 Detailed Description

Class collecting parameters for all cut generators.

Each generator may have a derived class to add parameters. Each generator might also set different default values for the parameters in [CglParam](#).

Definition at line 22 of file CglParam.hpp.

### 6.29.2 Constructor & Destructor Documentation

**6.29.2.1** `CglParam::CglParam ( const double inf = COIN_DBL_MAX, const double eps = 1e-6, const double eps_c = 1e-5, const int max_s = COIN_INT_MAX )`

Default constructor.

**6.29.2.2** `CglParam::CglParam ( const CglParam & )`

Copy constructor.

**6.29.2.3** `virtual CglParam::~~CglParam ( ) [virtual]`

Destructor.

### 6.29.3 Member Function Documentation

**6.29.3.1** `virtual void CglParam::setINFINIT ( const double inf ) [virtual]`

Set INFINIT.

**6.29.3.2** `double CglParam::getINFINIT ( ) const [inline]`

Get value of INFINIT.

Definition at line 32 of file CglParam.hpp.

**6.29.3.3** `virtual void CglParam::setEPS ( const double eps ) [virtual]`

Set EPS.

**6.29.3.4** `double CglParam::getEPS ( ) const [inline]`

Get value of EPS.

Definition at line 37 of file CglParam.hpp.

**6.29.3.5** `virtual void CglParam::setEPS_COEFF ( const double eps_c ) [virtual]`

Set EPS\_COEFF.

**6.29.3.6** `double CglParam::getEPS_COEFF ( ) const [inline]`

Get value of EPS\_COEFF.

Definition at line 42 of file CglParam.hpp.

6.29.3.7 `virtual void CglParam::setMax_SUPPORT ( const int max_s ) [virtual]`

Set MAX\_SUPPORT.

6.29.3.8 `int CglParam::getMAX_SUPPORT ( ) const [inline]`

Get value of MAX\_SUPPORT.

Definition at line 47 of file CglParam.hpp.

6.29.3.9 `virtual CglParam* CglParam::clone ( ) const [virtual]`

Clone.

Reimplemented in [CglRedSplit2Param](#), [CglGMIParam](#), and [CglRedSplitParam](#).

6.29.3.10 `CglParam& CglParam::operator= ( const CglParam & rhs )`

Assignment operator.

#### 6.29.4 Member Data Documentation

6.29.4.1 `double CglParam::INFINIT [protected]`

Definition at line 77 of file CglParam.hpp.

6.29.4.2 `double CglParam::EPS [protected]`

Definition at line 80 of file CglParam.hpp.

6.29.4.3 `double CglParam::EPS_COEFF [protected]`

Definition at line 84 of file CglParam.hpp.

6.29.4.4 `int CglParam::MAX_SUPPORT [protected]`

Maximum number of non zero coefficients in a generated cut; Default: COIN\_INT\_MAX.

Definition at line 88 of file CglParam.hpp.

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

- [src/CglParam.hpp](#)

## 6.30 CglPreProcess Class Reference

Class for preProcessing and postProcessing.

```
#include <CglPreProcess.hpp>
```

#### Public Member Functions

##### Main methods

- `OsiSolverInterface * preProcess (OsiSolverInterface &model, bool makeEquality=false, int numberPasses=5)`  
*preProcess problem - returning new problem.*

- OsiSolverInterface \* [preProcessNonDefault](#) (OsiSolverInterface &model, int makeEquality=0, int number-Passes=5, int tuning=0)  
*preProcess problem - returning new problem.*
- void [postProcess](#) (OsiSolverInterface &model, bool deleteStuff=true)  
*Creates solution in original model.*
- int [tightenPrimalBounds](#) (OsiSolverInterface &model, double factor=0.0)  
*Tightens primal bounds to make dual and branch and cutfaster.*
- OsiSolverInterface \* [someFixed](#) (OsiSolverInterface &model, double fractionToKeep=0.25, bool fixContinuous-AsWell=false, char \*keep=NULL) const  
*Fix some of problem - returning new problem.*
- OsiSolverInterface \* [cliquelt](#) (OsiSolverInterface &model, double cliquesNeeded=0.0) const  
*Replace cliques by more maximal cliques Returns NULL if rows not reduced by greater than cliquesNeeded\*rows.*
- int [reducedCostFix](#) (OsiSolverInterface &model)  
*If we have a cutoff - fix variables.*

### Parameter set/get methods

The set methods return true if the parameter was set to the given value, false if the value of the parameter is out of range.

The get methods return the value of the parameter.

- void [setCutoff](#) (double value)  
*Set cutoff bound on the objective function.*
- double [getCutoff](#) () const  
*Get the cutoff bound on the objective function - always as minimize.*
- OsiSolverInterface \* [originalModel](#) () const  
*The original solver associated with this model.*
- OsiSolverInterface \* [startModel](#) () const  
*Solver after making clique equalities (may == original)*
- OsiSolverInterface \* [modelAtPass](#) (int iPass) const  
*Copies of solver at various stages after presolve.*
- OsiSolverInterface \* [modifiedModel](#) (int iPass) const  
*Copies of solver at various stages after presolve after modifications.*
- OsiPresolve \* [presolve](#) (int iPass) const  
*Matching presolve information.*
- const int \* [originalColumns](#) ()  
*Return a pointer to the original columns (with possible clique slacks) MUST be called before postProcess otherwise you just get 0,1,2.*
- const int \* [originalRows](#) ()  
*Return a pointer to the original rows MUST be called before postProcess otherwise you just get 0,1,2.*
- int [numberSOS](#) () const  
*Number of SOS if found.*
- const int \* [typeSOS](#) () const  
*Type of each SOS.*
- const int \* [startSOS](#) () const  
*Start of each SOS.*
- const int \* [whichSOS](#) () const  
*Columns in SOS.*
- const double \* [weightSOS](#) () const  
*Weights for each SOS column.*
- void [passInProhibited](#) (const char \*prohibited, int numberColumns)  
*Pass in prohibited columns.*
- const char \* [prohibited](#) ()  
*Updated prohibited columns.*
- int [numberIterationsPre](#) () const

- *Number of iterations PreProcessing.*
- int `numberIterationsPost` () const  
*Number of iterations PostProcessing.*
- void `passInRowTypes` (const char \*`rowTypes`, int numberRows)  
*Pass in row types 0 normal 1 cut rows - will be dropped if remain in At end of preprocess cut rows will be dropped and put into cuts.*
- const char \* `rowTypes` ()  
*Updated row types - may be NULL Carried around and corresponds to existing rows -1 added by preprocess e.g.*
- const `CglStored` & `cuts` () const  
*Return cuts from dropped rows.*
- const `CglStored` \* `cutsPointer` () const  
*Return pointer to cuts from dropped rows.*
- void `update` (const OsiPresolve \*`pinfo`, const OsiSolverInterface \*`solver`)  
*Update prohibited and rowType.*
- void `setOptions` (int value)  
*Set options.*

### Cut generator methods

- int `numberCutGenerators` () const  
*Get the number of cut generators.*
- `CglCutGenerator` \*\* `cutGenerators` () const  
*Get the list of cut generators.*
- `CglCutGenerator` \* `cutGenerator` (int i) const  
*Get the specified cut generator.*
- void `addCutGenerator` (`CglCutGenerator` \*`generator`)  
*Add one generator - up to user to delete generators.*

### Setting/Accessing application data

- void `setApplicationData` (void \*`appData`)  
*Set application data.*
- void \* `getApplicationData` () const  
*Get application data.*

### Message handling

- void `passInMessageHandler` (CoinMessageHandler \*`handler`)  
*Pass in Message handler (not deleted at end)*
- void `newLanguage` (CoinMessages::Language language)  
*Set language.*
- void `setLanguage` (CoinMessages::Language language)
- CoinMessageHandler \* `messageHandler` () const  
*Return handler.*
- CoinMessages `messages` ()  
*Return messages.*
- CoinMessages \* `messagesPointer` ()  
*Return pointer to messages.*

### Constructors and destructors etc

- `CglPreProcess` ()  
*Constructor.*
- `CglPreProcess` (const `CglPreProcess` &`rhs`)

- Copy constructor .*
- `CglPreProcess & operator= (const CglPreProcess &rhs)`
- Assignment operator.*
- `~CglPreProcess ()`
- Destructor.*
- `void gutsOfDestructor ()`
- Clears out as much as possible.*

### 6.30.1 Detailed Description

Class for preProcessing and postProcessing.

While cuts can be added at any time in the tree, some cuts are actually just stronger versions of existing constraints. In this case they can replace those constraints rather than being added as new constraints. This is awkward in the tree but reasonable at the root node.

This is a general process class which uses other cut generators to strengthen constraints, establish that constraints are redundant, fix variables and find relationships such as  $x + y == 1$ .

Presolve will also be done.

If row names existed they may be replaced by R0000000 etc

Definition at line 36 of file CglPreProcess.hpp.

### 6.30.2 Constructor & Destructor Documentation

#### 6.30.2.1 CglPreProcess::CglPreProcess ( )

Constructor.

#### 6.30.2.2 CglPreProcess::CglPreProcess ( const CglPreProcess & rhs )

Copy constructor .

#### 6.30.2.3 CglPreProcess::~~CglPreProcess ( )

Destructor.

### 6.30.3 Member Function Documentation

#### 6.30.3.1 OsiSolverInterface\* CglPreProcess::preProcess ( OsiSolverInterface & model, bool makeEquality = false, int numberPasses = 5 )

preProcess problem - returning new problem.

If makeEquality true then  $\leq$  cliques converted to  $==$ . Presolve will be done numberPasses times.

Returns NULL if infeasible

This version uses default strategy. For more control copy and edit code from this function i.e. call preProcessNonDefault

#### 6.30.3.2 OsiSolverInterface\* CglPreProcess::preProcessNonDefault ( OsiSolverInterface & model, int makeEquality = 0, int numberPasses = 5, int tuning = 0 )

preProcess problem - returning new problem.

If makeEquality true then  $\leq$  cliques converted to  $=$ . Presolve will be done numberPasses times.

Returns NULL if infeasible

This version assumes user has added cut generators to [CglPreProcess](#) object before calling it. As an example use coding in preProcess If makeEquality is 1 add slacks to get cliques, if 2 add slacks to get sos (but only if looks plausible) and keep sos info

**6.30.3.3** `void CglPreProcess::postProcess ( OsiSolverInterface & model, bool deleteStuff=true )`

Creates solution in original model.

**6.30.3.4** `int CglPreProcess::tightenPrimalBounds ( OsiSolverInterface & model, double factor=0.0 )`

Tightens primal bounds to make dual and branch and cut faster.

Unless fixed or integral, bounds are slightly looser than they could be. Returns non-zero if problem infeasible Fudge for branch and bound - put bounds on columns of factor \* largest value (at continuous) - should improve stability in branch and bound on infeasible branches (0.0 is off)

**6.30.3.5** `OsiSolverInterface* CglPreProcess::someFixed ( OsiSolverInterface & model, double fractionToKeep=0.25, bool fixContinuousAsWell=false, char * keep=NULL ) const`

Fix some of problem - returning new problem.

Uses reduced costs. Optional signed character array 1 always keep, -1 always discard, 0 use djs

**6.30.3.6** `OsiSolverInterface* CglPreProcess::cliquest ( OsiSolverInterface & model, double cliquesNeeded=0.0 ) const`

Replace cliques by more maximal cliques Returns NULL if rows not reduced by greater than cliquesNeeded\*rows.

**6.30.3.7** `int CglPreProcess::reducedCostFix ( OsiSolverInterface & model )`

If we have a cutoff - fix variables.

**6.30.3.8** `void CglPreProcess::setCutoff ( double value )`

Set cutoff bound on the objective function.

When using strict comparison, the bound is adjusted by a tolerance to avoid accidentally cutting off the optimal solution.

**6.30.3.9** `double CglPreProcess::getCutoff ( ) const`

Get the cutoff bound on the objective function - always as minimize.

**6.30.3.10** `OsiSolverInterface* CglPreProcess::originalModel ( ) const [inline]`

The original solver associated with this model.

Definition at line 119 of file CglPreProcess.hpp.

**6.30.3.11** `OsiSolverInterface* CglPreProcess::startModel ( ) const [inline]`

Solver after making clique equalities (may == original)

Definition at line 122 of file CglPreProcess.hpp.

6.30.3.12 `OsiSolverInterface* CglPreProcess::modelAtPass ( int iPass ) const` `[inline]`

Copies of solver at various stages after presolve.

Definition at line 125 of file `CglPreProcess.hpp`.

6.30.3.13 `OsiSolverInterface* CglPreProcess::modifiedModel ( int iPass ) const` `[inline]`

Copies of solver at various stages after presolve after modifications.

Definition at line 128 of file `CglPreProcess.hpp`.

6.30.3.14 `OsiPresolve* CglPreProcess::presolve ( int iPass ) const` `[inline]`

Matching presolve information.

Definition at line 131 of file `CglPreProcess.hpp`.

6.30.3.15 `const int* CglPreProcess::originalColumns ( )`

Return a pointer to the original columns (with possible clique slacks) MUST be called before `postProcess` otherwise you just get 0,1,2.

6.30.3.16 `const int* CglPreProcess::originalRows ( )`

Return a pointer to the original rows MUST be called before `postProcess` otherwise you just get 0,1,2.

6.30.3.17 `int CglPreProcess::numberSOS ( ) const` `[inline]`

Number of SOS if found.

Definition at line 140 of file `CglPreProcess.hpp`.

6.30.3.18 `const int* CglPreProcess::typeSOS ( ) const` `[inline]`

Type of each SOS.

Definition at line 143 of file `CglPreProcess.hpp`.

6.30.3.19 `const int* CglPreProcess::startSOS ( ) const` `[inline]`

Start of each SOS.

Definition at line 146 of file `CglPreProcess.hpp`.

6.30.3.20 `const int* CglPreProcess::whichSOS ( ) const` `[inline]`

Columns in SOS.

Definition at line 149 of file `CglPreProcess.hpp`.

6.30.3.21 `const double* CglPreProcess::weightSOS ( ) const` `[inline]`

Weights for each SOS column.

Definition at line 152 of file `CglPreProcess.hpp`.

6.30.3.22 `void CglPreProcess::passInProhibited ( const char * prohibited, int numberColumns )`

Pass in prohibited columns.

**6.30.3.23** `const char* CglPreProcess::prohibited ( ) [inline]`

Updated prohibited columns.

Definition at line 157 of file CglPreProcess.hpp.

**6.30.3.24** `int CglPreProcess::numberIterationsPre ( ) const [inline]`

Number of iterations PreProcessing.

Definition at line 160 of file CglPreProcess.hpp.

**6.30.3.25** `int CglPreProcess::numberIterationsPost ( ) const [inline]`

Number of iterations PostProcessing.

Definition at line 163 of file CglPreProcess.hpp.

**6.30.3.26** `void CglPreProcess::passInRowTypes ( const char * rowTypes, int numberRows )`

Pass in row types 0 normal 1 cut rows - will be dropped if remain in At end of preprocess cut rows will be dropped and put into cuts.

**6.30.3.27** `const char* CglPreProcess::rowTypes ( ) [inline]`

Updated row types - may be NULL Carried around and corresponds to existing rows -1 added by preprocess e.g.

x+y=1 0 normal 1 cut rows - can be dropped if wanted

Definition at line 178 of file CglPreProcess.hpp.

**6.30.3.28** `const CglStored& CglPreProcess::cuts ( ) const [inline]`

Return cuts from dropped rows.

Definition at line 181 of file CglPreProcess.hpp.

**6.30.3.29** `const CglStored* CglPreProcess::cutsPointer ( ) const [inline]`

Return pointer to cuts from dropped rows.

Definition at line 184 of file CglPreProcess.hpp.

**6.30.3.30** `void CglPreProcess::update ( const OsiPresolve * pinfo, const OsiSolverInterface * solver )`

Update prohibited and rowType.

**6.30.3.31** `void CglPreProcess::setOptions ( int value ) [inline]`

Set options.

Definition at line 189 of file CglPreProcess.hpp.

**6.30.3.32** `int CglPreProcess::numberCutGenerators ( ) const [inline]`

Get the number of cut generators.

Definition at line 196 of file CglPreProcess.hpp.

**6.30.3.33** `CglCutGenerator** CglPreProcess::cutGenerators ( ) const [inline]`

Get the list of cut generators.



Definition at line 199 of file CglPreProcess.hpp.

**6.30.3.34 CglCutGenerator\* CglPreProcess::cutGenerator ( int *i* ) const** [inline]

Get the specified cut generator.

Definition at line 202 of file CglPreProcess.hpp.

**6.30.3.35 void CglPreProcess::addCutGenerator ( CglCutGenerator \* *generator* )**

Add one generator - up to user to delete generators.

**6.30.3.36 void CglPreProcess::setApplicationData ( void \* *appData* )**

Set application data.

This is a pointer that the application can store into and retrieve. This field is available for the application to optionally define and use.

**6.30.3.37 void\* CglPreProcess::getApplicationData ( ) const**

Get application data.

**6.30.3.38 void CglPreProcess::passInMessageHandler ( CoinMessageHandler \* *handler* )**

Pass in Message handler (not deleted at end)

**6.30.3.39 void CglPreProcess::newLanguage ( CoinMessages::Language *language* )**

Set language.

**6.30.3.40 void CglPreProcess::setLanguage ( CoinMessages::Language *language* )** [inline]

Definition at line 232 of file CglPreProcess.hpp.

**6.30.3.41 CoinMessageHandler\* CglPreProcess::messageHandler ( ) const** [inline]

Return handler.

Definition at line 235 of file CglPreProcess.hpp.

**6.30.3.42 CoinMessages CglPreProcess::messages ( )** [inline]

Return messages.

Definition at line 238 of file CglPreProcess.hpp.

**6.30.3.43 CoinMessages\* CglPreProcess::messagesPointer ( )** [inline]

Return pointer to messages.

Definition at line 241 of file CglPreProcess.hpp.

**6.30.3.44 CglPreProcess& CglPreProcess::operator= ( const CglPreProcess & *rhs* )**

Assignment operator.

**6.30.3.45 void CglPreProcess::gutsOfDestructor ( )**

Clears out as much as possible.

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

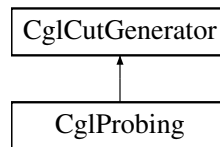
- [src/CglPreProcess/CglPreProcess.hpp](#)

## 6.31 CglProbing Class Reference

Probing Cut Generator Class.

```
#include <CglProbing.hpp>
```

Inheritance diagram for CglProbing:



### Public Member Functions

#### Generate Cuts

- virtual void [generateCuts](#) (const OsiSolverInterface &si, OsiCuts &cs, const [CglTreeInfo](#) info=[CglTreeInfo](#)())  
*Generate probing/disaggregation cuts for the model of the solver interface, si.*
- int [generateCutsAndModify](#) (const OsiSolverInterface &si, OsiCuts &cs, [CglTreeInfo](#) \*info)

#### snapshot etc

- int [snapshot](#) (const OsiSolverInterface &si, char \*possible=NULL, bool withObjective=true)  
*Create a copy of matrix which is to be used this is to speed up process and to give global cuts Can give an array with 1 set to select, 0 to ignore column bounds are tightened If array given then values of 1 will be set to 0 if redundant.*
- void [deleteSnapshot](#) ()  
*Deletes snapshot.*
- int [createCliques](#) (OsiSolverInterface &si, int minimumSize=2, int maximumSize=100)  
*Creates cliques for use by probing.*
- void [deleteCliques](#) ()  
*Delete all clique information.*

#### Get tighter column bounds

- const double \* [tightLower](#) () const  
*Lower.*
- const double \* [tightUpper](#) () const  
*Upper.*
- const char \* [tightenBounds](#) () const  
*Array which says tighten continuous.*

#### Get possible freed up row bounds - only valid after mode==3

- const double \* [relaxedRowLower](#) () const  
*Lower.*
- const double \* [relaxedRowUpper](#) () const  
*Upper.*

### Change mode

- void `setMode` (int mode)  
*Set.*
- int `getMode` () const  
*Get.*

### Change maxima

- void `setMaxPass` (int value)  
*Set maximum number of passes per node.*
- int `getMaxPass` () const  
*Get maximum number of passes per node.*
- void `setLogLevel` (int value)  
*Set log level - 0 none, 1 - a bit, 2 - more details.*
- int `getLogLevel` () const  
*Get log level.*
- void `setMaxProbe` (int value)  
*Set maximum number of unsatisfied variables to look at.*
- int `getMaxProbe` () const  
*Get maximum number of unsatisfied variables to look at.*
- void `setMaxLook` (int value)  
*Set maximum number of variables to look at in one probe.*
- int `getMaxLook` () const  
*Get maximum number of variables to look at in one probe.*
- void `setMaxElements` (int value)  
*Set maximum number of elements in row for it to be considered.*
- int `getMaxElements` () const  
*Get maximum number of elements in row for it to be considered.*
- void `setMaxPassRoot` (int value)  
*Set maximum number of passes per node (root node)*
- int `getMaxPassRoot` () const  
*Get maximum number of passes per node (root node)*
- void `setMaxProbeRoot` (int value)  
*Set maximum number of unsatisfied variables to look at (root node)*
- int `getMaxProbeRoot` () const  
*Get maximum number of unsatisfied variables to look at (root node)*
- void `setMaxLookRoot` (int value)  
*Set maximum number of variables to look at in one probe (root node)*
- int `getMaxLookRoot` () const  
*Get maximum number of variables to look at in one probe (root node)*
- void `setMaxElementsRoot` (int value)  
*Set maximum number of elements in row for it to be considered (root node)*
- int `getMaxElementsRoot` () const  
*Get maximum number of elements in row for it to be considered (root node)*
- virtual bool `mayGenerateRowCutsInTree` () const  
*Returns true if may generate Row cuts in tree (rather than root node).*

### Get information back from probing

- int `numberThisTime` () const  
*Number looked at this time.*
- const int \* `lookedAt` () const  
*Which ones looked at this time.*

**Stop or restart row cuts (otherwise just fixing from probing)**

- void [setRowCuts](#) (int type)  
*Set 0 no cuts, 1 just disaggregation type, 2 coefficient ( 3 both)*
- int [rowCuts](#) () const  
*Get.*

**Whether use objective as constraint**

- void [setUsingObjective](#) (int yesNo)  
*Set 0 don't 1 do -1 don't even think about it.*
- int [getUsingObjective](#) () const  
*Get.*

**Mark which continuous variables are to be tightened**

- void [tightenThese](#) (const OsiSolverInterface &solver, int number, const int \*which)  
*Mark variables to be tightened.*

**Constructors and destructors**

- [CglProbing](#) ()  
*Default constructor.*
- [CglProbing](#) (const [CglProbing](#) &)  
*Copy constructor.*
- virtual [CglCutGenerator](#) \* [clone](#) () const  
*Clone.*
- [CglProbing](#) & [operator=](#) (const [CglProbing](#) &rhs)  
*Assignment operator.*
- virtual [~CglProbing](#) ()  
*Destructor.*
- virtual void [refreshSolver](#) (OsiSolverInterface \*solver)  
*This can be used to refresh any informantion.*
- virtual std::string [generateCpp](#) (FILE \*fp)  
*Create C++ lines to get to current state.*

**Friends**

- struct [CglProbing::disaggregation\\_struct\\_tag](#)
- void [CglProbingUnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)  
*A function that tests the methods in the [CglProbing](#) class.*

**Additional Inherited Members****6.31.1 Detailed Description**

Probing Cut Generator Class.

Definition at line 25 of file [CglProbing.hpp](#).

### 6.31.2 Constructor & Destructor Documentation

#### 6.31.2.1 CglProbing::CglProbing ( )

Default constructor.

#### 6.31.2.2 CglProbing::CglProbing ( const CglProbing & )

Copy constructor.

#### 6.31.2.3 virtual CglProbing::~~CglProbing ( ) [virtual]

Destructor.

### 6.31.3 Member Function Documentation

#### 6.31.3.1 virtual void CglProbing::generateCuts ( const OsiSolverInterface & si, OsiCuts & cs, const CglTreeInfo info = CglTreeInfo ( ) ) [virtual]

Generate probing/disaggregation cuts for the model of the solver interface, si.

This is a simplification of probing ideas put into OSL about ten years ago. The only known documentation is a copy of a talk handout - we think Robin Lougee-Heimer has a copy!

For selected integer variables (e.g. unsatisfied ones) the effect of setting them up or down is investigated. Setting a variable up may in turn set other variables (continuous as well as integer). There are various possible results:

- 1) It is shown that problem is infeasible (this may also be because objective function or reduced costs show worse than best solution). If the other way is feasible we can generate a column cut (and continue probing), if not feasible we can say problem infeasible.
- 2) If both ways are feasible, it can happen that  $x$  to 0 implies  $y$  to 1 and  $x$  to 1 implies  $y$  to 1 (again a column cut). More common is that  $x$  to 0 implies  $y$  to 1 and  $x$  to 1 implies  $y$  to 0 so we could substitute for  $y$  which might lead later to more powerful cuts. This is not done in this code as there is no mechanism for returning information.
- 3) When  $x$  to 1 a constraint went slack by  $c$ . We can tighten the constraint  $ax + \dots \leq b$  (where  $a$  may be zero) to  $(a+c)x + \dots \leq b$ . If this cut is violated then it is generated.
- 4) Similarly we can generate implied disaggregation cuts

Note - differences to cuts in OSL.

- a) OSL had structures intended to make this faster.
  - b) The "chaining" in 2) was done
  - c) Row cuts modified original constraint rather than adding cut
- b) This code can cope with general integer variables.

Insert the generated cuts into OsiCut, cs.

If a "snapshot" of a matrix exists then this will be used. Presumably this will give global cuts and will be faster. No check is done to see if cuts will be global.

Otherwise use current matrix.

Both row cuts and column cuts may be returned

The mode options are: 0) Only unsatisfied integer variables will be looked at. If no information exists for that variable then probing will be done so as a by-product you "may" get a fixing or infeasibility. This will be fast and is only available if a snapshot exists (otherwise as 1). The bounds in the snapshot are the ones used. 1) Look at unsatisfied integer variables, using current bounds. Probing will be done on all looked at. 2) Look at all integer variables, using current bounds. Probing will be done on all

If generateCutsAndModify is used then new relaxed row bounds and tightened column bounds are generated Returns

number of infeasibilities

Implements [CglCutGenerator](#).

6.31.3.2 `int CglProbing::generateCutsAndModify ( const OsiSolverInterface & si, OsiCuts & cs, CglTreeInfo * info )`

6.31.3.3 `int CglProbing::snapshot ( const OsiSolverInterface & si, char * possible = NULL, bool withObjective = true )`

Create a copy of matrix which is to be used this is to speed up process and to give global cuts Can give an array with 1 set to select, 0 to ignore column bounds are tightened If array given then values of 1 will be set to 0 if redundant.

Objective may be added as constraint Returns 1 if infeasible otherwise 0

6.31.3.4 `void CglProbing::deleteSnapshot ( )`

Deletes snapshot.

6.31.3.5 `int CglProbing::createCliques ( OsiSolverInterface & si, int minimumSize = 2, int maximumSize = 100 )`

Creates cliques for use by probing.

Only cliques  $\geq$  minimumSize and  $<$  maximumSize created Can also try and extend cliques as a result of probing (root node). Returns number of cliques found.

6.31.3.6 `void CglProbing::deleteCliques ( )`

Delete all clique information.

6.31.3.7 `const double* CglProbing::tightLower ( ) const`

Lower.

6.31.3.8 `const double* CglProbing::tightUpper ( ) const`

Upper.

6.31.3.9 `const char* CglProbing::tightenBounds ( ) const` `[inline]`

Array which says tighten continuous.

Definition at line 138 of file CglProbing.hpp.

6.31.3.10 `const double* CglProbing::relaxedRowLower ( ) const`

Lower.

6.31.3.11 `const double* CglProbing::relaxedRowUpper ( ) const`

Upper.

6.31.3.12 `void CglProbing::setMode ( int mode )`

Set.

6.31.3.13 `int CglProbing::getMode ( ) const`

Get.

6.31.3.14 void CglProbing::setMaxPass ( int *value* )

Set maximum number of passes per node.

6.31.3.15 int CglProbing::getMaxPass ( ) const

Get maximum number of passes per node.

6.31.3.16 void CglProbing::setLogLevel ( int *value* )

Set log level - 0 none, 1 - a bit, 2 - more details.

6.31.3.17 int CglProbing::getLogLevel ( ) const

Get log level.

6.31.3.18 void CglProbing::setMaxProbe ( int *value* )

Set maximum number of unsatisfied variables to look at.

6.31.3.19 int CglProbing::getMaxProbe ( ) const

Get maximum number of unsatisfied variables to look at.

6.31.3.20 void CglProbing::setMaxLook ( int *value* )

Set maximum number of variables to look at in one probe.

6.31.3.21 int CglProbing::getMaxLook ( ) const

Get maximum number of variables to look at in one probe.

6.31.3.22 void CglProbing::setMaxElements ( int *value* )

Set maximum number of elements in row for it to be considered.

6.31.3.23 int CglProbing::getMaxElements ( ) const

Get maximum number of elements in row for it to be considered.

6.31.3.24 void CglProbing::setMaxPassRoot ( int *value* )

Set maximum number of passes per node (root node)

6.31.3.25 int CglProbing::getMaxPassRoot ( ) const

Get maximum number of passes per node (root node)

6.31.3.26 void CglProbing::setMaxProbeRoot ( int *value* )

Set maximum number of unsatisfied variables to look at (root node)

6.31.3.27 int CglProbing::getMaxProbeRoot ( ) const

Get maximum number of unsatisfied variables to look at (root node)

6.31.3.28 void CglProbing::setMaxLookRoot ( int *value* )

Set maximum number of variables to look at in one probe (root node)

6.31.3.29 int CglProbing::getMaxLookRoot ( ) const

Get maximum number of variables to look at in one probe (root node)

6.31.3.30 void CglProbing::setMaxElementsRoot ( int *value* )

Set maximum number of elements in row for it to be considered (root node)

6.31.3.31 int CglProbing::getMaxElementsRoot ( ) const

Get maximum number of elements in row for it to be considered (root node)

6.31.3.32 virtual bool CglProbing::mayGenerateRowCutsInTree ( ) const [virtual]

Returns true if may generate Row cuts in tree (rather than root node).

Used so know if matrix will change in tree. Really meant so column cut generators can still be active without worrying code. Default is true

Reimplemented from [CglCutGenerator](#).

6.31.3.33 int CglProbing::numberThisTime ( ) const [inline]

Number looked at this time.

Definition at line 209 of file CglProbing.hpp.

6.31.3.34 const int\* CglProbing::lookedAt ( ) const [inline]

Which ones looked at this time.

Definition at line 212 of file CglProbing.hpp.

6.31.3.35 void CglProbing::setRowCuts ( int *type* )

Set 0 no cuts, 1 just disaggregation type, 2 coefficient ( 3 both)

6.31.3.36 int CglProbing::rowCuts ( ) const

Get.

6.31.3.37 void CglProbing::setUsingObjective ( int *yesNo* )

Set 0 don't 1 do -1 don't even think about it.

6.31.3.38 int CglProbing::getUsingObjective ( ) const

Get.

6.31.3.39 void CglProbing::tightenThese ( const OsiSolverInterface & *solver*, int *number*, const int \* *which* )

Mark variables to be tightened.

6.31.3.40 virtual CglCutGenerator\* CglProbing::clone ( ) const [virtual]

Clone.



Implements [CglCutGenerator](#).

6.31.3.41 `CglProbing& CglProbing::operator= ( const CglProbing & rhs )`

Assignment operator.

6.31.3.42 `virtual void CglProbing::refreshSolver ( OsiSolverInterface * solver ) [virtual]`

This can be used to refresh any information.

Reimplemented from [CglCutGenerator](#).

6.31.3.43 `virtual std::string CglProbing::generateCpp ( FILE * fp ) [virtual]`

Create C++ lines to get to current state.

Reimplemented from [CglCutGenerator](#).

## 6.31.4 Friends And Related Function Documentation

6.31.4.1 `friend struct CglProbing::disaggregation_struct_tag [friend]`

Definition at line 332 of file `CglProbing.hpp`.

6.31.4.2 `void CglProbingUnitTest ( const OsiSolverInterface * siP, const std::string mpdDir ) [friend]`

A function that tests the methods in the [CglProbing](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

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

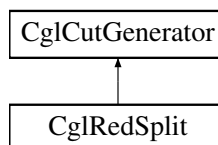
- `src/CglProbing/CglProbing.hpp`

## 6.32 CglRedSplit Class Reference

Gomory Reduce-and-Split Cut Generator Class; See method [generateCuts\(\)](#).

```
#include <CglRedSplit.hpp>
```

Inheritance diagram for `CglRedSplit`:



### Public Member Functions

#### **generateCuts**

- `virtual void generateCuts (const OsiSolverInterface &si, OsiCuts &cs, const CglTreeInfo info=CglTreeInfo())`

*Generate Reduce-and-Split Mixed Integer Gomory cuts for the model of the solver interface si.*

- virtual bool `needsOptimalBasis` () const  
*Return true if needs optimal basis to do cuts (will return true)*

### Public Methods

- void `setParam` (const `CglRedSplitParam` &source)
- `CglRedSplitParam` `getParam` () const
- void `compute_is_lub` ()
- void `compute_is_integer` ()
- void `set_given_optsol` (const double \*given\_sol, const int card\_sol)  
*Set given\_optsol to the given optimal solution given\_sol.*
- void `print` () const  
*Print some of the data members.*
- void `printOptTab` (`OsiSolverInterface` \*solver) const  
*Print the current simplex tableau.*

### Public Methods (soon to be obsolete)

- void `setLimit` (int limit)  
*Set limit, the maximum number of non zero coefficients in generated cut; Default: 50.*
- int `getLimit` () const  
*Get value of limit.*
- void `setAway` (double value)  
*Set away, the minimum distance from being integer used for selecting rows for cut generation; all rows whose pivot variable should be integer but is more than away from integrality will be selected; Default: 0.05.*
- double `getAway` () const  
*Get value of away.*
- void `setLUB` (double value)  
*Set the value of LUB, value considered large for the absolute value of a lower or upper bound on a variable; Default: 1000.*
- double `getLUB` () const  
*Get the value of LUB.*
- void `setEPS` (double value)  
*Set the value of EPS, epsilon for double computations; Default: 1e-7.*
- double `getEPS` () const  
*Get the value of EPS.*
- void `setEPS_COEFF` (double value)  
*Set the value of EPS\_COEFF, epsilon for values of coefficients; Default: 1e-8.*
- double `getEPS_COEFF` () const  
*Get the value of EPS\_COEFF.*
- void `setEPS_COEFF_LUB` (double value)  
*Set the value of EPS\_COEFF\_LUB, epsilon for values of coefficients for variables with absolute value of lower or upper bound larger than LUB; Default: 1e-13.*
- double `getEPS_COEFF_LUB` () const  
*Get the value of EPS\_COEFF\_LUB.*
- void `setEPS_RELAX` (double value)  
*Set the value of EPS\_RELAX, value used for relaxing the right hand side of each generated cut; Default: 1e-8.*
- double `getEPS_RELAX` () const  
*Get the value of EPS\_RELAX.*
- void `setNormIsZero` (double value)  
*Set the value of normIsZero, the threshold for considering a norm to be 0; Default: 1e-5.*
- double `getNormIsZero` () const  
*Get the value of normIsZero.*
- void `setMinReduc` (double value)

- *Set the value of `minReduc`, threshold for relative norm improvement for performing a reduction; Default: 0.05.*
- `double getMinReduc () const`  
*Get the value of `minReduc`.*
- `void setMaxTab (double value)`  
*Set the maximum allowed value for  $(mTab * mTab * CoinMax(mTab, nTab))$  where `mTab` is the number of rows used in the combinations and `nTab` is the number of continuous non basic variables.*
- `double getMaxTab () const`  
*Get the value of `maxTab`.*

### Constructors and destructors

- `CglRedSplit ()`  
*Default constructor.*
- `CglRedSplit (const CglRedSplitParam &RS_param)`  
*Constructor with specified parameters.*
- `CglRedSplit (const CglRedSplit &)`  
*Copy constructor.*
- `virtual CglCutGenerator * clone () const`  
*Clone.*
- `CglRedSplit & operator= (const CglRedSplit &rhs)`  
*Assignment operator.*
- `virtual ~CglRedSplit ()`  
*Destructor.*
- `virtual std::string generateCpp (FILE *fp)`  
*Create C++ lines to get to current state.*

### Friends

- `void CglRedSplitUnitTest (const OsiSolverInterface *siP, const std::string mpdDir)`  
*A function that tests the methods in the `CglRedSplit` class.*

### Additional Inherited Members

#### 6.32.1 Detailed Description

Gomory Reduce-and-Split Cut Generator Class; See method `generateCuts()`.

Based on the paper by K. Anderson, G. Cornuejols, Yanjun Li, "Reduce-and-Split Cuts: Improving the Performance of Mixed Integer Gomory Cuts", Management Science 51 (2005).

Definition at line 26 of file `CglRedSplit.hpp`.

#### 6.32.2 Constructor & Destructor Documentation

##### 6.32.2.1 `CglRedSplit::CglRedSplit ( )`

Default constructor.

##### 6.32.2.2 `CglRedSplit::CglRedSplit ( const CglRedSplitParam & RS_param )`

Constructor with specified parameters.

##### 6.32.2.3 `CglRedSplit::CglRedSplit ( const CglRedSplit & )`

Copy constructor.

6.32.2.4 `virtual CglRedSplit::~CglRedSplit ( ) [virtual]`

Destructor.

### 6.32.3 Member Function Documentation

6.32.3.1 `virtual void CglRedSplit::generateCuts ( const OsiSolverInterface & si, OsiCuts & cs, const CglTreeInfo info = CglTreeInfo ( ) ) [virtual]`

Generate Reduce-and-Split Mixed Integer Gomory cuts for the model of the solver interface *si*.

Insert the generated cuts into OsiCuts *cs*.

Warning: This generator currently works only with the Lp solvers Clp or Cplex9.0 or higher. It requires access to the optimal tableau and optimal basis inverse and makes assumptions on the way slack variables are added by the solver. The Osi implementations for Clp and Cplex verify these assumptions.

When calling the generator, the solver interface *si* must contain an optimized problem and information related to the optimal basis must be available through the OsiSolverInterface methods (*si*->optimalBasisIsAvailable() must return 'true'). It is also essential that the integrality of structural variable *i* can be obtained using *si*->isInteger(*i*).

Reduce-and-Split cuts are variants of Gomory cuts: Starting from the current optimal tableau, linear combinations of the rows of the current optimal simplex tableau are used for generating Gomory cuts. The choice of the linear combinations is driven by the objective of reducing the coefficients of the non basic continuous variables in the resulting row. Note that this generator might not be able to generate cuts for some solutions violating integrality constraints.

Implements [CglCutGenerator](#).

6.32.3.2 `virtual bool CglRedSplit::needsOptimalBasis ( ) const [virtual]`

Return true if needs optimal basis to do cuts (will return true)

Reimplemented from [CglCutGenerator](#).

6.32.3.3 `void CglRedSplit::setParam ( const CglRedSplitParam & source )`

6.32.3.4 `CglRedSplitParam CglRedSplit::getParam ( ) const [inline]`

Definition at line 75 of file CglRedSplit.hpp.

6.32.3.5 `void CglRedSplit::compute_is_lub ( )`

6.32.3.6 `void CglRedSplit::compute_is_integer ( )`

6.32.3.7 `void CglRedSplit::set_given_optsol ( const double * given_sol, const int card_sol )`

Set *given\_optsol* to the given optimal solution *given\_sol*.

If *given\_optsol* is set using this method, the code will stop as soon as a generated cut is violated by the given solution; exclusively for debugging purposes.

6.32.3.8 `void CglRedSplit::print ( ) const`

Print some of the data members.

6.32.3.9 `void CglRedSplit::printOptTab ( OsiSolverInterface * solver ) const`

Print the current simplex tableau.

**6.32.3.10 void CglRedSplit::setLimit ( int *limit* )**

Set limit, the maximum number of non zero coefficients in generated cut; Default: 50.

**6.32.3.11 int CglRedSplit::getLimit ( ) const**

Get value of limit.

**6.32.3.12 void CglRedSplit::setAway ( double *value* )**

Set away, the minimum distance from being integer used for selecting rows for cut generation; all rows whose pivot variable should be integer but is more than away from integrality will be selected; Default: 0.05.

**6.32.3.13 double CglRedSplit::getAway ( ) const**

Get value of away.

**6.32.3.14 void CglRedSplit::setLUB ( double *value* )**

Set the value of LUB, value considered large for the absolute value of a lower or upper bound on a variable; Default: 1000.

**6.32.3.15 double CglRedSplit::getLUB ( ) const**

Get the value of LUB.

**6.32.3.16 void CglRedSplit::setEPS ( double *value* )**

Set the value of EPS, epsilon for double computations; Default: 1e-7.

**6.32.3.17 double CglRedSplit::getEPS ( ) const**

Get the value of EPS.

**6.32.3.18 void CglRedSplit::setEPS\_COEFF ( double *value* )**

Set the value of EPS\_COEFF, epsilon for values of coefficients; Default: 1e-8.

**6.32.3.19 double CglRedSplit::getEPS\_COEFF ( ) const**

Get the value of EPS\_COEFF.

**6.32.3.20 void CglRedSplit::setEPS\_COEFF\_LUB ( double *value* )**

Set the value of EPS\_COEFF\_LUB, epsilon for values of coefficients for variables with absolute value of lower or upper bound larger than LUB; Default: 1e-13.

**6.32.3.21 double CglRedSplit::getEPS\_COEFF\_LUB ( ) const**

Get the value of EPS\_COEFF\_LUB.

**6.32.3.22 void CglRedSplit::setEPS\_RELAX ( double *value* )**

Set the value of EPS\_RELAX, value used for relaxing the right hand side of each generated cut; Default: 1e-8.

**6.32.3.23 double CglRedSplit::getEPS\_RELAX ( ) const**

Get the value of EPS\_RELAX.

6.32.3.24 `void CglRedSplit::setNormlsZero ( double value )`

Set the value of `normlsZero`, the threshold for considering a norm to be 0; Default: 1e-5.

6.32.3.25 `double CglRedSplit::getNormlsZero ( ) const`

Get the value of `normlsZero`.

6.32.3.26 `void CglRedSplit::setMinReduc ( double value )`

Set the value of `minReduc`, threshold for relative norm improvement for performing a reduction; Default: 0.05.

6.32.3.27 `double CglRedSplit::getMinReduc ( ) const`

Get the value of `minReduc`.

6.32.3.28 `void CglRedSplit::setMaxTab ( double value )`

Set the maximum allowed value for  $(mTab * mTab * \text{CoinMax}(mTab, nTab))$  where `mTab` is the number of rows used in the combinations and `nTab` is the number of continuous non basic variables.

The work of the generator is proportional to  $(mTab * mTab * \text{CoinMax}(mTab, nTab))$ . Reducing the value of `maxTab` makes the generator faster, but weaker. Default: 1e7.

6.32.3.29 `double CglRedSplit::getMaxTab ( ) const`

Get the value of `maxTab`.

6.32.3.30 `virtual CglCutGenerator* CglRedSplit::clone ( ) const` [virtual]

Clone.

Implements [CglCutGenerator](#).

6.32.3.31 `CglRedSplit& CglRedSplit::operator= ( const CglRedSplit & rhs )`

Assignment operator.

6.32.3.32 `virtual std::string CglRedSplit::generateCpp ( FILE * fp )` [virtual]

Create C++ lines to get to current state.

Reimplemented from [CglCutGenerator](#).

## 6.32.4 Friends And Related Function Documentation

6.32.4.1 `void CglRedSplitUnitTest ( const OsiSolverInterface * siP, const std::string mpdDir )` [friend]

A function that tests the methods in the [CglRedSplit](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

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

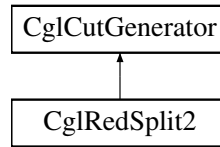
- `src/CglRedSplit/CglRedSplit.hpp`

### 6.33 CglRedSplit2 Class Reference

Reduce-and-Split Cut Generator Class; See method [generateCuts\(\)](#).

```
#include <CglRedSplit2.hpp>
```

Inheritance diagram for CglRedSplit2:



#### Public Member Functions

##### generateCuts

- virtual void [generateCuts](#) (const OsiSolverInterface &si, OsiCuts &cs, const [CglTreeInfo](#) info=[CglTreeInfo](#)())  
*Generate Reduce-and-Split Mixed Integer Gomory cuts for the model of the solver interface si.*
- virtual bool [needsOptimalBasis](#) () const  
*Return true if needs optimal basis to do cuts (will return true)*
- int [generateMultipliers](#) (const OsiSolverInterface &si, int \*lambda, int maxNumMultipliers, int \*basicVariables=NULL, OsiCuts \*cs=NULL)
- int [tiltLandPcut](#) (const OsiSolverInterface \*si, double \*row, double rowRhs, int rownumber, const double \*xbar, const int \*newnonbasics, OsiRowCut \*cs, int \*lambda=NULL)

#### Public Methods

- void [setParam](#) (const [CglRedSplit2Param](#) &source)
- [CglRedSplit2Param](#) & [getParam](#) ()
- void [print](#) () const  
*Print some of the data members; used for debugging.*
- void [printOptTab](#) (OsiSolverInterface \*solver) const  
*Print the current simplex tableau.*

#### Constructors and destructors

- [CglRedSplit2](#) ()  
*Default constructor.*
- [CglRedSplit2](#) (const [CglRedSplit2Param](#) &RS\_param)  
*Constructor with specified parameters.*
- [CglRedSplit2](#) (const [CglRedSplit2](#) &)  
*Copy constructor.*
- virtual [CglCutGenerator](#) \* [clone](#) () const  
*Clone.*
- [CglRedSplit2](#) & [operator=](#) (const [CglRedSplit2](#) &rhs)  
*Assignment operator.*
- virtual [~CglRedSplit2](#) ()  
*Destructor.*

#### Friends

- void [CglRedSplit2UnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)  
*A function that tests some of the methods in the [CglRedSplit2](#) class.*

## Additional Inherited Members

### 6.33.1 Detailed Description

Reduce-and-Split Cut Generator Class; See method [generateCuts\(\)](#).

Based on the papers "Practical strategies for generating rank-1 split cuts in mixed-integer linear programming" by G. Cornuejols and G. Nannicini, published on Mathematical Programming Computation, and "Combining Lift-and-Project and Reduce-and-Split" by E. Balas, G. Cornuejols, T. Kis and G. Nannicini, published on INFORMS Journal on Computing. Part of this code is based on [CglRedSplit](#) by F. Margot.

Definition at line 31 of file CglRedSplit2.hpp.

### 6.33.2 Constructor & Destructor Documentation

#### 6.33.2.1 CglRedSplit2::CglRedSplit2 ( )

Default constructor.

#### 6.33.2.2 CglRedSplit2::CglRedSplit2 ( const CglRedSplit2Param & *RS\_param* )

Constructor with specified parameters.

#### 6.33.2.3 CglRedSplit2::CglRedSplit2 ( const CglRedSplit2 & )

Copy constructor.

#### 6.33.2.4 virtual CglRedSplit2::~~CglRedSplit2 ( ) [virtual]

Destructor.

### 6.33.3 Member Function Documentation

#### 6.33.3.1 virtual void CglRedSplit2::generateCuts ( const OsiSolverInterface & *si*, OsiCuts & *cs*, const CglTreeInfo *info* = CglTreeInfo() ) [virtual]

Generate Reduce-and-Split Mixed Integer Gomory cuts for the model of the solver interface *si*.

Insert the generated cuts into OsiCuts *cs*.

This generator currently works only with the Lp solvers Clp or Cplex9.0 or higher. It requires access to the optimal tableau and optimal basis inverse and makes assumptions on the way slack variables are added by the solver. The Osi implementations for Clp and Cplex verify these assumptions.

When calling the generator, the solver interface *si* must contain an optimized problem and information related to the optimal basis must be available through the OsiSolverInterface methods (*si*->optimalBasisIsAvailable() must return 'true'). It is also essential that the integrality of structural variable *i* can be obtained using *si*->isInteger(*i*).

Reduce-and-Split cuts are a class of split cuts. We compute linear combinations of the rows of the simplex tableau, trying to reduce some of the coefficients on the nonbasic continuous columns. We have a large number of heuristics to choose which coefficients should be reduced, and by using which rows. The paper explains everything in detail.

Note that this generator can potentially generate a huge number of cuts, depending on how it is parametered. Default parameters should be good for most situations; if you want to go heavy on split cuts, use more row selection strategies or a different number of rows in the linear combinations. Again, look at the paper for details. If you want to generate a small number of cuts, default parameters are not the best choice.



A combination of Reduce-and-Split with Lift & Project is described in the paper "Combining Lift-and-Project and Reduce-and-Split". The Reduce-and-Split code for the implementation used in that paper is included here.

This generator does not generate the same cuts as [CglRedSplit](#), therefore both generators can be used in conjunction.

Implements [CglCutGenerator](#).

**6.33.3.2** `virtual bool CglRedSplit2::needsOptimalBasis ( ) const [virtual]`

Return true if needs optimal basis to do cuts (will return true)

Reimplemented from [CglCutGenerator](#).

**6.33.3.3** `int CglRedSplit2::generateMultipliers ( const OsiSolverInterface & si, int * lambda, int maxNumMultipliers, int * basicVariables = NULL, OsiCuts * cs = NULL )`

**6.33.3.4** `int CglRedSplit2::tiltLandPcut ( const OsiSolverInterface * si, double * row, double rowRhs, int rownumber, const double * xbar, const int * newnonbasics, OsiRowCut * cs, int * lambda = NULL )`

**6.33.3.5** `void CglRedSplit2::setParam ( const CglRedSplit2Param & source )`

**6.33.3.6** `CglRedSplit2Param& CglRedSplit2::getParam ( ) [inline]`

Definition at line 132 of file `CglRedSplit2.hpp`.

**6.33.3.7** `void CglRedSplit2::print ( ) const`

Print some of the data members; used for debugging.

**6.33.3.8** `void CglRedSplit2::printOptTab ( OsiSolverInterface * solver ) const`

Print the current simplex tableau.

**6.33.3.9** `virtual CglCutGenerator* CglRedSplit2::clone ( ) const [virtual]`

Clone.

Implements [CglCutGenerator](#).

**6.33.3.10** `CglRedSplit2& CglRedSplit2::operator= ( const CglRedSplit2 & rhs )`

Assignment operator.

#### 6.33.4 Friends And Related Function Documentation

**6.33.4.1** `void CglRedSplit2UnitTest ( const OsiSolverInterface * siP, const std::string mpdDir ) [friend]`

A function that tests some of the methods in the [CglRedSplit2](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

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

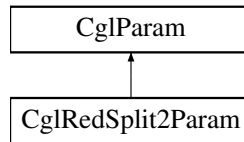
- `src/CglRedSplit2/CglRedSplit2.hpp`

## 6.34 CglRedSplit2Param Class Reference

Class collecting parameters the Reduced-and-split cut generator.

```
#include <CglRedSplit2Param.hpp>
```

Inheritance diagram for CglRedSplit2Param:



### Public Types

- enum [RowSelectionStrategy](#) {  
[RS1](#), [RS2](#), [RS3](#), [RS4](#),  
[RS5](#), [RS6](#), [RS7](#), [RS8](#),  
[RS\\_ALL](#), [RS\\_BEST](#) }  
*Enumerations for parameters.*
- enum [ColumnSelectionStrategy](#) {  
[CS1](#), [CS2](#), [CS3](#), [CS4](#),  
[CS5](#), [CS6](#), [CS7](#), [CS8](#),  
[CS9](#), [CS10](#), [CS11](#), [CS12](#),  
[CS13](#), [CS14](#), [CS15](#), [CS16](#),  
[CS17](#), [CS18](#), [CS19](#), [CS20](#),  
[CS21](#), [CS\\_ALL](#), [CS\\_BEST](#), [CS\\_ALLCONT](#),  
[CS\\_LAP\\_NONBASICS](#) }  
*Column selection strategies; again, look them up in the paper.*
- enum [ColumnScalingStrategy](#) {  
[SC\\_NONE](#), [SC\\_LINEAR](#), [SC\\_LINEAR\\_BOUNDED](#), [SC\\_LOG\\_BOUNDED](#),  
[SC\\_UNIFORM](#), [SC\\_UNIFORM\\_NZ](#) }  
*Scaling strategies for new nonbasic columns for Lift & Project; "factor" is the value of columnScalingBoundLAP\_.*

### Public Member Functions

#### Set/get methods

- virtual void [setAway](#) (double value)  
*Set away, the minimum distance from being integer used for selecting rows for cut generation; all rows whose pivot variable should be integer but is more than away from integrality will be selected; Default: 0.005.*
- double [getAway](#) () const  
*Get value of away.*
- void [setEPS\\_ELIM](#) (double value)  
*Set the value of EPS\_ELIM, epsilon for values of coefficients when eliminating slack variables; Default: 0.0.*
- double [getEPS\\_ELIM](#) () const  
*Get the value of EPS\_ELIM.*
- virtual void [setEPS\\_RELAX\\_ABS](#) (double eps\_ra)  
*Set EPS\_RELAX\_ABS.*
- double [getEPS\\_RELAX\\_ABS](#) () const  
*Get value of EPS\_RELAX\_ABS.*
- virtual void [setEPS\\_RELAX\\_REL](#) (double eps\_rr)

- *Set EPS\_RELAX\_REL.*
- double [getEPS\\_RELAX\\_REL](#) () const
- *Get value of EPS\_RELAX\_REL.*
- virtual void [setMaxDYN](#) (double value)
- double [getMAXDYN](#) () const
- *Get the value of MAXDYN.*
- virtual void [setMINVIOL](#) (double value)
- *Set the value of MINVIOL, the minimum violation for the current basic solution in a generated cut.*
- double [getMINVIOL](#) () const
- *Get the value of MINVIOL.*
- void [setMax\\_SUPP\\_ABS](#) (int value)
- *Maximum absolute support of the cutting planes.*
- int [getMAX\\_SUPP\\_ABS](#) () const
- void [setMax\\_SUPP\\_REL](#) (double value)
- *Maximum relative support of the cutting planes.*
- double [getMAX\\_SUPP\\_REL](#) () const
- virtual void [setUSE\\_INTSLACKS](#) (int value)
- *Set the value of USE\_INTSLACKS.*
- int [getUSE\\_INTSLACKS](#) () const
- *Get the value of USE\_INTSLACKS.*
- virtual void [setNormIsZero](#) (double value)
- *Set the value of normIsZero, the threshold for considering a norm to be 0; Default: 1e-5.*
- double [getNormIsZero](#) () const
- *Get the value of normIsZero.*
- virtual void [setMinNormReduction](#) (double value)
- *Set the value of minNormReduction; Default: 0.1.*
- double [getMinNormReduction](#) () const
- *Get the value of normIsZero.*
- virtual void [setMaxSumMultipliers](#) (int value)
- *Set the value of maxSumMultipliers; Default: 10.*
- int [getMaxSumMultipliers](#) () const
- *Get the value of maxSumMultipliers.*
- virtual void [setNormalization](#) (double value)
- *Set the value of normalization; Default: 0.0001.*
- double [getNormalization](#) () const
- *Get the value of normalization.*
- virtual void [addNumRowsReduction](#) (int value)
- *Set the value of numRowsReduction, max number of rows that are used for each row reduction step.*
- std::vector< int > [getNumRowsReduction](#) () const
- *get the value*
- void [resetNumRowsReduction](#) ()
- *reset*
- virtual void [addColumnSelectionStrategy](#) ([ColumnSelectionStrategy](#) value)
- *Add the value of columnSelectionStrategy.*
- std::vector
- < [ColumnSelectionStrategy](#) > [getColumnSelectionStrategy](#) () const
- *get the value*
- void [resetColumnSelectionStrategy](#) ()
- *reset*
- virtual void [addRowSelectionStrategy](#) ([RowSelectionStrategy](#) value)
- *Set the value for rowSelectionStrategy, which changes the way we choose the rows for the reduction step.*
- std::vector< [RowSelectionStrategy](#) > [getRowSelectionStrategy](#) () const
- *get the value*
- void [resetRowSelectionStrategy](#) ()
- *reset*

- virtual void [addNumRowsReductionLAP](#) (int value)  
*Set the value of numRowsReductionLAP, max number of rows that are used for each row reduction step during Lift & Project.*
- std::vector< int > [getNumRowsReductionLAP](#) () const  
*get the value*
- void [resetNumRowsReductionLAP](#) ()  
*reset*
- virtual void [addColumnSelectionStrategyLAP](#) (ColumnSelectionStrategy value)  
*Add the value of columnSelectionStrategyLAP.*
- std::vector< ColumnSelectionStrategy > [getColumnSelectionStrategyLAP](#) () const  
*get the value*
- void [resetColumnSelectionStrategyLAP](#) ()  
*reset*
- virtual void [addRowSelectionStrategyLAP](#) (RowSelectionStrategy value)  
*Set the value for rowSelectionStrategyLAP, which changes the way we choose the rows for the reduction step.*
- std::vector< RowSelectionStrategy > [getRowSelectionStrategyLAP](#) () const  
*get the value*
- void [resetRowSelectionStrategyLAP](#) ()  
*reset*
- virtual void [setColumnScalingStrategyLAP](#) (ColumnScalingStrategy value)  
*Set the value for columnScalingStrategyLAP, which sets the way nonbasic columns that are basic in the fractional point to cut off are scaled.*
- ColumnScalingStrategy [getColumnScalingStrategyLAP](#) () const  
*get the value*
- virtual void [setColumnScalingBoundLAP](#) (double value)  
*Set the value for the bound in the column scaling factor.*
- double [getColumnScalingBoundLAP](#) () const  
*get the value*
- virtual void [setTimeLimit](#) (double value)  
*Set the value of the time limit for cut generation (in seconds)*
- double [getTimeLimit](#) () const  
*get the value*
- virtual void [setMaxNumCuts](#) (int value)  
*Set the value for the maximum number of cuts that can be returned.*
- int [getNumMaxNumCuts](#) () const  
*get the value*
- virtual void [setMaxNumComputedCuts](#) (int value)  
*Set the value for the maximum number of cuts that can be computed.*
- int [getNumMaxNumComputedCuts](#) () const  
*get the value*
- virtual void [setMaxNonzeroesTab](#) (int value)  
*Set the value for the maximum number of nonzeroes in a row of the simplex tableau for the row to be considered.*
- int [getNumMaxNonzeroesTab](#) () const  
*get the value*
- virtual void [setSkipGomory](#) (int value)  
*Set the value of skipGomory: should we skip simple Gomory cuts, i.e.*
- int [getNumSkipGomory](#) () const  
*get the value*

### Constructors and destructors

- [CglRedSplit2Param](#) (bool use\_default\_strategies=true, double eps=1e-12, double eps\_coeff=1e-11, double eps\_elim=0.0, double eps\_relax\_abs=1e-11, double eps\_relax\_rel=1e-13, double max\_dyn=1e6, double max\_viol=1e-3, int max\_supp\_abs=1000, double max\_supp\_rel=0.1, int use\_int\_slacks=0, double norm\_zero=1e-5, double minNormReduction=0.1, int maxSumMultipliers=10, double normalization=0.0001, double away=0.005, double timeLimit=60, int maxNumCuts=10000, int maxNumComputedCuts=10000, int maxNonzeroesTab=1000, double columnScalingBoundLAP=5.0, int skipGomory=1)

*Default constructor.*

- `CglRedSplit2Param` (const `CglParam` &source, bool use\_default\_strategies=true, double eps\_elim=0.0, double eps\_relax\_abs=1e-11, double eps\_relax\_rel=1e-13, double max\_dyn=1e6, double min\_viol=1e-3, double max\_supp\_rel=0.1, int use\_int\_slacks=0, double norm\_zero=1e-5, double minNormReduction=0.1, int maxSumMultipliers=10, double normalization=0.0001, double away=0.005, double timeLimit=60, int maxNumCuts=10000, int maxNumComputedCuts=10000, int maxNonzeroesTab=1000, double columnScalingBoundLAP=5.0, int skipGomory=1)

*Constructor from `CglParam`.*

- `CglRedSplit2Param` (const `CglRedSplit2Param` &source)

*Copy constructor.*

- virtual `CglRedSplit2Param` \* clone () const

*Clone.*

- virtual `CglRedSplit2Param` & operator= (const `CglRedSplit2Param` &rhs)

*Assignment operator.*

- virtual ~`CglRedSplit2Param` ()

*Destructor.*

## Protected Attributes

### Parameters

- double `EPS_ELIM`

*Epsilon for value of coefficients when eliminating slack variables.*

- double `EPS_RELAX_ABS`

*Value added to the right hand side of each generated cut to relax it.*

- double `EPS_RELAX_REL`

*For a generated cut with right hand side rhs\_val,  $EPS\_RELAX\_EPS * fabs(rhs\_val)$  is used to relax the constraint.*

- double `MAXDYN`

- double `MINVIOL`

*Minimum violation for the current basic solution in a generated cut.*

- double `MAX_SUPP_REL`

*Maximum support - relative part of the formula.*

- int `USE_INTSLACKS`

*Use integer slacks to generate cuts if `USE_INTSLACKS` = 1. Default: 0.*

- double `normIsZero_`

*Norm of a vector is considered zero if smaller than `normIsZero`; Default: 1e-5.*

- double `minNormReduction_`

*Minimum reduction to accept a new row.*

- int `maxSumMultipliers_`

*Maximum sum of the vector of row multipliers to generate a cut.*

- double `normalization_`

*Normalization factor for the norm of lambda in the quadratic minimization problem that is solved during the coefficient reduction step.*

- double `away_`

*Use row only if pivot variable should be integer but is more than `away_` from being integer.*

- std::vector< int > `numRowsReduction_`

*Maximum number of rows to use for the reduction of a given row.*

- std::vector

< `ColumnSelectionStrategy` > `columnSelectionStrategy_`

*Column selection method.*

- std::vector< `RowSelectionStrategy` > `rowSelectionStrategy_`

*Row selection method.*

- std::vector< int > `numRowsReductionLAP_`

*Maximum number of rows to use for the reduction during Lift & Project.*

- `std::vector`  
`< ColumnSelectionStrategy > columnSelectionStrategyLAP_`  
*Column selection method for Lift & Project.*
- `std::vector< RowSelectionStrategy > rowSelectionStrategyLAP_`  
*Row selection method for Lift & Project.*
- `ColumnScalingStrategy columnScalingStrategyLAP_`  
*Column scaling strategy for the nonbasics columns that were basic in the point that we want to cut off (Lift & Project only)*
- `double columnScalingBoundLAP_`  
*Minimum value for column scaling (Lift & Project only)*
- `double timeLimit_`  
*Time limit.*
- `int maxNumCuts_`  
*Maximum number of returned cuts.*
- `int maxNumComputedCuts_`  
*Maximum number of computed cuts.*
- `int maxNonzeroesTab_`  
*Maximum number of nonzeros in tableau row for reduction.*
- `int skipGomory_`  
*Skip simple Gomory cuts.*

#### 6.34.1 Detailed Description

Class collecting parameters the Reduced-and-split cut generator.

An important thing to note is that the cut generator allows for the selection of a number of strategies that can be combined together. By default, a selection that typically yields a good compromise between speed and cut strenght is made. The selection can be changed by resetting the default choices (see the functions whose name starts with "reset") or by setting the parameter `use_default_strategies` to false in the constructors. After this, the chosen strategies can be added to the list by using the functions whose name starts with "add". All strategies will be combined together: if we choose 3 row selection strategies, 2 column selection strategies, and 2 possible numbers of rows, we end up with a total of  $3 \times 2 \times 2$  combinations.

For a detailed explanation of the parameters and their meaning, see the paper by Cornuejols and Nannicini: "Practical strategies for generating rank-1 split cuts in mixed-integer linear programming", on Mathematical Programming Computation.

Parameters of the generator are listed below.

- `MAXDYN`: Maximum ratio between largest and smallest non zero coefficients in a cut. See method `setMaxDYN()`.
- `EPS_ELIM`: Precision for deciding if a coefficient is zero when eliminating slack variables. See method `setEPS_ELIM()`.
- `MINVIOL`: Minimum violation for the current basic solution in a generated cut. See method `setMINVIOL()`.
- `EPS_RELAX_ABS`: Absolute relaxation of cut rhs.
- `EPS_RELAX_REL`: Relative relaxation of cut rhs.
- `MAX_SUPP_ABS`: Maximum cut support (absolute).
- `MAX_SUPP_REL`: Maximum cut support (relative): the formula to compute maximum cut support is `MAX_SUPP_ABS + ncol*MAX_SUPP_REL`.
- `USE_INTSLACKS`: Use integer slacks to generate cuts. (not implemented). See method `setUSE_INTSLACKS()`.
- `normIsZero`: Norm of a vector is considered zero if smaller than this value. See method `setNormIsZero()`.

- minNormReduction: a cut is generated if the new norm of the row on the continuous nonbasics is reduced by at least this factor (relative reduction).
- away: Look only at basic integer variables whose current value is at least this value from being integer. See method [setAway\(\)](#).
- maxSumMultipliers: maximum sum (in absolute value) of row multipliers
- normalization: normalization factor for the norm of lambda in the coefficient reduction algorithm (convex min problem)
- numRowsReduction: Maximum number of rows in the linear system for norm reduction.
- columnSelectionStrategy: parameter to select which columns should be used for coefficient reduction.
- rowSelectionStrategy: parameter to select which rows should be used for coefficient reduction.
- timeLimit: Time limit (in seconds) for cut generation.
- maxNumCuts: Maximum number of cuts that can be returned at each pass; we could generate more cuts than this number (see below)
- maxNumComputedCuts: Maximum number of cuts that can be computed by the generator at each pass
- maxNonzeroesTab : Rows of the simplex tableau with more than this number of nonzeroes will not be considered for reduction. Only works if RS\_FAST\_\* are defined in [CglRedSplit2](#).
- skipGomory: Skip traditional Gomory cuts, i.e. GMI cuts arising from a single row of the tableau (instead of a combination). Default is 1 (true), because we assume that they are generated by a traditional Gomory generator anyway.

Definition at line 88 of file CglRedSplit2Param.hpp.

### 6.34.2 Member Enumeration Documentation

#### 6.34.2.1 enum CglRedSplit2Param::RowSelectionStrategy

Enumerations for parameters.

Row selection strategies; same names as in the paper

Enumerator

***RS1***  
***RS2***  
***RS3***  
***RS4***  
***RS5***  
***RS6***  
***RS7***  
***RS8***  
***RS\_ALL***  
***RS\_BEST***

Definition at line 94 of file CglRedSplit2Param.hpp.

#### 6.34.2.2 enum CglRedSplit2Param::ColumnSelectionStrategy

Column selection strategies; again, look them up in the paper.

Enumerator

***CS1***  
***CS2***  
***CS3***  
***CS4***  
***CS5***  
***CS6***  
***CS7***  
***CS8***  
***CS9***  
***CS10***  
***CS11***  
***CS12***  
***CS13***  
***CS14***  
***CS15***  
***CS16***  
***CS17***  
***CS18***  
***CS19***  
***CS20***  
***CS21***  
***CS\_ALL***  
***CS\_BEST***  
***CS\_ALLCONT***  
***CS\_LAP\_NONBASICS***

Definition at line 122 of file CglRedSplit2Param.hpp.

#### 6.34.2.3 enum CglRedSplit2Param::ColumnScalingStrategy

Scaling strategies for new nonbasic columns for Lift & Project; "factor" is the value of columnScalingBoundLAP\_.

Enumerator

***SC\_NONE***  
***SC\_LINEAR***  
***SC\_LINEAR\_BOUNDED***  
***SC\_LOG\_BOUNDED***  
***SC\_UNIFORM***  
***SC\_UNIFORM\_NZ***

Definition at line 154 of file CglRedSplit2Param.hpp.



### 6.34.3 Constructor & Destructor Documentation

**6.34.3.1** `CglRedSplit2Param::CglRedSplit2Param ( bool use_default_strategies = true, double eps = 1e-12, double eps_coeff = 1e-11, double eps_elim = 0.0, double eps_relax_abs = 1e-11, double eps_relax_rel = 1e-13, double max_dyn = 1e6, double min_viol = 1e-3, int max_supp_abs = 1000, double max_supp_rel = 0.1, int use_int_slacks = 0, double norm_zero = 1e-5, double minNormReduction = 0.1, int maxSumMultipliers = 10, double normalization = 0.0001, double away = 0.005, double timeLimit = 60, int maxNumCuts = 10000, int maxNumComputedCuts = 10000, int maxNonzeroesTab = 1000, double columnScalingBoundLAP = 5.0, int skipGomory = 1 )`

Default constructor.

If `use_default_strategies` is true, we add to the list of strategies the default ones. If is false, the list of strategies is left empty (must be populated before usage!).

**6.34.3.2** `CglRedSplit2Param::CglRedSplit2Param ( const CglParam & source, bool use_default_strategies = true, double eps_elim = 0.0, double eps_relax_abs = 1e-11, double eps_relax_rel = 1e-13, double max_dyn = 1e6, double min_viol = 1e-3, double max_supp_rel = 0.1, int use_int_slacks = 0, double norm_zero = 1e-5, double minNormReduction = 0.1, int maxSumMultipliers = 10, double normalization = 0.0001, double away = 0.005, double timeLimit = 60, int maxNumCuts = 10000, int maxNumComputedCuts = 10000, int maxNonzeroesTab = 1000, double columnScalingBoundLAP = 5.0, int skipGomory = 1 )`

Constructor from [CglParam](#).

If `use_default_strategies` is true, we add to the list of strategies the default ones. If is false, the list of strategies is left empty (must be populated before usage!).

**6.34.3.3** `CglRedSplit2Param::CglRedSplit2Param ( const CglRedSplit2Param & source )`

Copy constructor.

**6.34.3.4** `virtual CglRedSplit2Param::~CglRedSplit2Param ( ) [virtual]`

Destructor.

### 6.34.4 Member Function Documentation

**6.34.4.1** `virtual void CglRedSplit2Param::setAway ( double value ) [virtual]`

Set away, the minimum distance from being integer used for selecting rows for cut generation; all rows whose pivot variable should be integer but is more than away from integrality will be selected; Default: 0.005.

**6.34.4.2** `double CglRedSplit2Param::getAway ( ) const [inline]`

Get value of away.

Definition at line 178 of file `CglRedSplit2Param.hpp`.

**6.34.4.3** `void CglRedSplit2Param::setEPS_ELIM ( double value )`

Set the value of EPS\_ELIM, epsilon for values of coefficients when eliminating slack variables; Default: 0.0.

**6.34.4.4** `double CglRedSplit2Param::getEPS_ELIM ( ) const [inline]`

Get the value of EPS\_ELIM.

Definition at line 185 of file `CglRedSplit2Param.hpp`.

6.34.4.5 `virtual void CglRedSplit2Param::setEPS_RELAX_ABS ( double eps_ra ) [virtual]`

Set EPS\_RELAX\_ABS.

6.34.4.6 `double CglRedSplit2Param::getEPS_RELAX_ABS ( ) const [inline]`

Get value of EPS\_RELAX\_ABS.

Definition at line 190 of file CglRedSplit2Param.hpp.

6.34.4.7 `virtual void CglRedSplit2Param::setEPS_RELAX_REL ( double eps_rr ) [virtual]`

Set EPS\_RELAX\_REL.

6.34.4.8 `double CglRedSplit2Param::getEPS_RELAX_REL ( ) const [inline]`

Get value of EPS\_RELAX\_REL.

Definition at line 195 of file CglRedSplit2Param.hpp.

6.34.4.9 `virtual void CglRedSplit2Param::setMaxDYN ( double value ) [virtual]`

6.34.4.10 `double CglRedSplit2Param::getMAXDYN ( ) const [inline]`

Get the value of MAXDYN.

Definition at line 201 of file CglRedSplit2Param.hpp.

6.34.4.11 `virtual void CglRedSplit2Param::setMINVIOL ( double value ) [virtual]`

Set the value of MINVIOL, the minimum violation for the current basic solution in a generated cut.

Default: 1e-3

6.34.4.12 `double CglRedSplit2Param::getMINVIOL ( ) const [inline]`

Get the value of MINVIOL.

Definition at line 207 of file CglRedSplit2Param.hpp.

6.34.4.13 `void CglRedSplit2Param::setMax_SUPP_ABS ( int value ) [inline]`

Maximum absolute support of the cutting planes.

Default: INT\_MAX. Aliases for consistency with our naming scheme.

Definition at line 211 of file CglRedSplit2Param.hpp.

6.34.4.14 `int CglRedSplit2Param::getMAX_SUPP_ABS ( ) const [inline]`

Definition at line 212 of file CglRedSplit2Param.hpp.

6.34.4.15 `void CglRedSplit2Param::setMax_SUPP_REL ( double value ) [inline]`

Maximum relative support of the cutting planes.

Default: 0.0. The maximum support is MAX\_SUPP\_ABS + MAX\_SUPPREL\*ncols.

6.34.4.16 `double CglRedSplit2Param::getMAX_SUPP_REL ( ) const [inline]`

Definition at line 217 of file CglRedSplit2Param.hpp.

6.34.4.17 `virtual void CglRedSplit2Param::setUSE_INTSLACKS ( int value ) [virtual]`

Set the value of USE\_INTSLACKS.

Default: 0

6.34.4.18 `int CglRedSplit2Param::getUSE_INTSLACKS ( ) const [inline]`

Get the value of USE\_INTSLACKS.

Definition at line 222 of file CglRedSplit2Param.hpp.

6.34.4.19 `virtual void CglRedSplit2Param::setNormIsZero ( double value ) [virtual]`

Set the value of normIsZero, the threshold for considering a norm to be 0; Default: 1e-5.

6.34.4.20 `double CglRedSplit2Param::getNormIsZero ( ) const [inline]`

Get the value of normIsZero.

Definition at line 228 of file CglRedSplit2Param.hpp.

6.34.4.21 `virtual void CglRedSplit2Param::setMinNormReduction ( double value ) [virtual]`

Set the value of minNormReduction; Default: 0.1.

6.34.4.22 `double CglRedSplit2Param::getMinNormReduction ( ) const [inline]`

Get the value of normIsZero.

Definition at line 233 of file CglRedSplit2Param.hpp.

6.34.4.23 `virtual void CglRedSplit2Param::setMaxSumMultipliers ( int value ) [virtual]`

Set the value of maxSumMultipliers; Default: 10.

6.34.4.24 `int CglRedSplit2Param::getMaxSumMultipliers ( ) const [inline]`

Get the value of maxSumMultipliers.

Definition at line 238 of file CglRedSplit2Param.hpp.

6.34.4.25 `virtual void CglRedSplit2Param::setNormalization ( double value ) [virtual]`

Set the value of normalization; Default: 0.0001.

6.34.4.26 `double CglRedSplit2Param::getNormalization ( ) const [inline]`

Get the value of normalization.

Definition at line 243 of file CglRedSplit2Param.hpp.

6.34.4.27 `virtual void CglRedSplit2Param::addNumRowsReduction ( int value ) [virtual]`

Set the value of numRowsReduction, max number of rows that are used for each row reduction step.

In particular, the linear system will involve a numRowsReduction\*numRowsReduction matrix

6.34.4.28 `std::vector<int> CglRedSplit2Param::getNumRowsReduction ( ) const [inline]`

get the value

Definition at line 250 of file CglRedSplit2Param.hpp.

6.34.4.29 `void CglRedSplit2Param::resetNumRowsReduction ( ) [inline]`

reset

Definition at line 252 of file CglRedSplit2Param.hpp.

6.34.4.30 `virtual void CglRedSplit2Param::addColumnSelectionStrategy ( ColumnSelectionStrategy value ) [virtual]`

Add the value of columnSelectionStrategy.

6.34.4.31 `std::vector<ColumnSelectionStrategy> CglRedSplit2Param::getColumnSelectionStrategy ( ) const [inline]`

get the value

Definition at line 257 of file CglRedSplit2Param.hpp.

6.34.4.32 `void CglRedSplit2Param::resetColumnSelectionStrategy ( ) [inline]`

reset

Definition at line 259 of file CglRedSplit2Param.hpp.

6.34.4.33 `virtual void CglRedSplit2Param::addRowSelectionStrategy ( RowSelectionStrategy value ) [virtual]`

Set the value for rowSelectionStrategy, which changes the way we choose the rows for the reduction step.

6.34.4.34 `std::vector<RowSelectionStrategy> CglRedSplit2Param::getRowSelectionStrategy ( ) const [inline]`

get the value

Definition at line 265 of file CglRedSplit2Param.hpp.

6.34.4.35 `void CglRedSplit2Param::resetRowSelectionStrategy ( ) [inline]`

reset

Definition at line 267 of file CglRedSplit2Param.hpp.

6.34.4.36 `virtual void CglRedSplit2Param::addNumRowsReductionLAP ( int value ) [virtual]`

Set the value of numRowsReductionLAP, max number of rows that are used for each row reduction step during Lift & Project.

In particular, the linear system will involve a numRowsReduction\*numRowsReduction matrix

6.34.4.37 `std::vector<int> CglRedSplit2Param::getNumRowsReductionLAP ( ) const [inline]`

get the value

Definition at line 275 of file CglRedSplit2Param.hpp.

6.34.4.38 `void CglRedSplit2Param::resetNumRowsReductionLAP ( ) [inline]`

reset

Definition at line 277 of file CglRedSplit2Param.hpp.

6.34.4.39 `virtual void CglRedSplit2Param::addColumnSelectionStrategyLAP ( ColumnSelectionStrategy value )`  
`[virtual]`

Add the value of columnSelectionStrategyLAP.

6.34.4.40 `std::vector<ColumnSelectionStrategy> CglRedSplit2Param::getColumnSelectionStrategyLAP ( ) const`  
`[inline]`

get the value

Definition at line 282 of file CglRedSplit2Param.hpp.

6.34.4.41 `void CglRedSplit2Param::resetColumnSelectionStrategyLAP ( )` `[inline]`

reset

Definition at line 284 of file CglRedSplit2Param.hpp.

6.34.4.42 `virtual void CglRedSplit2Param::addRowSelectionStrategyLAP ( RowSelectionStrategy value )` `[virtual]`

Set the value for rowSelectionStrategyLAP, which changes the way we choose the rows for the reduction step.

6.34.4.43 `std::vector<RowSelectionStrategy> CglRedSplit2Param::getRowSelectionStrategyLAP ( ) const` `[inline]`

get the value

Definition at line 290 of file CglRedSplit2Param.hpp.

6.34.4.44 `void CglRedSplit2Param::resetRowSelectionStrategyLAP ( )` `[inline]`

reset

Definition at line 292 of file CglRedSplit2Param.hpp.

6.34.4.45 `virtual void CglRedSplit2Param::setColumnScalingStrategyLAP ( ColumnScalingStrategy value )` `[virtual]`

Set the value for columnScalingStrategyLAP, which sets the way nonbasic columns that are basic in the fractional point to cut off are scaled.

6.34.4.46 `ColumnScalingStrategy CglRedSplit2Param::getColumnScalingStrategyLAP ( ) const` `[inline]`

get the value

Definition at line 298 of file CglRedSplit2Param.hpp.

6.34.4.47 `virtual void CglRedSplit2Param::setColumnScalingBoundLAP ( double value )` `[virtual]`

Set the value for the bound in the column scaling factor.

6.34.4.48 `double CglRedSplit2Param::getColumnScalingBoundLAP ( ) const` `[inline]`

get the value

Definition at line 303 of file CglRedSplit2Param.hpp.

6.34.4.49 `virtual void CglRedSplit2Param::setTimeLimit ( double value )` `[virtual]`

Set the value of the time limit for cut generation (in seconds)

6.34.4.50 `double CglRedSplit2Param::getTimeLimit ( ) const` `[inline]`

get the value

Definition at line 308 of file CglRedSplit2Param.hpp.

6.34.4.51 `virtual void CglRedSplit2Param::setMaxNumCuts ( int value )` `[virtual]`

Set the value for the maximum number of cuts that can be returned.

6.34.4.52 `int CglRedSplit2Param::getMaxNumCuts ( ) const` `[inline]`

get the value

Definition at line 313 of file CglRedSplit2Param.hpp.

6.34.4.53 `virtual void CglRedSplit2Param::setMaxNumComputedCuts ( int value )` `[virtual]`

Set the value for the maximum number of cuts that can be computed.

6.34.4.54 `int CglRedSplit2Param::getMaxNumComputedCuts ( ) const` `[inline]`

get the value

Definition at line 318 of file CglRedSplit2Param.hpp.

6.34.4.55 `virtual void CglRedSplit2Param::setMaxNonzeroesTab ( int value )` `[virtual]`

Set the value for the maximum number of nonzeroes in a row of the simplex tableau for the row to be considered.

6.34.4.56 `int CglRedSplit2Param::getMaxNonzeroesTab ( ) const` `[inline]`

get the value

Definition at line 324 of file CglRedSplit2Param.hpp.

6.34.4.57 `virtual void CglRedSplit2Param::setSkipGomory ( int value )` `[virtual]`

Set the value of skipGomory: should we skip simple Gomory cuts, i.e.

GMI cuts derived from a single row of the simple tableau? This is 1 (true) by default: we only generate cuts from linear combinations of at least two rows.

6.34.4.58 `int CglRedSplit2Param::getSkipGomory ( ) const` `[inline]`

get the value

Definition at line 332 of file CglRedSplit2Param.hpp.

6.34.4.59 `virtual CglRedSplit2Param* CglRedSplit2Param::clone ( ) const` `[virtual]`

Clone.

Reimplemented from [CglParam](#).

6.34.4.60 `virtual CglRedSplit2Param& CglRedSplit2Param::operator= ( const CglRedSplit2Param & rhs )` `[virtual]`

Assignment operator.

### 6.34.5 Member Data Documentation

#### 6.34.5.1 `double CglRedSplit2Param::EPS_ELIM` [protected]

Epsilon for value of coefficients when eliminating slack variables.

Default: 0.0.

Definition at line 409 of file CglRedSplit2Param.hpp.

#### 6.34.5.2 `double CglRedSplit2Param::EPS_RELAX_ABS` [protected]

Value added to the right hand side of each generated cut to relax it.

Default: 1e-11

Definition at line 413 of file CglRedSplit2Param.hpp.

#### 6.34.5.3 `double CglRedSplit2Param::EPS_RELAX_REL` [protected]

For a generated cut with right hand side `rhs_val`, `EPS_RELAX_EPS * fabs(rhs_val)` is used to relax the constraint.

Default: 1e-13

Definition at line 418 of file CglRedSplit2Param.hpp.

#### 6.34.5.4 `double CglRedSplit2Param::MAXDYN` [protected]

Definition at line 422 of file CglRedSplit2Param.hpp.

#### 6.34.5.5 `double CglRedSplit2Param::MINVIOL` [protected]

Minimum violation for the current basic solution in a generated cut.

Default: 1e-3.

Definition at line 426 of file CglRedSplit2Param.hpp.

#### 6.34.5.6 `double CglRedSplit2Param::MAX_SUPP_REL` [protected]

Maximum support - relative part of the formula.

Definition at line 429 of file CglRedSplit2Param.hpp.

#### 6.34.5.7 `int CglRedSplit2Param::USE_INTSLACKS` [protected]

Use integer slacks to generate cuts if `USE_INTSLACKS = 1`. Default: 0.

Definition at line 432 of file CglRedSplit2Param.hpp.

#### 6.34.5.8 `double CglRedSplit2Param::normIsZero_` [protected]

Norm of a vector is considered zero if smaller than `normIsZero`; Default: 1e-5.

Definition at line 436 of file CglRedSplit2Param.hpp.

#### 6.34.5.9 `double CglRedSplit2Param::minNormReduction_` [protected]

Minimum reduction to accept a new row.

Definition at line 439 of file CglRedSplit2Param.hpp.

**6.34.5.10** `int CglRedSplit2Param::maxSumMultipliers_` `[protected]`

Maximum sum of the vector of row multipliers to generate a cut.

Definition at line 442 of file CglRedSplit2Param.hpp.

**6.34.5.11** `double CglRedSplit2Param::normalization_` `[protected]`

Normalization factor for the norm of lambda in the quadratic minimization problem that is solved during the coefficient reduction step.

Definition at line 446 of file CglRedSplit2Param.hpp.

**6.34.5.12** `double CglRedSplit2Param::away_` `[protected]`

Use row only if pivot variable should be integer but is more than away\_ from being integer.

Default: 0.005

Definition at line 450 of file CglRedSplit2Param.hpp.

**6.34.5.13** `std::vector<int> CglRedSplit2Param::numRowsReduction_` `[protected]`

Maximum number of rows to use for the reduction of a given row.

Definition at line 453 of file CglRedSplit2Param.hpp.

**6.34.5.14** `std::vector<ColumnSelectionStrategy> CglRedSplit2Param::columnSelectionStrategy_` `[protected]`

Column selection method.

Definition at line 456 of file CglRedSplit2Param.hpp.

**6.34.5.15** `std::vector<RowSelectionStrategy> CglRedSplit2Param::rowSelectionStrategy_` `[protected]`

Row selection method.

Definition at line 459 of file CglRedSplit2Param.hpp.

**6.34.5.16** `std::vector<int> CglRedSplit2Param::numRowsReductionLAP_` `[protected]`

Maximum number of rows to use for the reduction during Lift & Project.

Definition at line 462 of file CglRedSplit2Param.hpp.

**6.34.5.17** `std::vector<ColumnSelectionStrategy> CglRedSplit2Param::columnSelectionStrategyLAP_` `[protected]`

Column selection method for Lift & Project.

Definition at line 465 of file CglRedSplit2Param.hpp.

**6.34.5.18** `std::vector<RowSelectionStrategy> CglRedSplit2Param::rowSelectionStrategyLAP_` `[protected]`

Row selection method for Lift & Project.

Definition at line 468 of file CglRedSplit2Param.hpp.

**6.34.5.19** `ColumnScalingStrategy CglRedSplit2Param::columnScalingStrategyLAP_` `[protected]`

Column scaling strategy for the nonbasics columns that were basic in the point that we want to cut off (Lift & Project only)



Definition at line 472 of file CglRedSplit2Param.hpp.

**6.34.5.20** `double CglRedSplit2Param::columnScalingBoundLAP_ [protected]`

Minimum value for column scaling (Lift & Project only)

Definition at line 475 of file CglRedSplit2Param.hpp.

**6.34.5.21** `double CglRedSplit2Param::timeLimit_ [protected]`

Time limit.

Definition at line 478 of file CglRedSplit2Param.hpp.

**6.34.5.22** `int CglRedSplit2Param::maxNumCuts_ [protected]`

Maximum number of returned cuts.

Definition at line 481 of file CglRedSplit2Param.hpp.

**6.34.5.23** `int CglRedSplit2Param::maxNumComputedCuts_ [protected]`

Maximum number of computed cuts.

Definition at line 484 of file CglRedSplit2Param.hpp.

**6.34.5.24** `int CglRedSplit2Param::maxNonzeroesTab_ [protected]`

Maximum number of nonzeroes in tableau row for reduction.

Definition at line 487 of file CglRedSplit2Param.hpp.

**6.34.5.25** `int CglRedSplit2Param::skipGomory_ [protected]`

Skip simple Gomory cuts.

Definition at line 490 of file CglRedSplit2Param.hpp.

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

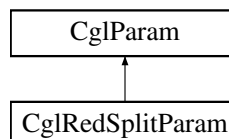
- src/CglRedSplit2/CglRedSplit2Param.hpp

## 6.35 CglRedSplitParam Class Reference

Class collecting parameters the Reduced-and-split cut generator.

```
#include <CglRedSplitParam.hpp>
```

Inheritance diagram for CglRedSplitParam:



### Public Member Functions

#### Set/get methods

- virtual void [setAway](#) (const double value)  
*Set away, the minimum distance from being integer used for selecting rows for cut generation; all rows whose pivot variable should be integer but is more than away from integrality will be selected; Default: 0.05.*
- double [getAway](#) () const  
*Get value of away.*
- virtual void [setLUB](#) (const double value)  
*Set the value of LUB, value considered large for the absolute value of a lower or upper bound on a variable; Default: 1000.*
- double [getLUB](#) () const  
*Get the value of LUB.*
- void [setEPS\\_ELIM](#) (const double value)  
*Set the value of EPS\_ELIM, epsilon for values of coefficients when eliminating slack variables; Default: 1e-12.*
- double [getEPS\\_ELIM](#) () const  
*Get the value of EPS\_ELIM.*
- virtual void [setEPS\\_RELAX\\_ABS](#) (const double eps\_ra)  
*Set EPS\_RELAX\_ABS.*
- double [getEPS\\_RELAX\\_ABS](#) () const  
*Get value of EPS\_RELAX\_ABS.*
- virtual void [setEPS\\_RELAX\\_REL](#) (const double eps\_rr)  
*Set EPS\_RELAX\_REL.*
- double [getEPS\\_RELAX\\_REL](#) () const  
*Get value of EPS\_RELAX\_REL.*
- virtual void [setMaxDYN](#) (double value)
- double [getMaxDYN](#) () const  
*Get the value of MAXDYN.*
- virtual void [setMaxDYN\\_LUB](#) (double value)
- double [getMaxDYN\\_LUB](#) () const  
*Get the value of MAXDYN\_LUB.*
- virtual void [setEPS\\_COEFF\\_LUB](#) (const double value)  
*Set the value of EPS\_COEFF\_LUB, epsilon for values of coefficients for variables with absolute value of lower or upper bound larger than LUB; Default: 1e-13.*
- double [getEPS\\_COEFF\\_LUB](#) () const  
*Get the value of EPS\_COEFF\_LUB.*
- virtual void [setMINVIOL](#) (double value)  
*Set the value of MINVIOL, the minimum violation for the current basic solution in a generated cut.*
- double [getMINVIOL](#) () const  
*Get the value of MINVIOL.*
- virtual void [setUSE\\_INTSLACKS](#) (int value)  
*Set the value of USE\_INTSLACKS.*
- int [getUSE\\_INTSLACKS](#) () const  
*Get the value of USE\_INTSLACKS.*
- virtual void [setUSE\\_CG2](#) (int value)  
*Set the value of USE\_CG2.*
- int [getUSE\\_CG2](#) () const  
*Get the value of USE\_CG2.*
- virtual void [setNormIsZero](#) (const double value)  
*Set the value of normIsZero, the threshold for considering a norm to be 0; Default: 1e-5.*
- double [getNormIsZero](#) () const  
*Get the value of normIsZero.*
- virtual void [setMinReduc](#) (const double value)  
*Set the value of minReduc, threshold for relative norm improvement for performing a reduction; Default: 0.05.*
- double [getMinReduc](#) () const  
*Get the value of minReduc.*
- virtual void [setMaxTab](#) (const double value)  
*Set the maximum allowed value for  $(mTab * mTab * \text{CoinMax}(mTab, nTab))$  where  $mTab$  is the number of rows used in the combinations and  $nTab$  is the number of continuous non basic variables.*

- double [getMaxTab](#) () const  
*Get the value of maxTab.*

### Constructors and destructors

- [CglRedSplitParam](#) (const double lub=1000.0, const double eps\_elim=1e-12, const double eps\_relax\_abs=1e-8, const double eps\_relax\_rel=0.0, const double max\_dyn=1e8, const double max\_dyn\_lub=1e13, const double eps\_coeff\_lub=1e-13, const double min\_viol=1e-7, const int use\_int\_slacks=0, const int use\_cg2=0, const double norm\_zero=1e-5, const double min\_reduc=0.05, const double away=0.05, const double max\_tab=1e7)  
*Default constructor.*
- [CglRedSplitParam](#) (const [CglParam](#) &source, const double lub=1000.0, const double eps\_elim=1e-12, const double eps\_relax\_abs=1e-8, const double eps\_relax\_rel=0.0, const double max\_dyn=1e8, const double max\_dyn\_lub=1e13, const double eps\_coeff\_lub=1e-13, const double min\_viol=1e-7, const int use\_int\_slacks=0, const int use\_cg2=0, const double norm\_zero=1e-5, const double min\_reduc=0.05, const double away=0.05, const double max\_tab=1e7)  
*Constructor from [CglParam](#).*
- [CglRedSplitParam](#) (const [CglRedSplitParam](#) &source)  
*Copy constructor.*
- virtual [CglRedSplitParam](#) \* [clone](#) () const  
*Clone.*
- virtual [CglRedSplitParam](#) & [operator=](#) (const [CglRedSplitParam](#) &rhs)  
*Assignment operator.*
- virtual [~CglRedSplitParam](#) ()  
*Destructor.*

### Protected Attributes

#### Parameters

- double [LUB](#)  
*Value considered large for the absolute value of lower or upper bound on a variable.*
- double [EPS\\_ELIM](#)  
*Epsilon for value of coefficients when eliminating slack variables.*
- double [EPS\\_RELAX\\_ABS](#)  
*Value added to the right hand side of each generated cut to relax it.*
- double [EPS\\_RELAX\\_REL](#)  
*For a generated cut with right hand side rhs\_val, EPS\_RELAX\_EPS \* fabs(rhs\_val) is used to relax the constraint.*
- double [MAXDYN](#)
- double [MAXDYN\\_LUB](#)
- double [EPS\\_COEFF\\_LUB](#)  
*Epsilon for value of coefficients for variables with absolute value of lower or upper bound larger than LUB.*
- double [MINVIOL](#)  
*Minimum violation for the current basic solution in a generated cut.*
- int [USE\\_INTSLACKS](#)  
*Use integer slacks to generate cuts if USE\_INTSLACKS = 1. Default: 0.*
- int [USE\\_CG2](#)  
*Use second way to generate a mixed integer Gomory cut (see methods generate\_cgcut()) and generate\_cgcut\_2()).*
- double [normIsZero](#)  
*Norm of a vector is considered zero if smaller than normIsZero; Default: 1e-5.*
- double [minReduc](#)  
*Minimum reduction in percent that must be achieved by a potential reduction step in order to be performed; Between 0 and 1, default: 0.05.*
- double [away\\_](#)  
*Use row only if pivot variable should be integer but is more than away\_ from being integer.*
- double [maxTab\\_](#)  
*Maximum value for (mTab \* mTab \* CoinMax(mTab, nTab)).*

## 6.35.1 Detailed Description

Class collecting parameters the Reduced-and-split cut generator.

Parameters of the generator are listed below. Modifying the default values for parameters other than the last four might result in invalid cuts.

- LUB: Value considered large for the absolute value of a lower or upper bound on a variable. See method [setLUB\(\)](#).
- MAXDYN: Maximum ratio between largest and smallest non zero coefficients in a cut. See method [setMaxDYN\(\)](#).
- MAXDYN\_LUB: Maximum ratio between largest and smallest non zero coefficients in a cut involving structural variables with lower or upper bound in absolute value larger than LUB. Should logically be larger or equal to MAXDYN. See method [setMaxDYN\\_LUB\(\)](#).
- EPS\_ELIM: Precision for deciding if a coefficient is zero when eliminating slack variables. See method [setEPS\\_ELIM\(\)](#).
- EPS\_COEFF\_LUB: Precision for deciding if a coefficient of a generated cut is zero when the corresponding variable has a lower or upper bound larger than LUB in absolute value. See method [setEPS\\_COEFF\\_LUB\(\)](#).
- MINVIOL: Minimum violation for the current basic solution in a generated cut. See method [setMINVIOL\(\)](#).
- USE\_INTSLACKS: Use integer slacks to generate cuts. (not implemented). See method [setUSE\\_INTSLACKS\(\)](#).
- USE\_CG2: Use alternative formula to generate a mixed integer Gomory cut (see methods [CglRedSplit::generate\\_cgcut\(\)](#) and [CglRedSplit::generate\\_cgcut\\_2\(\)](#)). See method [setUSE\\_CG2\(\)](#).
- normIsZero: Norm of a vector is considered zero if smaller than this value. See method [setNormIsZero\(\)](#).
- minReduc: Reduction is performed only if the norm of the vector is reduced by this fraction. See method [setMinReduc\(\)](#).
- away: Look only at basic integer variables whose current value is at least this value from being integer. See method [setAway\(\)](#).
- maxTab: Controls the number of rows selected for the generation. See method [setMaxTab\(\)](#).

Definition at line 61 of file [CglRedSplitParam.hpp](#).

## 6.35.2 Constructor &amp; Destructor Documentation

**6.35.2.1** `CglRedSplitParam::CglRedSplitParam ( const double lub = 1000.0, const double eps_elim = 1e-12, const double eps_relax_abs = 1e-8, const double eps_relax_rel = 0.0, const double max_dyn = 1e8, const double max_dyn_lub = 1e13, const double eps_coeff_lub = 1e-13, const double min_viol = 1e-7, const int use_int_slacks = 0, const int use_cg2 = 0, const double norm_zero = 1e-5, const double min_reduc = 0.05, const double away = 0.05, const double max_tab = 1e7 )`

Default constructor.

**6.35.2.2** `CglRedSplitParam::CglRedSplitParam ( const CglParam & source, const double lub = 1000.0, const double eps_elim = 1e-12, const double eps_relax_abs = 1e-8, const double eps_relax_rel = 0.0, const double max_dyn = 1e8, const double max_dyn_lub = 1e13, const double eps_coeff_lub = 1e-13, const double min_viol = 1e-7, const int use_int_slacks = 0, const int use_cg2 = 0, const double norm_zero = 1e-5, const double min_reduc = 0.05, const double away = 0.05, const double max_tab = 1e7 )`

Constructor from [CglParam](#).

### 6.35.2.3 CglRedSplitParam::CglRedSplitParam ( const CglRedSplitParam & source )

Copy constructor.

### 6.35.2.4 virtual CglRedSplitParam::~~CglRedSplitParam ( ) [virtual]

Destructor.

## 6.35.3 Member Function Documentation

### 6.35.3.1 virtual void CglRedSplitParam::setAway ( const double value ) [virtual]

Set away, the minimum distance from being integer used for selecting rows for cut generation; all rows whose pivot variable should be integer but is more than away from integrality will be selected; Default: 0.05.

### 6.35.3.2 double CglRedSplitParam::getAway ( ) const [inline]

Get value of away.

Definition at line 73 of file CglRedSplitParam.hpp.

### 6.35.3.3 virtual void CglRedSplitParam::setLUB ( const double value ) [virtual]

Set the value of LUB, value considered large for the absolute value of a lower or upper bound on a variable; Default: 1000.

### 6.35.3.4 double CglRedSplitParam::getLUB ( ) const [inline]

Get the value of LUB.

Definition at line 80 of file CglRedSplitParam.hpp.

### 6.35.3.5 void CglRedSplitParam::setEPS\_ELIM ( const double value )

Set the value of EPS\_ELIM, epsilon for values of coefficients when eliminating slack variables; Default: 1e-12.

### 6.35.3.6 double CglRedSplitParam::getEPS\_ELIM ( ) const [inline]

Get the value of EPS\_ELIM.

Definition at line 87 of file CglRedSplitParam.hpp.

### 6.35.3.7 virtual void CglRedSplitParam::setEPS\_RELAX\_ABS ( const double eps\_ra ) [virtual]

Set EPS\_RELAX\_ABS.

### 6.35.3.8 double CglRedSplitParam::getEPS\_RELAX\_ABS ( ) const [inline]

Get value of EPS\_RELAX\_ABS.

Definition at line 92 of file CglRedSplitParam.hpp.

### 6.35.3.9 virtual void CglRedSplitParam::setEPS\_RELAX\_REL ( const double eps\_rr ) [virtual]

Set EPS\_RELAX\_REL.

6.35.3.10 `double CglRedSplitParam::getEPS_RELAX_REL ( ) const [inline]`

Get value of EPS\_RELAX\_REL.

Definition at line 97 of file CglRedSplitParam.hpp.

6.35.3.11 `virtual void CglRedSplitParam::setMaxDYN ( double value ) [virtual]`

6.35.3.12 `double CglRedSplitParam::getMAXDYN ( ) const [inline]`

Get the value of MAXDYN.

Definition at line 103 of file CglRedSplitParam.hpp.

6.35.3.13 `virtual void CglRedSplitParam::setMaxDYN_LUB ( double value ) [virtual]`

6.35.3.14 `double CglRedSplitParam::getMAXDYN_LUB ( ) const [inline]`

Get the value of MAXDYN\_LUB.

Definition at line 111 of file CglRedSplitParam.hpp.

6.35.3.15 `virtual void CglRedSplitParam::setEPS_COEFF_LUB ( const double value ) [virtual]`

Set the value of EPS\_COEFF\_LUB, epsilon for values of coefficients for variables with absolute value of lower or upper bound larger than LUB; Default: 1e-13.

6.35.3.16 `double CglRedSplitParam::getEPS_COEFF_LUB ( ) const [inline]`

Get the value of EPS\_COEFF\_LUB.

Definition at line 118 of file CglRedSplitParam.hpp.

6.35.3.17 `virtual void CglRedSplitParam::setMINVIOL ( double value ) [virtual]`

Set the value of MINVIOL, the minimum violation for the current basic solution in a generated cut.

Default: 1e-7

6.35.3.18 `double CglRedSplitParam::getMINVIOL ( ) const [inline]`

Get the value of MINVIOL.

Definition at line 124 of file CglRedSplitParam.hpp.

6.35.3.19 `virtual void CglRedSplitParam::setUSE_INTSLACKS ( int value ) [virtual]`

Set the value of USE\_INTSLACKS.

Default: 0

6.35.3.20 `int CglRedSplitParam::getUSE_INTSLACKS ( ) const [inline]`

Get the value of USE\_INTSLACKS.

Definition at line 129 of file CglRedSplitParam.hpp.

6.35.3.21 `virtual void CglRedSplitParam::setUSE_CG2 ( int value ) [virtual]`

Set the value of USE\_CG2.

Default: 0

**6.35.3.22** `int CglRedSplitParam::getUSE_CG2 ( ) const [inline]`

Get the value of USE\_CG2.

Definition at line 134 of file CglRedSplitParam.hpp.

**6.35.3.23** `virtual void CglRedSplitParam::setNormIsZero ( const double value ) [virtual]`

Set the value of normIsZero, the threshold for considering a norm to be 0; Default: 1e-5.

**6.35.3.24** `double CglRedSplitParam::getNormIsZero ( ) const [inline]`

Get the value of normIsZero.

Definition at line 140 of file CglRedSplitParam.hpp.

**6.35.3.25** `virtual void CglRedSplitParam::setMinReduc ( const double value ) [virtual]`

Set the value of minReduc, threshold for relative norm improvement for performing a reduction; Default: 0.05.

**6.35.3.26** `double CglRedSplitParam::getMinReduc ( ) const [inline]`

Get the value of minReduc.

Definition at line 146 of file CglRedSplitParam.hpp.

**6.35.3.27** `virtual void CglRedSplitParam::setMaxTab ( const double value ) [virtual]`

Set the maximum allowed value for  $(mTab * mTab * CoinMax(mTab, nTab))$  where  $mTab$  is the number of rows used in the combinations and  $nTab$  is the number of continuous non basic variables.

The work of the generator is proportional to  $(mTab * mTab * CoinMax(mTab, nTab))$ . Reducing the value of  $maxTab$  makes the generator faster, but weaker. Default: 1e7.

**6.35.3.28** `double CglRedSplitParam::getMaxTab ( ) const [inline]`

Get the value of  $maxTab$ .

Definition at line 155 of file CglRedSplitParam.hpp.

**6.35.3.29** `virtual CglRedSplitParam* CglRedSplitParam::clone ( ) const [virtual]`

Clone.

Reimplemented from [CglParam](#).

**6.35.3.30** `virtual CglRedSplitParam& CglRedSplitParam::operator= ( const CglRedSplitParam & rhs ) [virtual]`

Assignment operator.

## 6.35.4 Member Data Documentation

**6.35.4.1** `double CglRedSplitParam::LUB [protected]`

Value considered large for the absolute value of lower or upper bound on a variable.

Default: 1000.

Definition at line 213 of file CglRedSplitParam.hpp.

**6.35.4.2 double CglRedSplitParam::EPS\_ELIM [protected]**

Epsilon for value of coefficients when eliminating slack variables.

Default: 1e-12.

Definition at line 217 of file CglRedSplitParam.hpp.

**6.35.4.3 double CglRedSplitParam::EPS\_RELAX\_ABS [protected]**

Value added to the right hand side of each generated cut to relax it.

Default: 1e-8

Definition at line 221 of file CglRedSplitParam.hpp.

**6.35.4.4 double CglRedSplitParam::EPS\_RELAX\_REL [protected]**

For a generated cut with right hand side rhs\_val, EPS\_RELAX\_EPS \* fabs(rhs\_val) is used to relax the constraint.

Default: 0

Definition at line 226 of file CglRedSplitParam.hpp.

**6.35.4.5 double CglRedSplitParam::MAXDYN [protected]**

Definition at line 230 of file CglRedSplitParam.hpp.

**6.35.4.6 double CglRedSplitParam::MAXDYN\_LUB [protected]**

Definition at line 236 of file CglRedSplitParam.hpp.

**6.35.4.7 double CglRedSplitParam::EPS\_COEFF\_LUB [protected]**

Epsilon for value of coefficients for variables with absolute value of lower or upper bound larger than LUB.

Default: 1e-13.

Definition at line 240 of file CglRedSplitParam.hpp.

**6.35.4.8 double CglRedSplitParam::MINVIOL [protected]**

Minimum violation for the current basic solution in a generated cut.

Default: 1e-7.

Definition at line 244 of file CglRedSplitParam.hpp.

**6.35.4.9 int CglRedSplitParam::USE\_INTSLACKS [protected]**

Use integer slacks to generate cuts if USE\_INTSLACKS = 1. Default: 0.

Definition at line 247 of file CglRedSplitParam.hpp.

**6.35.4.10 int CglRedSplitParam::USE\_CG2 [protected]**

Use second way to generate a mixed integer Gomory cut (see methods generate\_cgcut()) and generate\_cgcut\_2()).

Default: 0.

Definition at line 251 of file CglRedSplitParam.hpp.



#### 6.35.4.11 double CglRedSplitParam::normIsZero [protected]

Norm of a vector is considered zero if smaller than normIsZero; Default: 1e-5.

Definition at line 255 of file CglRedSplitParam.hpp.

#### 6.35.4.12 double CglRedSplitParam::minReduc [protected]

Minimum reduction in percent that must be achieved by a potential reduction step in order to be performed; Between 0 and 1, default: 0.05.

Definition at line 259 of file CglRedSplitParam.hpp.

#### 6.35.4.13 double CglRedSplitParam::away\_ [protected]

Use row only if pivot variable should be integer but is more than away\_ from being integer.

Definition at line 263 of file CglRedSplitParam.hpp.

#### 6.35.4.14 double CglRedSplitParam::maxTab\_ [protected]

Maximum value for (mTab \* mTab \* CoinMax(mTab, nTab)).

See method [setMaxTab\(\)](#).

Definition at line 267 of file CglRedSplitParam.hpp.

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

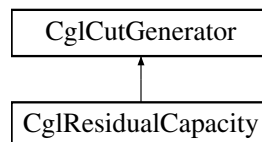
- src/CglRedSplit/[CglRedSplitParam.hpp](#)

## 6.36 CglResidualCapacity Class Reference

Residual Capacity Inequalities Cut Generator Class.

```
#include <CglResidualCapacity.hpp>
```

Inheritance diagram for CglResidualCapacity:



### Public Member Functions

#### Get and Set Parameters

- void [setEpsilon](#) (double value)  
*Set Epsilon.*
- double [getEpsilon](#) () const  
*Get Epsilon.*
- void [setTolerance](#) (double value)  
*Set Tolerance.*
- double [getTolerance](#) () const  
*Get Tolerance.*

- void [setDoPreproc](#) (int value)  
*Set doPreproc.*
- bool [getDoPreproc](#) () const  
*Get doPreproc.*

### Generate Cuts

- virtual void [generateCuts](#) (const OsiSolverInterface &si, OsiCuts &cs, const [CglTreeInfo](#) info=[CglTreeInfo](#)())  
*Generate Residual Capacity cuts for the model data contained in si.*

### Constructors and destructors

- [CglResidualCapacity](#) ()  
*Default constructor.*
- [CglResidualCapacity](#) (const double tolerance)  
*Alternate Constructor.*
- [CglResidualCapacity](#) (const [CglResidualCapacity](#) &)  
*Copy constructor.*
- virtual [CglCutGenerator](#) \* [clone](#) () const  
*Clone.*
- [CglResidualCapacity](#) & [operator=](#) (const [CglResidualCapacity](#) &rhs)  
*Assignment operator.*
- virtual [~CglResidualCapacity](#) ()  
*Destructor.*
- virtual void [refreshPrep](#) ()  
*This is to refresh preprocessing.*

### Friends

- void [CglResidualCapacityUnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)  
*A function that tests the methods in the [CglResidualCapacity](#) class.*

### Additional Inherited Members

#### 6.36.1 Detailed Description

Residual Capacity Inequalities Cut Generator Class.

References: T Magnanti, P Mirchandani, R Vachani, "The convex hull of two core capacitated network design problems," Math Programming 60 (1993), 233-250.

A Atamturk, D Rajan, "On splittable and unsplittable flow capacitated network design arc-set polyhedra," Math Programming 92 (2002), 315-333.

Definition at line 47 of file CglResidualCapacity.hpp.

#### 6.36.2 Constructor & Destructor Documentation

##### 6.36.2.1 CglResidualCapacity::CglResidualCapacity ( )

Default constructor.

### 6.36.2.2 CglResidualCapacity::CglResidualCapacity ( const double *tolerance* )

Alternate Constructor.

### 6.36.2.3 CglResidualCapacity::CglResidualCapacity ( const CglResidualCapacity & )

Copy constructor.

### 6.36.2.4 virtual CglResidualCapacity::~CglResidualCapacity ( ) [virtual]

Destructor.

## 6.36.3 Member Function Documentation

### 6.36.3.1 void CglResidualCapacity::setEpsilon ( double *value* )

Set Epsilon.

### 6.36.3.2 double CglResidualCapacity::getEpsilon ( ) const

Get Epsilon.

### 6.36.3.3 void CglResidualCapacity::setTolerance ( double *value* )

Set Tolerance.

### 6.36.3.4 double CglResidualCapacity::getTolerance ( ) const

Get Tolerance.

### 6.36.3.5 void CglResidualCapacity::setDoPreproc ( int *value* )

Set doPreproc.

### 6.36.3.6 bool CglResidualCapacity::getDoPreproc ( ) const

Get doPreproc.

### 6.36.3.7 virtual void CglResidualCapacity::generateCuts ( const OsiSolverInterface & *si*, OsiCuts & *cs*, const CglTreeInfo *info* = CglTreeInfo ( ) ) [virtual]

Generate Residual Capacity cuts for the model data contained in *si*.

The generated cuts are inserted in the collection of cuts *cs*.

Implements [CglCutGenerator](#).

### 6.36.3.8 virtual CglCutGenerator\* CglResidualCapacity::clone ( ) const [virtual]

Clone.

Implements [CglCutGenerator](#).

### 6.36.3.9 CglResidualCapacity& CglResidualCapacity::operator= ( const CglResidualCapacity & *rhs* )

Assignment operator.

6.36.3.10 `virtual void CglResidualCapacity::refreshPrep ( ) [virtual]`

This is to refresh preprocessing.

#### 6.36.4 Friends And Related Function Documentation

6.36.4.1 `void CglResidualCapacityUnitTest ( const OsiSolverInterface * siP, const std::string mpdDir ) [friend]`

A function that tests the methods in the [CglResidualCapacity](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

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

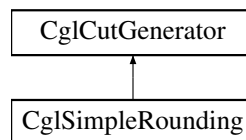
- `src/CglResidualCapacity/CglResidualCapacity.hpp`

## 6.37 CglSimpleRounding Class Reference

Simple Rounding Cut Generator Class.

```
#include <CglSimpleRounding.hpp>
```

Inheritance diagram for CglSimpleRounding:



### Public Member Functions

#### Generate Cuts

- `virtual void generateCuts (const OsiSolverInterface &si, OsiCuts &cs, const CglTreeInfo info=CglTreeInfo())`  
*Generate simple rounding cuts for the model accessed through the solver interface.*

#### Constructors and destructors

- `CglSimpleRounding ()`  
*Default constructor.*
- `CglSimpleRounding (const CglSimpleRounding &)`  
*Copy constructor.*
- `virtual CglCutGenerator * clone () const`  
*Clone.*
- `CglSimpleRounding & operator= (const CglSimpleRounding &rhs)`  
*Assignment operator.*
- `virtual ~CglSimpleRounding ()`  
*Destructor.*
- `virtual std::string generateCpp (FILE *fp)`  
*Create C++ lines to get to current state.*

## Friends

- void [CglSimpleRoundingUnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)

*A function that tests the methods in the [CglSimpleRounding](#) class.*

## Additional Inherited Members

### 6.37.1 Detailed Description

Simple Rounding Cut Generator Class.

This class generates simple rounding cuts via the following method: For each constraint, attempt to derive a  $\leq$  inequality in all integer variables by netting out any continuous variables. Divide the resulting integer inequality through by the greatest common denominator (gcd) of the lhs coefficients. Round down the rhs.

Warning: Use with careful attention to data precision.

(Reference: Nemhauser and Wolsey, Integer and Combinatorial Optimization, 1988, pg 211.)

Definition at line 29 of file CglSimpleRounding.hpp.

### 6.37.2 Constructor & Destructor Documentation

#### 6.37.2.1 CglSimpleRounding::CglSimpleRounding ( )

Default constructor.

#### 6.37.2.2 CglSimpleRounding::CglSimpleRounding ( const CglSimpleRounding & )

Copy constructor.

#### 6.37.2.3 virtual CglSimpleRounding::~~CglSimpleRounding ( ) [virtual]

Destructor.

### 6.37.3 Member Function Documentation

#### 6.37.3.1 virtual void CglSimpleRounding::generateCuts ( const OsiSolverInterface & si, OsiCuts & cs, const CglTreeInfo info = CglTreeInfo ( ) ) [virtual]

Generate simple rounding cuts for the model accessed through the solver interface.

Insert generated cuts into the cut set cs.

Implements [CglCutGenerator](#).

#### 6.37.3.2 virtual CglCutGenerator\* CglSimpleRounding::clone ( ) const [virtual]

Clone.

Implements [CglCutGenerator](#).

#### 6.37.3.3 CglSimpleRounding& CglSimpleRounding::operator= ( const CglSimpleRounding & rhs )

Assignment operator.

6.37.3.4 `virtual std::string CglSimpleRounding::generateCpp ( FILE * fp ) [virtual]`

Create C++ lines to get to current state.

Reimplemented from [CglCutGenerator](#).

#### 6.37.4 Friends And Related Function Documentation

6.37.4.1 `void CglSimpleRoundingUnitTest ( const OsiSolverInterface * siP, const std::string mpdDir ) [friend]`

A function that tests the methods in the [CglSimpleRounding](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

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

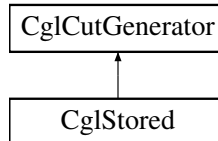
- [src/CglSimpleRounding/CglSimpleRounding.hpp](#)

## 6.38 CglStored Class Reference

Stored Cut Generator Class.

```
#include <CglStored.hpp>
```

Inheritance diagram for CglStored:



### Public Member Functions

#### Generate Cuts

- `virtual void generateCuts (const OsiSolverInterface &si, OsiCuts &cs, const CglTreeInfo info=CglTreeInfo())`  
*Generate Mixed Integer Stored cuts for the model of the solver interface, si.*

#### Change criterion on whether to include cut.

*Violations of more than this will be added to current cut list (default 1.0e-5)*

- `void setRequiredViolation (double value)`  
*Set.*
- `double getRequiredViolation () const`  
*Get.*
- `void setProbingInfo (CglTreeProbingInfo *info)`  
*Takes over ownership of probing info.*

#### Cut stuff

- `void addCut (const OsiCuts &cs)`

- *Add cuts.*
- void [addCut](#) (const OsiRowCut &[cut](#))
- *Add a row cut.*
- void [addCut](#) (double lb, double ub, const CoinPackedVector &vector)
- *Add a row cut from a packed vector.*
- void [addCut](#) (double lb, double ub, int size, const int \*colIndices, const double \*elements)
- *Add a row cut from elements.*
- int [sizeRowCuts](#) () const
- const OsiRowCut \* [rowCutPointer](#) (int index) const
- void [saveStuff](#) (double [bestObjective](#), const double \*[bestSolution](#), const double \*lower, const double \*upper)
- *Save stuff.*
- const double \* [bestSolution](#) () const
- *Best solution (or NULL)*
- double [bestObjective](#) () const
- *Best objective.*
- const double \* [tightLower](#) () const
- *Tight lower bounds.*
- const double \* [tightUpper](#) () const
- *Tight upper bounds.*

## Constructors and destructors

- [CglStored](#) (int numberColumns=0)
- *Default constructor.*
- [CglStored](#) (const [CglStored](#) &rhs)
- *Copy constructor.*
- [CglStored](#) (const char \*fileName)
- *Constructor from file.*
- virtual [CglCutGenerator](#) \* [clone](#) () const
- *Clone.*
- [CglStored](#) & [operator=](#) (const [CglStored](#) &rhs)
- *Assignment operator.*
- virtual [~CglStored](#) ()
- *Destructor.*

## Protected Attributes

### Protected member data

- double [requiredViolation\\_](#)
- *Only add if more than this requiredViolation.*
- [CglTreeProbingInfo](#) \* [probingInfo\\_](#)
- *Pointer to probing information.*
- OsiCuts [cuts\\_](#)
- *Cuts.*
- int [numberColumns\\_](#)
- *Number of columns in model.*
- double \* [bestSolution\\_](#)
- *Best solution (objective at end)*
- double \* [bounds\\_](#)
- *Tight bounds.*

## Additional Inherited Members

### 6.38.1 Detailed Description

Stored Cut Generator Class.

Definition at line 16 of file CglStored.hpp.

### 6.38.2 Constructor & Destructor Documentation

#### 6.38.2.1 CglStored::CglStored ( int *numberOfColumns* = 0 )

Default constructor.

#### 6.38.2.2 CglStored::CglStored ( const CglStored & *rhs* )

Copy constructor.

#### 6.38.2.3 CglStored::CglStored ( const char \* *fileName* )

Constructor from file.

#### 6.38.2.4 virtual CglStored::~~CglStored ( ) [virtual]

Destructor.

### 6.38.3 Member Function Documentation

#### 6.38.3.1 virtual void CglStored::generateCuts ( const OsiSolverInterface & *si*, OsiCuts & *cs*, const CglTreeInfo *info* = CglTreeInfo() ) [virtual]

Generate Mixed Integer Stored cuts for the model of the solver interface, *si*.

Insert the generated cuts into OsiCut, *cs*.

This generator just looks at previously stored cuts and inserts any that are violated by enough

Implements [CglCutGenerator](#).

#### 6.38.3.2 void CglStored::setRequiredViolation ( double *value* ) [inline]

Set.

Definition at line 40 of file CglStored.hpp.

#### 6.38.3.3 double CglStored::getRequiredViolation ( ) const [inline]

Get.

Definition at line 43 of file CglStored.hpp.

#### 6.38.3.4 void CglStored::setProbingInfo ( CglTreeProbingInfo \* *info* ) [inline]

Takes over ownership of probing info.

Definition at line 46 of file CglStored.hpp.



6.38.3.5 `void CglStored::addCut ( const OsiCuts & cs )`

Add cuts.

6.38.3.6 `void CglStored::addCut ( const OsiRowCut & cut )`

Add a row cut.

6.38.3.7 `void CglStored::addCut ( double lb, double ub, const CoinPackedVector & vector )`

Add a row cut from a packed vector.

6.38.3.8 `void CglStored::addCut ( double lb, double ub, int size, const int * colIndices, const double * elements )`

Add a row cut from elements.

6.38.3.9 `int CglStored::sizeRowCuts ( ) const [inline]`

Definition at line 60 of file CglStored.hpp.

6.38.3.10 `const OsiRowCut* CglStored::rowCutPointer ( int index ) const [inline]`

Definition at line 62 of file CglStored.hpp.

6.38.3.11 `void CglStored::saveStuff ( double bestObjective, const double * bestSolution, const double * lower, const double * upper )`

Save stuff.

6.38.3.12 `const double* CglStored::bestSolution ( ) const [inline]`

Best solution (or NULL)

Definition at line 68 of file CglStored.hpp.

6.38.3.13 `double CglStored::bestObjective ( ) const`

Best objective.

6.38.3.14 `const double* CglStored::tightLower ( ) const [inline]`

Tight lower bounds.

Definition at line 73 of file CglStored.hpp.

6.38.3.15 `const double* CglStored::tightUpper ( ) const [inline]`

Tight upper bounds.

Definition at line 76 of file CglStored.hpp.

6.38.3.16 `virtual CglCutGenerator* CglStored::clone ( ) const [virtual]`

Clone.

Implements [CglCutGenerator](#).

6.38.3.17 `CglStored& CglStored::operator= ( const CglStored & rhs )`

Assignment operator.

#### 6.38.4 Member Data Documentation

##### 6.38.4.1 `double CglStored::requiredViolation_` [protected]

Only add if more than this requiredViolation.

Definition at line 112 of file CglStored.hpp.

##### 6.38.4.2 `CglTreeProbingInfo* CglStored::probingInfo_` [protected]

Pointer to probing information.

Definition at line 114 of file CglStored.hpp.

##### 6.38.4.3 `OsiCuts CglStored::cuts_` [protected]

Cuts.

Definition at line 116 of file CglStored.hpp.

##### 6.38.4.4 `int CglStored::numberOfColumns_` [protected]

Number of columns in model.

Definition at line 118 of file CglStored.hpp.

##### 6.38.4.5 `double* CglStored::bestSolution_` [protected]

Best solution (objective at end)

Definition at line 120 of file CglStored.hpp.

##### 6.38.4.6 `double* CglStored::bounds_` [protected]

Tight bounds.

Definition at line 122 of file CglStored.hpp.

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

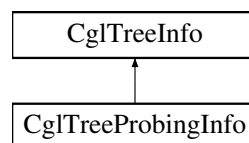
- [src/CglStored.hpp](#)

## 6.39 CglTreeInfo Class Reference

Information about where the cut generator is invoked from.

```
#include <CglTreeInfo.hpp>
```

Inheritance diagram for CglTreeInfo:



## Public Member Functions

- [CglTreeInfo](#) ()  
*Default constructor.*
- [CglTreeInfo](#) (const [CglTreeInfo](#) &)  
*Copy constructor.*
- virtual [CglTreeInfo](#) \* [clone](#) () const  
*Clone.*
- [CglTreeInfo](#) & [operator=](#) (const [CglTreeInfo](#) &rhs)  
*Assignment operator.*
- virtual [~CglTreeInfo](#) ()  
*Destructor.*
- virtual bool [fixes](#) (int, int, int, bool)  
*Take action if cut generator can fix a variable (toValue -1 for down, +1 for up)*
- virtual int [initializeFixing](#) (const OsiSolverInterface \*)  
*Initializes fixing arrays etc - returns >0 if we want to save info 0 if we don't and -1 if is to be used.*

## Public Attributes

- int [level](#)  
*The level of the search tree node.*
- int [pass](#)  
*How many times the cut generator was already invoked in this search tree node.*
- int [formulation\\_rows](#)  
*The number of rows in the original formulation.*
- int [options](#)  
*Options 1 - treat costed integers as important 2 - switch off some stuff as variables semi-integer 4 - set global cut flag if at root node 8 - set global cut flag if at root node and first pass 16 - set global cut flag and make cuts globally valid 32 - last round of cuts did nothing - maybe be more aggressive 64 - in preprocessing stage 128 - looks like solution 256 - want alternate cuts 512 - in sub tree (i.e.*
- bool [inTree](#)  
*Set true if in tree (to avoid ambiguity at first branch)*
- OsiRowCut \*\* [strengthenRow](#)  
*Replacement array.*
- CoinThreadRandom \* [randomNumberGenerator](#)  
*Optional pointer to thread specific random number generator.*

### 6.39.1 Detailed Description

Information about where the cut generator is invoked from.

Definition at line 15 of file [CglTreeInfo.hpp](#).

### 6.39.2 Constructor & Destructor Documentation

#### 6.39.2.1 [CglTreeInfo::CglTreeInfo](#) ( )

Default constructor.

### 6.39.2.2 CglTreeInfo::CglTreeInfo ( const CglTreeInfo & )

Copy constructor.

### 6.39.2.3 virtual CglTreeInfo::~~CglTreeInfo ( ) [virtual]

Destructor.

## 6.39.3 Member Function Documentation

### 6.39.3.1 virtual CglTreeInfo\* CglTreeInfo::clone ( ) const [virtual]

Clone.

Reimplemented in [CglTreeProbingInfo](#).

### 6.39.3.2 CglTreeInfo& CglTreeInfo::operator= ( const CglTreeInfo & rhs )

Assignment operator.

### 6.39.3.3 virtual bool CglTreeInfo::fixes ( int , int , int , bool ) [inline],[virtual]

Take action if cut generator can fix a variable (toValue -1 for down, +1 for up)

Reimplemented in [CglTreeProbingInfo](#).

Definition at line 71 of file CglTreeInfo.hpp.

### 6.39.3.4 virtual int CglTreeInfo::initializeFixing ( const OsiSolverInterface \* ) [inline],[virtual]

Initializes fixing arrays etc - returns >0 if we want to save info 0 if we don't and -1 if is to be used.

Reimplemented in [CglTreeProbingInfo](#).

Definition at line 74 of file CglTreeInfo.hpp.

## 6.39.4 Member Data Documentation

### 6.39.4.1 int CglTreeInfo::level

The level of the search tree node.

Definition at line 18 of file CglTreeInfo.hpp.

### 6.39.4.2 int CglTreeInfo::pass

How many times the cut generator was already invoked in this search tree node.

Definition at line 21 of file CglTreeInfo.hpp.

### 6.39.4.3 int CglTreeInfo::formulation\_rows

The number of rows in the original formulation.

Some generators may not want to consider already generated rows when generating new ones.

Definition at line 24 of file CglTreeInfo.hpp.

#### 6.39.4.4 int CglTreeInfo::options

Options 1 - treat costed integers as important 2 - switch off some stuff as variables semi-integer 4 - set global cut flag if at root node 8 - set global cut flag if at root node and first pass 16 - set global cut flag and make cuts globally valid 32 - last round of cuts did nothing - maybe be more aggressive 64 - in preprocessing stage 128 - looks like solution 256 - want alternate cuts 512 - in sub tree (i.e.

parent model) 1024 - in must call again mode or after everything mode

Definition at line 38 of file CglTreeInfo.hpp.

#### 6.39.4.5 bool CglTreeInfo::inTree

Set true if in tree (to avoid ambiguity at first branch)

Definition at line 40 of file CglTreeInfo.hpp.

#### 6.39.4.6 OsiRowCut\*\* CglTreeInfo::strengthenRow

Replacement array.

Before Branch and Cut it may be beneficial to strengthen rows rather than adding cuts. If this array is not NULL then the cut generator can place a pointer to the stronger cut in this array which is number of rows in size.

A null (i.e. zero elements and free rhs) cut indicates that the row is useless and can be removed.

The calling function can then replace those rows.

Definition at line 50 of file CglTreeInfo.hpp.

#### 6.39.4.7 CoinThreadRandom\* CglTreeInfo::randomNumberGenerator

Optional pointer to thread specific random number generator.

Definition at line 52 of file CglTreeInfo.hpp.

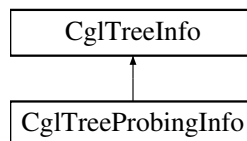
The documentation for this class was generated from the following file:

- [src/CglTreeInfo.hpp](#)

## 6.40 CglTreeProbingInfo Class Reference

```
#include <CglTreeInfo.hpp>
```

Inheritance diagram for CglTreeProbingInfo:



### Public Member Functions

- [CglTreeProbingInfo \(\)](#)  
*Default constructor.*
- [CglTreeProbingInfo \(const OsiSolverInterface \\*model\)](#)  
*Constructor from model.*

- [CglTreeProbingInfo](#) (const [CglTreeProbingInfo](#) &)  
*Copy constructor.*
- virtual [CglTreeInfo](#) \* [clone](#) () const  
*Clone.*
- [CglTreeProbingInfo](#) & [operator=](#) (const [CglTreeProbingInfo](#) &rhs)  
*Assignment operator.*
- virtual [~CglTreeProbingInfo](#) ()  
*Destructor.*
- [OsiSolverInterface](#) \* [analyze](#) (const [OsiSolverInterface](#) &si, int createSolver=0)
- virtual bool [fixes](#) (int variable, int toValue, int fixedVariable, bool fixedToLower)  
*Take action if cut generator can fix a variable (toValue -1 for down, +1 for up) Returns true if still room, false if not.*
- virtual int [initializeFixing](#) (const [OsiSolverInterface](#) \*model)  
*Initializes fixing arrays etc - returns >0 if we want to save info 0 if we don't and -1 if is to be used.*
- int [fixColumns](#) ([OsiSolverInterface](#) &si) const  
*Fix entries in a solver using implications.*
- int [fixColumns](#) (int iColumn, int value, [OsiSolverInterface](#) &si) const  
*Fix entries in a solver using implications for one variable.*
- int [packDown](#) ()  
*Packs down entries.*
- void [generateCuts](#) (const [OsiSolverInterface](#) &si, [OsiCuts](#) &cs, const [CglTreeInfo](#) info) const  
*Generate cuts from implications.*
- [cliqueEntry](#) \* [fixEntries](#) ()  
*Entries for fixing variables.*
- int \* [toZero](#) ()  
*Starts of integer variable going to zero.*
- int \* [toOne](#) ()  
*Starts of integer variable going to one.*
- int \* [integerVariable](#) () const  
*List of 0-1 integer variables.*
- int \* [backward](#) () const  
*Backward look up.*
- int [numberVariables](#) () const  
*Number of variables.*
- int [numberIntegers](#) () const  
*Number of 0-1 variables.*

#### Protected Attributes

- [cliqueEntry](#) \* [fixEntry\\_](#)  
*Entries for fixing variables.*
- int \* [toZero\\_](#)  
*Starts of integer variable going to zero.*
- int \* [toOne\\_](#)  
*Starts of integer variable going to one.*
- int \* [integerVariable\\_](#)  
*List of 0-1 integer variables.*
- int \* [backward\\_](#)

- *Backward look up.*
- `int * fixingEntry_`  
*Entries for fixing variable when collecting.*
- `int numberVariables_`  
*Number of variables.*
- `int numberIntegers_`  
*Number of 0-1 variables.*
- `int maximumEntries_`  
*Maximum number in fixEntry\_.*
- `int numberEntries_`  
*Number entries in fixingEntry\_ (and fixEntry\_) or -2 if correct style.*

## Additional Inherited Members

### 6.40.1 Detailed Description

Definition at line 85 of file CglTreeInfo.hpp.

### 6.40.2 Constructor & Destructor Documentation

#### 6.40.2.1 CglTreeProbingInfo::CglTreeProbingInfo ( )

Default constructor.

#### 6.40.2.2 CglTreeProbingInfo::CglTreeProbingInfo ( const OsiSolverInterface \* model )

Constructor from model.

#### 6.40.2.3 CglTreeProbingInfo::CglTreeProbingInfo ( const CglTreeProbingInfo & )

Copy constructor.

#### 6.40.2.4 virtual CglTreeProbingInfo::~CglTreeProbingInfo ( ) [virtual]

Destructor.

### 6.40.3 Member Function Documentation

#### 6.40.3.1 virtual CglTreeInfo\* CglTreeProbingInfo::clone ( ) const [virtual]

Clone.

Reimplemented from [CglTreeInfo](#).

#### 6.40.3.2 CglTreeProbingInfo& CglTreeProbingInfo::operator= ( const CglTreeProbingInfo & rhs )

Assignment operator.

#### 6.40.3.3 OsiSolverInterface\* CglTreeProbingInfo::analyze ( const OsiSolverInterface & si, int createSolver = 0 )

#### 6.40.3.4 virtual bool CglTreeProbingInfo::fixes ( int variable, int toValue, int fixedVariable, bool fixedToLower ) [virtual]

Take action if cut generator can fix a variable (toValue -1 for down, +1 for up) Returns true if still room, false if not.

Reimplemented from [CglTreeInfo](#).

**6.40.3.5** `virtual int CglTreeProbingInfo::initializeFixing ( const OsiSolverInterface * model ) [virtual]`

Initializes fixing arrays etc - returns >0 if we want to save info 0 if we don't and -1 if is to be used.

Reimplemented from [CglTreeInfo](#).

**6.40.3.6** `int CglTreeProbingInfo::fixColumns ( OsiSolverInterface & si ) const`

Fix entries in a solver using implications.

**6.40.3.7** `int CglTreeProbingInfo::fixColumns ( int iColumn, int value, OsiSolverInterface & si ) const`

Fix entries in a solver using implications for one variable.

**6.40.3.8** `int CglTreeProbingInfo::packDown ( )`

Packs down entries.

**6.40.3.9** `void CglTreeProbingInfo::generateCuts ( const OsiSolverInterface & si, OsiCuts & cs, const CglTreeInfo info ) const`

Generate cuts from implications.

**6.40.3.10** `cliqueEntry* CglTreeProbingInfo::fixEntries ( ) [inline]`

Entries for fixing variables.

Definition at line 124 of file [CglTreeInfo.hpp](#).

**6.40.3.11** `int* CglTreeProbingInfo::toZero ( ) [inline]`

Starts of integer variable going to zero.

Definition at line 127 of file [CglTreeInfo.hpp](#).

**6.40.3.12** `int* CglTreeProbingInfo::toOne ( ) [inline]`

Starts of integer variable going to one.

Definition at line 130 of file [CglTreeInfo.hpp](#).

**6.40.3.13** `int* CglTreeProbingInfo::integerVariable ( ) const [inline]`

List of 0-1 integer variables.

Definition at line 133 of file [CglTreeInfo.hpp](#).

**6.40.3.14** `int* CglTreeProbingInfo::backward ( ) const [inline]`

Backward look up.

Definition at line 136 of file [CglTreeInfo.hpp](#).

**6.40.3.15** `int CglTreeProbingInfo::numberVariables ( ) const [inline]`

Number of variables.

Definition at line 139 of file [CglTreeInfo.hpp](#).



**6.40.3.16** `int CglTreeProbingInfo::numberIntegers ( ) const [inline]`

Number of 0-1 variables.

Definition at line 142 of file CglTreeInfo.hpp.

#### **6.40.4 Member Data Documentation**

**6.40.4.1** `cliqueEntry* CglTreeProbingInfo::fixEntry_ [protected]`

Entries for fixing variables.

Definition at line 149 of file CglTreeInfo.hpp.

**6.40.4.2** `int* CglTreeProbingInfo::toZero_ [protected]`

Starts of integer variable going to zero.

Definition at line 151 of file CglTreeInfo.hpp.

**6.40.4.3** `int* CglTreeProbingInfo::toOne_ [protected]`

Starts of integer variable going to one.

Definition at line 153 of file CglTreeInfo.hpp.

**6.40.4.4** `int* CglTreeProbingInfo::integerVariable_ [protected]`

List of 0-1 integer variables.

Definition at line 155 of file CglTreeInfo.hpp.

**6.40.4.5** `int* CglTreeProbingInfo::backward_ [protected]`

Backward look up.

Definition at line 157 of file CglTreeInfo.hpp.

**6.40.4.6** `int* CglTreeProbingInfo::fixingEntry_ [protected]`

Entries for fixing variable when collecting.

Definition at line 159 of file CglTreeInfo.hpp.

**6.40.4.7** `int CglTreeProbingInfo::numberVariables_ [protected]`

Number of variables.

Definition at line 161 of file CglTreeInfo.hpp.

**6.40.4.8** `int CglTreeProbingInfo::numberIntegers_ [protected]`

Number of 0-1 variables.

Definition at line 163 of file CglTreeInfo.hpp.

**6.40.4.9** `int CglTreeProbingInfo::maximumEntries_ [protected]`

Maximum number in fixEntry\_.

Definition at line 165 of file CglTreeInfo.hpp.

## 6.40.4.10 int CglTreeProbingInfo::numberEntries\_ [protected]

Number entries in fixingEntry\_ (and fixEntry\_) or -2 if correct style.

Definition at line 167 of file CglTreeInfo.hpp.

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

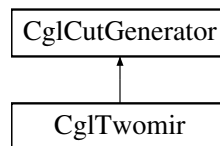
- src/CglTreeInfo.hpp

## 6.41 CglTwomir Class Reference

Twostep MIR Cut Generator Class.

```
#include <CglTwomir.hpp>
```

Inheritance diagram for CglTwomir:



## Public Member Functions

## Generate Cuts

- virtual void [generateCuts](#) (const OsiSolverInterface &si, OsiCuts &cs, const [CglTreeInfo](#) info=[CglTreeInfo](#)())  
*Generate Two step MIR cuts either from the tableau rows or from the formulation rows.*
- virtual bool [needsOptimalBasis](#) () const  
*Return true if needs optimal basis to do cuts (will return true)*

## Change criterion on which scalings to use (default = 1,1,1,1)

- void [setMirScale](#) (int tmin, int tmax)  
*Set.*
- void [setTwomirScale](#) (int qmin, int qmax)
- void [setAMax](#) (int a)
- void [setMaxElements](#) (int n)
- void [setMaxElementsRoot](#) (int n)
- void [setCutTypes](#) (bool mir, bool twomir, bool tab, bool form)
- void [setFormulationRows](#) (int n)
- int [getTmin](#) () const  
*Get.*
- int [getTmax](#) () const
- int [getQmin](#) () const
- int [getQmax](#) () const
- int [getAMax](#) () const
- int [getMaxElements](#) () const
- int [setMaxElementsRoot](#) () const
- int [getIfMir](#) () const
- int [getIfTwomir](#) () const
- int [getIfTableau](#) () const
- int [getIfFormulation](#) () const

### Change criterion on which variables to look at. All ones

*more than "away" away from integrality will be investigated (default 0.05)*

- void `setAway` (double value)  
*Set away.*
- double `getAway` () const  
*Get away.*
- void `setAwayAtRoot` (double value)  
*Set away at root.*
- double `getAwayAtRoot` () const  
*Get away at root.*
- virtual int `maxLengthOfCutInTree` () const  
*Return maximum length of cut in tree.*

### Change way TwoMir works

- void `passInOriginalSolver` (OsiSolverInterface \*solver)  
*Pass in a copy of original solver (clone it)*
- OsiSolverInterface \* `originalSolver` () const  
*Returns original solver.*
- void `setTwomirType` (int type)  
*Set type - 0 normal, 1 add original matrix one, 2 replace.*
- int `twomirType` () const  
*Return type.*

### Constructors and destructors

- `CglTwomir` ()  
*Default constructor.*
- `CglTwomir` (const `CglTwomir` &)  
*Copy constructor.*
- virtual `CglCutGenerator` \* `clone` () const  
*Clone.*
- `CglTwomir` & `operator=` (const `CglTwomir` &rhs)  
*Assignment operator.*
- virtual `~CglTwomir` ()  
*Destructor.*
- virtual std::string `generateCpp` (FILE \*fp)  
*Create C++ lines to get to current state.*
- virtual void `refreshSolver` (OsiSolverInterface \*solver)  
*This can be used to refresh any inforamtion.*

### Public Attributes

- std::string `probnam_`  
*Problem name.*

### Friends

- void `CglTwomirUnitTest` (const OsiSolverInterface \*siP, const std::string mpdDir)  
*A function that tests the methods in the `CglTwomir` class.*

### 6.41.1 Detailed Description

Twostep MIR Cut Generator Class.

Definition at line 91 of file CglTwomir.hpp.

### 6.41.2 Constructor & Destructor Documentation

#### 6.41.2.1 CglTwomir::CglTwomir ( )

Default constructor.

#### 6.41.2.2 CglTwomir::CglTwomir ( const CglTwomir & )

Copy constructor.

#### 6.41.2.3 virtual CglTwomir::~CglTwomir ( ) [virtual]

Destructor.

### 6.41.3 Member Function Documentation

#### 6.41.3.1 virtual void CglTwomir::generateCuts ( const OsiSolverInterface & *si*, OsiCuts & *cs*, const CglTreeInfo *info* = CglTreeInfo ( ) ) [virtual]

Generate Two step MIR cuts either from the tableau rows or from the formulation rows.

Implements [CglCutGenerator](#).

#### 6.41.3.2 virtual bool CglTwomir::needsOptimalBasis ( ) const [virtual]

Return true if needs optimal basis to do cuts (will return true)

Reimplemented from [CglCutGenerator](#).

#### 6.41.3.3 void CglTwomir::setMirScale ( int *tmin*, int *tmax* ) [inline]

Set.

Definition at line 115 of file CglTwomir.hpp.

#### 6.41.3.4 void CglTwomir::setTwomirScale ( int *qmin*, int *qmax* ) [inline]

Definition at line 116 of file CglTwomir.hpp.

#### 6.41.3.5 void CglTwomir::setAMax ( int *a* ) [inline]

Definition at line 117 of file CglTwomir.hpp.

#### 6.41.3.6 void CglTwomir::setMaxElements ( int *n* ) [inline]

Definition at line 118 of file CglTwomir.hpp.

#### 6.41.3.7 void CglTwomir::setMaxElementsRoot ( int *n* ) [inline]

Definition at line 119 of file CglTwomir.hpp.

6.41.3.8 void CglTwomir::setCutTypes ( bool *mir*, bool *twomir*, bool *tab*, bool *form* ) [inline]

Definition at line 120 of file CglTwomir.hpp.

6.41.3.9 void CglTwomir::setFormulationRows ( int *n* ) [inline]

Definition at line 122 of file CglTwomir.hpp.

6.41.3.10 int CglTwomir::getTmin ( ) const [inline]

Get.

Definition at line 125 of file CglTwomir.hpp.

6.41.3.11 int CglTwomir::getTmax ( ) const [inline]

Definition at line 126 of file CglTwomir.hpp.

6.41.3.12 int CglTwomir::getQmin ( ) const [inline]

Definition at line 127 of file CglTwomir.hpp.

6.41.3.13 int CglTwomir::getQmax ( ) const [inline]

Definition at line 128 of file CglTwomir.hpp.

6.41.3.14 int CglTwomir::getAmax ( ) const [inline]

Definition at line 129 of file CglTwomir.hpp.

6.41.3.15 int CglTwomir::getMaxElements ( ) const [inline]

Definition at line 130 of file CglTwomir.hpp.

6.41.3.16 int CglTwomir::getMaxElementsRoot ( ) const [inline]

Definition at line 131 of file CglTwomir.hpp.

6.41.3.17 int CglTwomir::getIfMir ( ) const [inline]

Definition at line 132 of file CglTwomir.hpp.

6.41.3.18 int CglTwomir::getIfTwomir ( ) const [inline]

Definition at line 133 of file CglTwomir.hpp.

6.41.3.19 int CglTwomir::getIfTableau ( ) const [inline]

Definition at line 134 of file CglTwomir.hpp.

6.41.3.20 int CglTwomir::getIfFormulation ( ) const [inline]

Definition at line 135 of file CglTwomir.hpp.

6.41.3.21 void CglTwomir::setAway ( double *value* )

Set away.

6.41.3.22 `double CglTwomir::getAway ( ) const`

Get away.

6.41.3.23 `void CglTwomir::setAwayAtRoot ( double value )`

Set away at root.

6.41.3.24 `double CglTwomir::getAwayAtRoot ( ) const`

Get away at root.

6.41.3.25 `virtual int CglTwomir::maxLengthOfCutInTree ( ) const` `[inline]`, `[virtual]`

Return maximum length of cut in tree.

Reimplemented from [CglCutGenerator](#).

Definition at line 151 of file `CglTwomir.hpp`.

6.41.3.26 `void CglTwomir::passInOriginalSolver ( OsiSolverInterface * solver )`

Pass in a copy of original solver (clone it)

6.41.3.27 `OsiSolverInterface* CglTwomir::originalSolver ( ) const` `[inline]`

Returns original solver.

Definition at line 160 of file `CglTwomir.hpp`.

6.41.3.28 `void CglTwomir::setTwomirType ( int type )` `[inline]`

Set type - 0 normal, 1 add original matrix one, 2 replace.

Definition at line 163 of file `CglTwomir.hpp`.

6.41.3.29 `int CglTwomir::twomirType ( ) const` `[inline]`

Return type.

Definition at line 166 of file `CglTwomir.hpp`.

6.41.3.30 `virtual CglCutGenerator* CglTwomir::clone ( ) const` `[virtual]`

Clone.

Implements [CglCutGenerator](#).

6.41.3.31 `CglTwomir& CglTwomir::operator= ( const CglTwomir & rhs )`

Assignment operator.

6.41.3.32 `virtual std::string CglTwomir::generateCpp ( FILE * fp )` `[virtual]`

Create C++ lines to get to current state.

Reimplemented from [CglCutGenerator](#).

6.41.3.33 `virtual void CglTwomir::refreshSolver ( OsiSolverInterface * solver )` `[virtual]`

This can be used to refresh any inforamtion.

Reimplemented from [CglCutGenerator](#).

#### 6.41.4 Friends And Related Function Documentation

##### 6.41.4.1 void CglTwomirUnitTest ( const OsiSolverInterface \* *siP*, const std::string *mpdDir* ) [friend]

A function that tests the methods in the [CglTwomir](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

#### 6.41.5 Member Data Documentation

##### 6.41.5.1 std::string CglTwomir::probname\_

Problem name.

Definition at line 100 of file [CglTwomir.hpp](#).

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

- [src/CglTwomir/CglTwomir.hpp](#)

## 6.42 CglUniqueRowCuts Class Reference

```
#include <CglPreProcess.hpp>
```

### Public Member Functions

- [CglUniqueRowCuts](#) (int initialMaxSize=0, int hashMultiplier=4)
- [~CglUniqueRowCuts](#) ()
- [CglUniqueRowCuts](#) (const [CglUniqueRowCuts](#) &rhs)
- [CglUniqueRowCuts](#) & [operator=](#) (const [CglUniqueRowCuts](#) &rhs)
- OsiRowCut \* [cut](#) (int sequence) const
- int [numberCuts](#) () const
- int [sizeRowCuts](#) () const
- OsiRowCut \* [rowCutPtr](#) (int sequence)
- void [eraseRowCut](#) (int sequence)
- void [insert](#) (const OsiRowCut &[cut](#))
- int [insertIfNotDuplicate](#) (const OsiRowCut &[cut](#))
- void [addCuts](#) (OsiCuts &cs)

#### 6.42.1 Detailed Description

Definition at line 458 of file [CglPreProcess.hpp](#).

## 6.42.2 Constructor &amp; Destructor Documentation

6.42.2.1 CglUniqueRowCuts::CglUniqueRowCuts ( int *initialMaxSize* = 0, int *hashMultiplier* = 4 )

6.42.2.2 CglUniqueRowCuts::~~CglUniqueRowCuts ( )

6.42.2.3 CglUniqueRowCuts::CglUniqueRowCuts ( const CglUniqueRowCuts & *rhs* )

## 6.42.3 Member Function Documentation

6.42.3.1 CglUniqueRowCuts& CglUniqueRowCuts::operator= ( const CglUniqueRowCuts & *rhs* )6.42.3.2 OsiRowCut\* CglUniqueRowCuts::cut ( int *sequence* ) const [inline]

Definition at line 465 of file CglPreProcess.hpp.

6.42.3.3 int CglUniqueRowCuts::numberCuts ( ) const [inline]

Definition at line 467 of file CglPreProcess.hpp.

6.42.3.4 int CglUniqueRowCuts::sizeRowCuts ( ) const [inline]

Definition at line 469 of file CglPreProcess.hpp.

6.42.3.5 OsiRowCut\* CglUniqueRowCuts::rowCutPtr ( int *sequence* ) [inline]

Definition at line 471 of file CglPreProcess.hpp.

6.42.3.6 void CglUniqueRowCuts::eraseRowCut ( int *sequence* )6.42.3.7 void CglUniqueRowCuts::insert ( const OsiRowCut & *cut* ) [inline]

Definition at line 475 of file CglPreProcess.hpp.

6.42.3.8 int CglUniqueRowCuts::insertIfNotDuplicate ( const OsiRowCut & *cut* )6.42.3.9 void CglUniqueRowCuts::addCuts ( OsiCuts & *cs* )

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

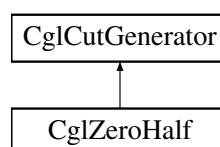
- src/CglPreProcess/[CglPreProcess.hpp](#)

## 6.43 CglZeroHalf Class Reference

Zero Half Cut Generator Class.

#include &lt;CglZeroHalf.hpp&gt;

Inheritance diagram for CglZeroHalf:





## Public Member Functions

### Generate Cuts

- virtual void [generateCuts](#) (const OsiSolverInterface &si, OsiCuts &cs, const [CglTreeInfo](#) info=[CglTreeInfo](#)())  
*Generate zero half cuts for the model accessed through the solver interface.*

### Sets and Gets

- int [getFlags](#) () const  
*Get flags.*
- void [setFlags](#) (int value)  
*Set flags.*

### Constructors and destructors

- [CglZeroHalf](#) ()  
*Default constructor.*
- [CglZeroHalf](#) (const [CglZeroHalf](#) &)  
*Copy constructor.*
- virtual [CglCutGenerator](#) \* [clone](#) () const  
*Clone.*
- [CglZeroHalf](#) & [operator=](#) (const [CglZeroHalf](#) &rhs)  
*Assignment operator.*
- virtual [~CglZeroHalf](#) ()  
*Destructor.*
- virtual std::string [generateCpp](#) (FILE \*fp)  
*Create C++ lines to get to current state.*
- virtual void [refreshSolver](#) (OsiSolverInterface \*solver)  
*This can be used to refresh any information.*

## Friends

- void [CglZeroHalfUnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)  
*A function that tests the methods in the [CglZeroHalf](#) class.*

## Additional Inherited Members

### 6.43.1 Detailed Description

Zero Half Cut Generator Class.

This class generates zero half cuts via the following method:

See -

G. Andreello, A. Caprara, M. Fischetti, "Embedding Cuts in a Branch and Cut Framework: a Computational Study with {0,1/2}-Cuts", INFORMS Journal on Computing 19(2), 229-238, 2007.

Definition at line 26 of file [CglZeroHalf.hpp](#).

### 6.43.2 Constructor & Destructor Documentation

#### 6.43.2.1 [CglZeroHalf::CglZeroHalf](#) ( )

Default constructor.

#### 6.43.2.2 CglZeroHalf::CglZeroHalf ( const CglZeroHalf & )

Copy constructor.

#### 6.43.2.3 virtual CglZeroHalf::~~CglZeroHalf ( ) [virtual]

Destructor.

### 6.43.3 Member Function Documentation

#### 6.43.3.1 virtual void CglZeroHalf::generateCuts ( const OsiSolverInterface & *si*, OsiCuts & *cs*, const CglTreeInfo *info* = CglTreeInfo ( ) ) [virtual]

Generate zero half cuts for the model accessed through the solver interface.

Insert generated cuts into the cut set *cs*.

Implements [CglCutGenerator](#).

#### 6.43.3.2 int CglZeroHalf::getFlags ( ) const [inline]

Get flags.

Definition at line 44 of file CglZeroHalf.hpp.

#### 6.43.3.3 void CglZeroHalf::setFlags ( int *value* ) [inline]

Set flags.

Definition at line 47 of file CglZeroHalf.hpp.

#### 6.43.3.4 virtual CglCutGenerator\* CglZeroHalf::clone ( ) const [virtual]

Clone.

Implements [CglCutGenerator](#).

#### 6.43.3.5 CglZeroHalf& CglZeroHalf::operator= ( const CglZeroHalf & *rhs* )

Assignment operator.

#### 6.43.3.6 virtual std::string CglZeroHalf::generateCpp ( FILE \* *fp* ) [virtual]

Create C++ lines to get to current state.

Reimplemented from [CglCutGenerator](#).

#### 6.43.3.7 virtual void CglZeroHalf::refreshSolver ( OsiSolverInterface \* *solver* ) [virtual]

This can be used to refresh any information.

Reimplemented from [CglCutGenerator](#).

### 6.43.4 Friends And Related Function Documentation

#### 6.43.4.1 void CglZeroHalfUnitTest ( const OsiSolverInterface \* *siP*, const std::string *mpdDir* ) [friend]

A function that tests the methods in the [CglZeroHalf](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

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

- [src/CglZeroHalf/CglZeroHalf.hpp](#)

## 6.44 cliqueEntry Struct Reference

Derived class to pick up probing info.

```
#include <CglTreeInfo.hpp>
```

### Public Attributes

- unsigned int [fixes](#)

### 6.44.1 Detailed Description

Derived class to pick up probing info.

Definition at line 79 of file CglTreeInfo.hpp.

### 6.44.2 Member Data Documentation

#### 6.44.2.1 unsigned int cliqueEntry::fixes

Definition at line 82 of file CglTreeInfo.hpp.

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

- [src/CglTreeInfo.hpp](#)

## 6.45 cut Struct Reference

```
#include <Cgl012cut.hpp>
```

### Public Attributes

- int [n\\_of\\_constr](#)
- int \* [constr\\_list](#)
- short int \* [in\\_constr\\_list](#)
- int [cnzcnt](#)
- int \* [cind](#)
- int \* [cval](#)
- int [crhs](#)
- char [csense](#)
- double [violation](#)

#### 6.45.1 Detailed Description

Definition at line 153 of file Cgl012cut.hpp.

#### 6.45.2 Member Data Documentation

##### 6.45.2.1 int cut::n\_of\_constr

Definition at line 154 of file Cgl012cut.hpp.

##### 6.45.2.2 int\* cut::constr\_list

Definition at line 155 of file Cgl012cut.hpp.

##### 6.45.2.3 short int\* cut::in\_constr\_list

Definition at line 156 of file Cgl012cut.hpp.

##### 6.45.2.4 int cut::cnzcnt

Definition at line 159 of file Cgl012cut.hpp.

##### 6.45.2.5 int\* cut::cind

Definition at line 160 of file Cgl012cut.hpp.

##### 6.45.2.6 int\* cut::cval

Definition at line 161 of file Cgl012cut.hpp.

##### 6.45.2.7 int cut::crhs

Definition at line 162 of file Cgl012cut.hpp.

##### 6.45.2.8 char cut::csense

Definition at line 163 of file Cgl012cut.hpp.

##### 6.45.2.9 double cut::violation

Definition at line 164 of file Cgl012cut.hpp.

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

- [src/CglZeroHalf/Cgl012cut.hpp](#)

## 6.46 cut\_list Struct Reference

```
#include <Cgl012cut.hpp>
```

#### Public Attributes

- int [cnum](#)
- [cut](#) \*\* [list](#)

#### 6.46.1 Detailed Description

Definition at line 167 of file Cgl012cut.hpp.

#### 6.46.2 Member Data Documentation

##### 6.46.2.1 int cut\_list::cnum

Definition at line 168 of file Cgl012cut.hpp.

##### 6.46.2.2 cut\*\* cut\_list::list

Definition at line 169 of file Cgl012cut.hpp.

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

- [src/CglZeroHalf/Cgl012cut.hpp](#)

### 6.47 cutParams Struct Reference

```
#include <CglTwomir.hpp>
```

#### Public Attributes

- int [q\\_min](#)
- int [q\\_max](#)
- int [t\\_min](#)
- int [t\\_max](#)
- int [a\\_max](#)
- int [max\\_elements](#)

#### 6.47.1 Detailed Description

Definition at line 33 of file CglTwomir.hpp.

#### 6.47.2 Member Data Documentation

##### 6.47.2.1 int cutParams::q\_min

Definition at line 34 of file CglTwomir.hpp.

##### 6.47.2.2 int cutParams::q\_max

Definition at line 35 of file CglTwomir.hpp.

##### 6.47.2.3 int cutParams::t\_min

Definition at line 36 of file CglTwomir.hpp.

##### 6.47.2.4 int cutParams::t\_max

Definition at line 37 of file CglTwomir.hpp.

## 6.47.2.5 int cutParams::a\_max

Definition at line 38 of file CglTwomir.hpp.

## 6.47.2.6 int cutParams::max\_elements

Definition at line 39 of file CglTwomir.hpp.

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

- src/CglTwomir/CglTwomir.hpp

## 6.48 LAP::Cuts Struct Reference

To store extra cuts generated by columns from which they origin.

```
#include <CglLandPUtils.hpp>
```

## Public Member Functions

- [Cuts](#) ()
- int [insertAll](#) (OsiCuts &cs, CoinRelFltEq &eq)  
*Puts all the cuts into an OsiCuts.*
- [~Cuts](#) ()  
*Destructor.*
- OsiRowCut \* [rowCut](#) (unsigned int i)  
*Access to row cut indexed by i.*
- const OsiRowCut \* [rowCut](#) (unsigned int i) const  
*const access to row cut indexed by i*
- void [insert](#) (int i, OsiRowCut \*cut)  
*insert a cut for variable i and count number of cuts.*
- int [numberCuts](#) ()  
*Access to number of cuts.*
- void [resize](#) (unsigned int i)  
*resize vector.*

## 6.48.1 Detailed Description

To store extra cuts generated by columns from which they origin.

Definition at line 59 of file CglLandPUtils.hpp.

## 6.48.2 Constructor &amp; Destructor Documentation

## 6.48.2.1 LAP::Cuts::Cuts ( ) [inline]

Definition at line 61 of file CglLandPUtils.hpp.

## 6.48.2.2 LAP::Cuts::~~Cuts ( ) [inline]

Destructor.

Definition at line 67 of file CglLandPUtils.hpp.

### 6.48.3 Member Function Documentation

#### 6.48.3.1 `int LAP::Cuts::insertAll ( OsiCuts & cs, CoinRelFitEq & eq )`

Puts all the cuts into an OsiCuts.

#### 6.48.3.2 `OsiRowCut* LAP::Cuts::rowCut ( unsigned int i ) [inline]`

Access to row cut indexed by i.

Definition at line 69 of file CglLandPUtils.hpp.

#### 6.48.3.3 `const OsiRowCut* LAP::Cuts::rowCut ( unsigned int i ) const [inline]`

const access to row cut indexed by i

Definition at line 74 of file CglLandPUtils.hpp.

#### 6.48.3.4 `void LAP::Cuts::insert ( int i, OsiRowCut * cut )`

insert a cut for variable i and count number of cuts.

#### 6.48.3.5 `int LAP::Cuts::numberCuts ( ) [inline]`

Access to number of cuts.

Definition at line 81 of file CglLandPUtils.hpp.

#### 6.48.3.6 `void LAP::Cuts::resize ( unsigned int i ) [inline]`

resize vector.

Definition at line 86 of file CglLandPUtils.hpp.

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

- [src/CglLandP/CglLandPUtils.hpp](#)

## 6.49 cycle Struct Reference

```
#include <Cgl012cut.hpp>
```

### Public Attributes

- double [weight](#)
- int [length](#)
- [edge](#) \*\* [edge\\_list](#)

### 6.49.1 Detailed Description

Definition at line 142 of file Cgl012cut.hpp.

### 6.49.2 Member Data Documentation

#### 6.49.2.1 `double cycle::weight`

Definition at line 143 of file `Cgl012cut.hpp`.

#### 6.49.2.2 `int cycle::length`

Definition at line 144 of file `Cgl012cut.hpp`.

#### 6.49.2.3 `edge** cycle::edge_list`

Definition at line 145 of file `Cgl012cut.hpp`.

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

- `src/CglZeroHalf/Cgl012cut.hpp`

### 6.50 `cycle_list` Struct Reference

```
#include <Cgl012cut.hpp>
```

#### Public Attributes

- `int cnum`
- `cycle** list`

#### 6.50.1 Detailed Description

Definition at line 148 of file `Cgl012cut.hpp`.

#### 6.50.2 Member Data Documentation

##### 6.50.2.1 `int cycle_list::cnum`

Definition at line 149 of file `Cgl012cut.hpp`.

##### 6.50.2.2 `cycle** cycle_list::list`

Definition at line 150 of file `Cgl012cut.hpp`.

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

- `src/CglZeroHalf/Cgl012cut.hpp`

### 6.51 `DGG_constraint_t` Struct Reference

```
#include <CglTwomir.hpp>
```

#### Public Attributes

- `int nz`
- `int max_nz`



- double \* [coeff](#)
- int \* [index](#)
- double [rhs](#)
- char [sense](#)

#### 6.51.1 Detailed Description

Definition at line 13 of file CglTwomir.hpp.

#### 6.51.2 Member Data Documentation

##### 6.51.2.1 int DGG\_constraint\_t::nz

Definition at line 16 of file CglTwomir.hpp.

##### 6.51.2.2 int DGG\_constraint\_t::max\_nz

Definition at line 17 of file CglTwomir.hpp.

##### 6.51.2.3 double\* DGG\_constraint\_t::coeff

Definition at line 18 of file CglTwomir.hpp.

##### 6.51.2.4 int\* DGG\_constraint\_t::index

Definition at line 19 of file CglTwomir.hpp.

##### 6.51.2.5 double DGG\_constraint\_t::rhs

Definition at line 20 of file CglTwomir.hpp.

##### 6.51.2.6 char DGG\_constraint\_t::sense

Definition at line 21 of file CglTwomir.hpp.

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

- src/CglTwomir/[CglTwomir.hpp](#)

## 6.52 DGG\_data\_t Struct Reference

```
#include <CglTwomir.hpp>
```

#### Public Attributes

- double [gomory\\_threshold](#)
- int [ncol](#)
- int [nrow](#)
- int [ninteger](#)
- int [nbasic\\_col](#)
- int [nbasic\\_row](#)
- int \* [info](#)

- double \* [lb](#)
- double \* [ub](#)
- double \* [x](#)
- double \* [rc](#)
- double \* [opt\\_x](#)
- [cutParams](#) [cparams](#)

### 6.52.1 Detailed Description

Definition at line 42 of file CglTwomir.hpp.

### 6.52.2 Member Data Documentation

#### 6.52.2.1 double DGG\_data\_t::gomory\_threshold

Definition at line 44 of file CglTwomir.hpp.

#### 6.52.2.2 int DGG\_data\_t::ncol

Definition at line 45 of file CglTwomir.hpp.

#### 6.52.2.3 int DGG\_data\_t::nrow

Definition at line 45 of file CglTwomir.hpp.

#### 6.52.2.4 int DGG\_data\_t::ninteger

Definition at line 45 of file CglTwomir.hpp.

#### 6.52.2.5 int DGG\_data\_t::nbasic\_col

Definition at line 49 of file CglTwomir.hpp.

#### 6.52.2.6 int DGG\_data\_t::nbasic\_row

Definition at line 49 of file CglTwomir.hpp.

#### 6.52.2.7 int\* DGG\_data\_t::info

Definition at line 53 of file CglTwomir.hpp.

#### 6.52.2.8 double\* DGG\_data\_t::lb

Definition at line 54 of file CglTwomir.hpp.

#### 6.52.2.9 double\* DGG\_data\_t::ub

Definition at line 55 of file CglTwomir.hpp.

#### 6.52.2.10 double\* DGG\_data\_t::x

Definition at line 56 of file CglTwomir.hpp.

#### 6.52.2.11 double\* DGG\_data\_t::rc

Definition at line 57 of file CglTwomir.hpp.

#### 6.52.2.12 double\* DGG\_data\_t::opt\_x

Definition at line 58 of file CglTwomir.hpp.

#### 6.52.2.13 cutParams DGG\_data\_t::cparams

Definition at line 60 of file CglTwomir.hpp.

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

- src/CglTwomir/[CglTwomir.hpp](#)

### 6.53 DGG\_list\_t Struct Reference

```
#include <CglTwomir.hpp>
```

#### Public Attributes

- int [n](#)
- [DGG\\_constraint\\_t](#) \*\* [c](#)
- int \* [ctype](#)
- double \* [alpha](#)

#### 6.53.1 Detailed Description

Definition at line 25 of file CglTwomir.hpp.

#### 6.53.2 Member Data Documentation

##### 6.53.2.1 int DGG\_list\_t::n

Definition at line 26 of file CglTwomir.hpp.

##### 6.53.2.2 DGG\_constraint\_t\*\* DGG\_list\_t::c

Definition at line 27 of file CglTwomir.hpp.

##### 6.53.2.3 int\* DGG\_list\_t::ctype

Definition at line 28 of file CglTwomir.hpp.

##### 6.53.2.4 double\* DGG\_list\_t::alpha

Definition at line 29 of file CglTwomir.hpp.

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

- src/CglTwomir/[CglTwomir.hpp](#)

## 6.54 disaggregationAction Struct Reference

Only useful type of disaggregation is most normal For now just done for 0-1 variables Can be used for building cliques.

```
#include <CglProbing.hpp>
```

### Public Attributes

- unsigned int [affected](#)

#### 6.54.1 Detailed Description

Only useful type of disaggregation is most normal For now just done for 0-1 variables Can be used for building cliques.

Definition at line 16 of file CglProbing.hpp.

#### 6.54.2 Member Data Documentation

##### 6.54.2.1 unsigned int disaggregationAction::affected

Definition at line 21 of file CglProbing.hpp.

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

- src/CglProbing/[CglProbing.hpp](#)

## 6.55 edge Struct Reference

```
#include <Cgl012cut.hpp>
```

### Public Attributes

- int [endpoint1](#)
- int [endpoint2](#)
- double [weight](#)
- short int [parity](#)
- int [constr](#)
- [info\\_weak](#) \* [weak](#)

#### 6.55.1 Detailed Description

Definition at line 104 of file Cgl012cut.hpp.

#### 6.55.2 Member Data Documentation

##### 6.55.2.1 int edge::endpoint1

Definition at line 105 of file Cgl012cut.hpp.

#### 6.55.2.2 int edge::endpoint2

Definition at line 105 of file Cgl012cut.hpp.

#### 6.55.2.3 double edge::weight

Definition at line 106 of file Cgl012cut.hpp.

#### 6.55.2.4 short int edge::parity

Definition at line 107 of file Cgl012cut.hpp.

#### 6.55.2.5 int edge::constr

Definition at line 108 of file Cgl012cut.hpp.

#### 6.55.2.6 info\_weak\* edge::weak

Definition at line 109 of file Cgl012cut.hpp.

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

- src/CglZeroHalf/Cgl012cut.hpp

### 6.56 ilp Struct Reference

```
#include <Cgl012cut.hpp>
```

#### Public Attributes

- int [mr](#)
- int [mc](#)
- int [mnz](#)
- int \* [mtbeg](#)
- int \* [mtcnt](#)
- int \* [mtind](#)
- int \* [mtval](#)
- int \* [vlb](#)
- int \* [vub](#)
- int \* [mrhs](#)
- char \* [msense](#)
- const double \* [xstar](#)

#### 6.56.1 Detailed Description

Definition at line 60 of file Cgl012cut.hpp.

#### 6.56.2 Member Data Documentation

##### 6.56.2.1 int ilp::mr

Definition at line 61 of file Cgl012cut.hpp.

#### 6.56.2.2 int ilp::mc

Definition at line 62 of file Cgl012cut.hpp.

#### 6.56.2.3 int ilp::mnz

Definition at line 63 of file Cgl012cut.hpp.

#### 6.56.2.4 int\* ilp::mtbeg

Definition at line 64 of file Cgl012cut.hpp.

#### 6.56.2.5 int\* ilp::mtcnt

Definition at line 65 of file Cgl012cut.hpp.

#### 6.56.2.6 int\* ilp::mtind

Definition at line 66 of file Cgl012cut.hpp.

#### 6.56.2.7 int\* ilp::mtval

Definition at line 67 of file Cgl012cut.hpp.

#### 6.56.2.8 int\* ilp::vlb

Definition at line 68 of file Cgl012cut.hpp.

#### 6.56.2.9 int\* ilp::vub

Definition at line 69 of file Cgl012cut.hpp.

#### 6.56.2.10 int\* ilp::mrhs

Definition at line 70 of file Cgl012cut.hpp.

#### 6.56.2.11 char\* ilp::msense

Definition at line 71 of file Cgl012cut.hpp.

#### 6.56.2.12 const double\* ilp::xstar

Definition at line 72 of file Cgl012cut.hpp.

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

- [src/CglZeroHalf/Cgl012cut.hpp](#)

## 6.57 info\_weak Struct Reference

```
#include <Cgl012cut.hpp>
```

### Public Attributes

- int [nweak](#)
- int \* [var](#)

- `short int * type`

#### 6.57.1 Detailed Description

Definition at line 98 of file Cgl012cut.hpp.

#### 6.57.2 Member Data Documentation

##### 6.57.2.1 `int info_weak::nweak`

Definition at line 99 of file Cgl012cut.hpp.

##### 6.57.2.2 `int* info_weak::var`

Definition at line 100 of file Cgl012cut.hpp.

##### 6.57.2.3 `short int* info_weak::type`

Definition at line 101 of file Cgl012cut.hpp.

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

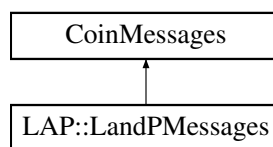
- `src/CglZeroHalf/Cgl012cut.hpp`

## 6.58 LAP::LandPMessages Class Reference

Message handler for lift-and-project simplex.

```
#include <CglLandPMessages.hpp>
```

Inheritance diagram for LAP::LandPMessages:



#### Public Member Functions

- `LandPMessages ()`

*Constructor.*

#### 6.58.1 Detailed Description

Message handler for lift-and-project simplex.

Definition at line 50 of file CglLandPMessages.hpp.

### 6.58.2 Constructor & Destructor Documentation

#### 6.58.2.1 LAP::LandPMessages::LandPMessages ( )

Constructor.

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

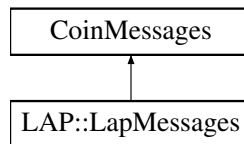
- [src/CglLandP/CglLandPMessages.hpp](#)

## 6.59 LAP::LapMessages Class Reference

Output messages for Cgl.

```
#include <CglLandP.hpp>
```

Inheritance diagram for LAP::LapMessages:



### Public Member Functions

- [LapMessages](#) ()  
*Constructor.*
- virtual [~LapMessages](#) ()  
*destructor.*

### 6.59.1 Detailed Description

Output messages for Cgl.

Definition at line 38 of file CglLandP.hpp.

### 6.59.2 Constructor & Destructor Documentation

#### 6.59.2.1 LAP::LapMessages::LapMessages ( )

Constructor.

#### 6.59.2.2 virtual LAP::LapMessages::~~LapMessages ( ) [inline], [virtual]

destructor.

Definition at line 44 of file CglLandP.hpp.

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

- [src/CglLandP/CglLandP.hpp](#)



## 6.60 log\_var Struct Reference

```
#include <Cgl012cut.hpp>
```

### Public Attributes

- [int n\\_it\\_zero](#)

### 6.60.1 Detailed Description

Definition at line 197 of file Cgl012cut.hpp.

### 6.60.2 Member Data Documentation

#### 6.60.2.1 int log\_var::n\_it\_zero

Definition at line 198 of file Cgl012cut.hpp.

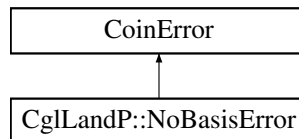
The documentation for this struct was generated from the following file:

- [src/CglZeroHalf/Cgl012cut.hpp](#)

## 6.61 CglLandP::NoBasisError Class Reference

```
#include <CglLandP.hpp>
```

Inheritance diagram for CglLandP::NoBasisError:



### Public Member Functions

- [NoBasisError\(\)](#)

### 6.61.1 Detailed Description

Definition at line 218 of file CglLandP.hpp.

### 6.61.2 Constructor & Destructor Documentation

#### 6.61.2.1 CglLandP::NoBasisError::NoBasisError ( ) [inline]

Definition at line 221 of file CglLandP.hpp.

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

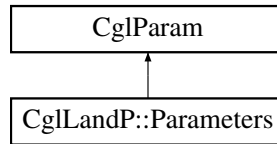
- [src/CglLandP/CglLandP.hpp](#)

## 6.62 CglLandP::Parameters Class Reference

Class storing parameters.

```
#include <CglLandP.hpp>
```

Inheritance diagram for CglLandP::Parameters:



### Public Member Functions

- [Parameters](#) ()  
*Default constructor (with default values)*
- [Parameters](#) (const [Parameters](#) &other)  
*Copy constructor.*
- [Parameters](#) & [operator=](#) (const [Parameters](#) &other)  
*Assignment operator.*

### Public Attributes

#### integer parameters

- int [pivotLimit](#)  
*Max number of pivots before we generate the cut 20.*
- int [pivotLimitInTree](#)  
*Max number of pivots at regular nodes.*
- int [maxCutPerRound](#)  
*Maximum number of cuts generated at a given round.*
- int [failedPivotLimit](#)  
*Maximum number of failed pivots before aborting.*
- int [degeneratePivotLimit](#)  
*maximum number of consecutive degenerate pivots 0*
- int [extraCutsLimit](#)  
*Maximum number of extra rows to generate per round.*

#### double parameters

- double [pivotTol](#)  
*Tolerance for small pivots values (should be the same as the solver.*
- double [away](#)  
*A variable have to be at least away from integrity to be generated.*
- double [timeLimit](#)  
*Total time limit for cut generation.*
- double [singleCutTimeLimit](#)  
*Time limit for generating a single cut.*
- double [rhsWeight](#)  
*Weight to put in RHS of normalization if static.*

## Flags

- bool [useTableauRow](#)  
*Do we use tableau row or the disjunction (I don't really get that there should be a way to always use the tableau)*
- bool [modularize](#)  
*Do we apply Egon Balas's Heuristic for modularized cuts.*
- bool [strengthen](#)  
*Do we strengthen the final cut (always do if modularize is 1)*
- bool [countMistakenRc](#)  
*Whether to limit or not the number of mistaken RC (when perturbation is applied).*
- [SeparationSpaces](#) [sepSpace](#)  
*Work in the reduced space (only non-structurals enter the basis)*
- bool [perturb](#)  
*Apply perturbation procedure.*
- [Normalization](#) [normalization](#)  
*How to weight normalization.*
- [RhsWeightType](#) [rhsWeightType](#)  
*How to weight RHS of normalization.*
- [LHSnorm](#) [lhs\\_norm](#)  
*How to weight LHS of normalization.*
- [ExtraCutsMode](#) [generateExtraCuts](#)  
*Generate extra constraints from optimal lift-and-project basis.*
- [SelectionRules](#) [pivotSelection](#)  
*Which rule to apply for choosing entering and leaving variables.*

## Additional Inherited Members

### 6.62.1 Detailed Description

Class storing parameters.

#### Remarks

I take all parameters from Ionut's code

Definition at line 107 of file CglLandP.hpp.

### 6.62.2 Constructor & Destructor Documentation

#### 6.62.2.1 CglLandP::Parameters::Parameters ( )

Default constructor (with default values)

#### 6.62.2.2 CglLandP::Parameters::Parameters ( const Parameters & other )

Copy constructor.

### 6.62.3 Member Function Documentation

#### 6.62.3.1 Parameters& CglLandP::Parameters::operator= ( const Parameters & other )

Assignment operator.

#### 6.62.4 Member Data Documentation

##### 6.62.4.1 int CglLandP::Parameters::pivotLimit

Max number of pivots before we generate the cut 20.

Definition at line 121 of file CglLandP.hpp.

##### 6.62.4.2 int CglLandP::Parameters::pivotLimitInTree

Max number of pivots at regular nodes.

Put a value if you want it lower than the global pivot limit. 100.

Definition at line 124 of file CglLandP.hpp.

##### 6.62.4.3 int CglLandP::Parameters::maxCutPerRound

Maximum number of cuts generated at a given round.

Definition at line 126 of file CglLandP.hpp.

##### 6.62.4.4 int CglLandP::Parameters::failedPivotLimit

Maximum number of failed pivots before aborting.

Definition at line 128 of file CglLandP.hpp.

##### 6.62.4.5 int CglLandP::Parameters::degeneratePivotLimit

maximum number of consecutive degenerate pivots 0

Definition at line 131 of file CglLandP.hpp.

##### 6.62.4.6 int CglLandP::Parameters::extraCutsLimit

Maximum number of extra rows to generate per round.

Definition at line 133 of file CglLandP.hpp.

##### 6.62.4.7 double CglLandP::Parameters::pivotTol

Tolerance for small pivots values (should be the same as the solver.

Definition at line 138 of file CglLandP.hpp.

##### 6.62.4.8 double CglLandP::Parameters::away

A variable have to be at least away from integrity to be generated.

Definition at line 140 of file CglLandP.hpp.

##### 6.62.4.9 double CglLandP::Parameters::timeLimit

Total time limit for cut generation.

Definition at line 142 of file CglLandP.hpp.

##### 6.62.4.10 double CglLandP::Parameters::singleCutTimeLimit

Time limit for generating a single cut.

Definition at line 144 of file CglLandP.hpp.

#### 6.62.4.11 `double CglLandP::Parameters::rhsWeight`

Weight to put in RHS of normalization if static.

Definition at line 146 of file CglLandP.hpp.

#### 6.62.4.12 `bool CglLandP::Parameters::useTableauRow`

Do we use tableau row or the disjunction (I don't really get that there should be a way to always use the tableau)

Definition at line 152 of file CglLandP.hpp.

#### 6.62.4.13 `bool CglLandP::Parameters::modularize`

Do we apply Egon Balas's Heuristic for modularized cuts.

Definition at line 154 of file CglLandP.hpp.

#### 6.62.4.14 `bool CglLandP::Parameters::strengthen`

Do we strengthen the final cut (always do if modularize is 1)

Definition at line 156 of file CglLandP.hpp.

#### 6.62.4.15 `bool CglLandP::Parameters::countMistakenRc`

Whether to limit or not the number of mistaken RC (when perturbation is applied).

Definition at line 158 of file CglLandP.hpp.

#### 6.62.4.16 `SeparationSpaces CglLandP::Parameters::sepSpace`

Work in the reduced space (only non-structurals enter the basis)

Definition at line 160 of file CglLandP.hpp.

#### 6.62.4.17 `bool CglLandP::Parameters::perturb`

Apply perturbation procedure.

Definition at line 162 of file CglLandP.hpp.

#### 6.62.4.18 `Normalization CglLandP::Parameters::normalization`

How to weight normalization.

Definition at line 164 of file CglLandP.hpp.

#### 6.62.4.19 `RhsWeightType CglLandP::Parameters::rhsWeightType`

How to weight RHS of normalization.

Definition at line 166 of file CglLandP.hpp.

#### 6.62.4.20 `LHSnorm CglLandP::Parameters::lhs_norm`

How to weight LHS of normalization.

Definition at line 168 of file CglLandP.hpp.

#### 6.62.4.21 ExtraCutsMode CglLandP::Parameters::generateExtraCuts

Generate extra constraints from optimal lift-and-project basis.

Definition at line 170 of file CglLandP.hpp.

#### 6.62.4.22 SelectionRules CglLandP::Parameters::pivotSelection

Which rule to apply for choosing entering and leaving variables.

Definition at line 172 of file CglLandP.hpp.

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

- [src/CglLandP/CglLandP.hpp](#)

## 6.63 parity\_ilp Struct Reference

```
#include <Cgl012cut.hpp>
```

### Public Attributes

- int [mr](#)
- int [mc](#)
- int [mnz](#)
- int \* [mtbeg](#)
- int \* [mtcnt](#)
- int \* [mtind](#)
- short int \* [mrhs](#)
- double \* [xstar](#)
- double \* [slack](#)
- short int \* [row\\_to\\_delete](#)
- short int \* [col\\_to\\_delete](#)
- int \* [gcd](#)
- short int \* [possible\\_weak](#)
- short int \* [type\\_even\\_weak](#)
- short int \* [type\\_odd\\_weak](#)
- double \* [loss\\_even\\_weak](#)
- double \* [loss\\_odd\\_weak](#)
- double \* [min\\_loss\\_by\\_weak](#)

#### 6.63.1 Detailed Description

Definition at line 75 of file Cgl012cut.hpp.

#### 6.63.2 Member Data Documentation

##### 6.63.2.1 int parity\_ilp::mr

Definition at line 76 of file Cgl012cut.hpp.

#### 6.63.2.2 `int parity_ilp::mc`

Definition at line 77 of file Cgl012cut.hpp.

#### 6.63.2.3 `int parity_ilp::mnz`

Definition at line 78 of file Cgl012cut.hpp.

#### 6.63.2.4 `int* parity_ilp::mtbeg`

Definition at line 79 of file Cgl012cut.hpp.

#### 6.63.2.5 `int* parity_ilp::mtcnt`

Definition at line 80 of file Cgl012cut.hpp.

#### 6.63.2.6 `int* parity_ilp::mtind`

Definition at line 81 of file Cgl012cut.hpp.

#### 6.63.2.7 `short int* parity_ilp::mrhs`

Definition at line 82 of file Cgl012cut.hpp.

#### 6.63.2.8 `double* parity_ilp::xstar`

Definition at line 83 of file Cgl012cut.hpp.

#### 6.63.2.9 `double* parity_ilp::slack`

Definition at line 84 of file Cgl012cut.hpp.

#### 6.63.2.10 `short int* parity_ilp::row_to_delete`

Definition at line 85 of file Cgl012cut.hpp.

#### 6.63.2.11 `short int* parity_ilp::col_to_delete`

Definition at line 86 of file Cgl012cut.hpp.

#### 6.63.2.12 `int* parity_ilp::gcd`

Definition at line 87 of file Cgl012cut.hpp.

#### 6.63.2.13 `short int* parity_ilp::possible_weak`

Definition at line 88 of file Cgl012cut.hpp.

#### 6.63.2.14 `short int* parity_ilp::type_even_weak`

Definition at line 89 of file Cgl012cut.hpp.

#### 6.63.2.15 `short int* parity_ilp::type_odd_weak`

Definition at line 91 of file Cgl012cut.hpp.

#### 6.63.2.16 double\* parity\_ilp::loss\_even\_weak

Definition at line 93 of file Cgl012cut.hpp.

#### 6.63.2.17 double\* parity\_ilp::loss\_odd\_weak

Definition at line 94 of file Cgl012cut.hpp.

#### 6.63.2.18 double\* parity\_ilp::min\_loss\_by\_weak

Definition at line 95 of file Cgl012cut.hpp.

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

- [src/CglZeroHalf/Cgl012cut.hpp](#)

## 6.64 pool\_cut Struct Reference

```
#include <Cgl012cut.hpp>
```

### Public Attributes

- int [n\\_of\\_constr](#)
- int \* [constr\\_list](#)
- int [code](#)
- int [n\\_it\\_violated](#)
- int [it\\_found](#)
- double [score](#)

#### 6.64.1 Detailed Description

Definition at line 172 of file Cgl012cut.hpp.

#### 6.64.2 Member Data Documentation

##### 6.64.2.1 int pool\_cut::n\_of\_constr

Definition at line 173 of file Cgl012cut.hpp.

##### 6.64.2.2 int\* pool\_cut::constr\_list

Definition at line 174 of file Cgl012cut.hpp.

##### 6.64.2.3 int pool\_cut::code

Definition at line 175 of file Cgl012cut.hpp.

##### 6.64.2.4 int pool\_cut::n\_it\_violated

Definition at line 176 of file Cgl012cut.hpp.

##### 6.64.2.5 int pool\_cut::it\_found

Definition at line 179 of file Cgl012cut.hpp.



#### 6.64.2.6 double pool\_cut::score

Definition at line 180 of file Cgl012cut.hpp.

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

- src/CglZeroHalf/Cgl012cut.hpp

### 6.65 pool\_cut\_list Struct Reference

```
#include <Cgl012cut.hpp>
```

#### Public Attributes

- int [cnum](#)
- [pool\\_cut](#) \*\* [list](#)
- int \* [ncod](#)

#### 6.65.1 Detailed Description

Definition at line 184 of file Cgl012cut.hpp.

#### 6.65.2 Member Data Documentation

##### 6.65.2.1 int pool\_cut\_list::cnum

Definition at line 185 of file Cgl012cut.hpp.

##### 6.65.2.2 pool\_cut\*\* pool\_cut\_list::list

Definition at line 186 of file Cgl012cut.hpp.

##### 6.65.2.3 int\* pool\_cut\_list::ncod

Definition at line 187 of file Cgl012cut.hpp.

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

- src/CglZeroHalf/Cgl012cut.hpp

### 6.66 select\_cut Struct Reference

```
#include <Cgl012cut.hpp>
```

#### Public Attributes

- int \* [ccoef](#)
- int [crhs](#)
- int [pool\\_index](#)
- double [score](#)

### 6.66.1 Detailed Description

Definition at line 190 of file Cgl012cut.hpp.

### 6.66.2 Member Data Documentation

#### 6.66.2.1 int\* select\_cut::ccoef

Definition at line 191 of file Cgl012cut.hpp.

#### 6.66.2.2 int select\_cut::crhs

Definition at line 192 of file Cgl012cut.hpp.

#### 6.66.2.3 int select\_cut::pool\_index

Definition at line 193 of file Cgl012cut.hpp.

#### 6.66.2.4 double select\_cut::score

Definition at line 194 of file Cgl012cut.hpp.

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

- src/CglZeroHalf/Cgl012cut.hpp

## 6.67 separation\_graph Struct Reference

```
#include <Cgl012cut.hpp>
```

### Public Attributes

- int [nnodes](#)
- int [nedges](#)
- int \* [nodes](#)
- int \* [ind](#)
- [edge](#) \*\* [even\\_adj\\_list](#)
- [edge](#) \*\* [odd\\_adj\\_list](#)

### 6.67.1 Detailed Description

Definition at line 112 of file Cgl012cut.hpp.

### 6.67.2 Member Data Documentation

#### 6.67.2.1 int separation\_graph::nnodes

Definition at line 113 of file Cgl012cut.hpp.

#### 6.67.2.2 int separation\_graph::nedges

Definition at line 114 of file Cgl012cut.hpp.

**6.67.2.3 int\* separation\_graph::nodes**

Definition at line 115 of file Cgl012cut.hpp.

**6.67.2.4 int\* separation\_graph::ind**

Definition at line 116 of file Cgl012cut.hpp.

**6.67.2.5 edge\*\* separation\_graph::even\_adj\_list**

Definition at line 117 of file Cgl012cut.hpp.

**6.67.2.6 edge\*\* separation\_graph::odd\_adj\_list**

Definition at line 118 of file Cgl012cut.hpp.

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

- src/CglZeroHalf/Cgl012cut.hpp

**6.68 short\_path\_node Struct Reference**

```
#include <Cgl012cut.hpp>
```

**Public Attributes**

- long [dist](#)
- int [pred](#)

**6.68.1 Detailed Description**

Definition at line 137 of file Cgl012cut.hpp.

**6.68.2 Member Data Documentation****6.68.2.1 long short\_path\_node::dist**

Definition at line 138 of file Cgl012cut.hpp.

**6.68.2.2 int short\_path\_node::pred**

Definition at line 139 of file Cgl012cut.hpp.

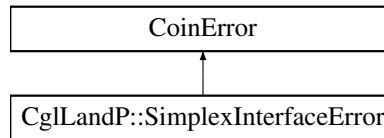
The documentation for this struct was generated from the following file:

- src/CglZeroHalf/Cgl012cut.hpp

**6.69 CglLandP::SimplexInterfaceError Class Reference**

```
#include <CglLandP.hpp>
```

Inheritance diagram for CglLandP::SimplexInterfaceError:



#### Public Member Functions

- [SimplexInterfaceError](#) ()

#### 6.69.1 Detailed Description

Definition at line 224 of file CglLandP.hpp.

#### 6.69.2 Constructor & Destructor Documentation

##### 6.69.2.1 CglLandP::SimplexInterfaceError::SimplexInterfaceError ( ) [inline]

Definition at line 227 of file CglLandP.hpp.

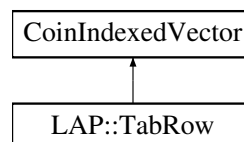
The documentation for this class was generated from the following file:

- src/CglLandP/[CglLandP.hpp](#)

## 6.70 LAP::TabRow Struct Reference

```
#include <CglLandPTabRow.hpp>
```

Inheritance diagram for LAP::TabRow:



#### Public Member Functions

- [TabRow](#) ()
- [TabRow](#) (const [CglLandPSimplex](#) \*si)
- [TabRow](#) (const [TabRow](#) &source)
- [TabRow](#) & [operator=](#) (const [TabRow](#) &r)
- bool [operator==](#) (const [TabRow](#) &r) const
- [~TabRow](#) ()
- void [modularize](#) (const bool \*integerVar)
- void [print](#) (std::ostream &os, int width=9, const int \*nonBasics=NULL, int m=0)
- const double & [operator\[\]](#) (const int &index) const
- double & [operator\[\]](#) (const int &index)

## Public Attributes

- int `num`  
*Row number.*
- double `rhs`  
*Row right-hand-side.*
- const `CgLLandPSimplex * si_`  
*Row of what?*
- bool `modularized_`  
*Flag to say if row is modularized.*

### 6.70.1 Detailed Description

Definition at line 21 of file `CgLLandPTabRow.hpp`.

### 6.70.2 Constructor & Destructor Documentation

#### 6.70.2.1 `LAP::TabRow::TabRow ( )` `[inline]`

Definition at line 33 of file `CgLLandPTabRow.hpp`.

#### 6.70.2.2 `LAP::TabRow::TabRow ( const CgLLandPSimplex * si )` `[inline]`

Definition at line 36 of file `CgLLandPTabRow.hpp`.

#### 6.70.2.3 `LAP::TabRow::TabRow ( const TabRow & source )` `[inline]`

Definition at line 39 of file `CgLLandPTabRow.hpp`.

#### 6.70.2.4 `LAP::TabRow::~~TabRow ( )` `[inline]`

Definition at line 57 of file `CgLLandPTabRow.hpp`.

### 6.70.3 Member Function Documentation

#### 6.70.3.1 `TabRow& LAP::TabRow::operator= ( const TabRow & r )` `[inline]`

Definition at line 44 of file `CgLLandPTabRow.hpp`.

#### 6.70.3.2 `bool LAP::TabRow::operator== ( const TabRow & r ) const`

#### 6.70.3.3 `void LAP::TabRow::modularize ( const bool * integerVar )`

#### 6.70.3.4 `void LAP::TabRow::print ( std::ostream & os, int width = 9, const int * nonBasics = NULL, int m = 0 )`

#### 6.70.3.5 `const double& LAP::TabRow::operator[] ( const int & index ) const` `[inline]`

Definition at line 66 of file `CgLLandPTabRow.hpp`.

#### 6.70.3.6 `double& LAP::TabRow::operator[] ( const int & index )` `[inline]`

Definition at line 72 of file `CgLLandPTabRow.hpp`.

## 6.70.4 Member Data Documentation

## 6.70.4.1 int LAP::TabRow::num

Row number.

Definition at line 24 of file CglLandPTabRow.hpp.

## 6.70.4.2 double LAP::TabRow::rhs

Row right-hand-side.

Definition at line 26 of file CglLandPTabRow.hpp.

## 6.70.4.3 const CglLandPSimplex\* LAP::TabRow::si\_

Row of what?

Definition at line 28 of file CglLandPTabRow.hpp.

## 6.70.4.4 bool LAP::TabRow::modularized\_

Flag to say if row is modularized.

Definition at line 31 of file CglLandPTabRow.hpp.

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

- [src/CglLandP/CglLandPTabRow.hpp](#)

## 6.71 LAP::Validator Class Reference

Class to validate or reject a cut.

```
#include <CglLandPValidator.hpp>
```

## Public Types

- enum [RejectionsReasons](#) {  
[NoneAccepted](#) =0, [SmallViolation](#), [SmallCoefficient](#), [BigDynamic](#),  
[DenseCut](#), [EmptyCut](#), [DummyEnd](#) }

*Reasons for rejecting a cut.*

## Public Member Functions

- [Validator](#) (double maxFillIn=1., double maxRatio=1e8, double minViolation=0, bool scale=false, double rhsScale=1)  
*Constructor with default values.*
- int [cleanCut](#) (OsiRowCut &aCut, const double \*solCut, const OsiSolverInterface &si, const [CglParam](#) &par, const double \*colLower, const double \*colUpper)  
*Clean an OsiCut.*
- int [cleanCut2](#) (OsiRowCut &aCut, const double \*solCut, const OsiSolverInterface &si, const [CglParam](#) &par, const double \*colLower, const double \*colUpper)  
*Clean an OsiCut by another method.*

- int [operator\(\)](#) (OsiRowCut &aCut, const double \*solCut, const OsiSolverInterface &si, const [CglParam](#) &par, const double \*colLower, const double \*colUpper)

*Call the cut cleaner.*

- const std::string & [failureString](#) ([RejectionsReasons](#) code) const
- const std::string & [failureString](#) (int code) const
- int [numRejected](#) ([RejectionsReasons](#) code) const
- int [numRejected](#) (int code) const

#### set functions

- void [setMaxFillIn](#) (double value)
- void [setMaxRatio](#) (double value)
- void [setMinViolation](#) (double value)
- void [setRhsScale](#) (double v)

#### get functions

- double [getMaxFillIn](#) ()
- double [getMaxRatio](#) ()
- double [getMinViolation](#) ()

#### 6.71.1 Detailed Description

Class to validate or reject a cut.

Definition at line 26 of file CglLandPValidator.hpp.

#### 6.71.2 Member Enumeration Documentation

##### 6.71.2.1 enum LAP::Validator::RejectionsReasons

Reasons for rejecting a cut.

Enumerator

***NoneAccepted***

***SmallViolation*** Violation of the cut is too small.

***SmallCoefficient*** There is a small coefficient we can not get rid off.

***BigDynamic*** Dynamic of coefficinet is too important.

***DenseCut*** cut is too dense

***EmptyCut*** After cleaning cut has become empty.

***DummyEnd*** dummy

Definition at line 30 of file CglLandPValidator.hpp.

#### 6.71.3 Constructor & Destructor Documentation

6.71.3.1 LAP::Validator::Validator ( double *maxFillIn* = 1 . , double *maxRatio* = 1e8, double *minViolation* = 0, bool *scale* = false, double *rhsScale* = 1 )

Constructor with default values.

## 6.71.4 Member Function Documentation

6.71.4.1 `int LAP::Validator::cleanCut ( OsiRowCut & aCut, const double * solCut, const OsiSolverInterface & si, const CglParam & par, const double * colLower, const double * colUpper )`

Clean an OsiCut.

6.71.4.2 `int LAP::Validator::cleanCut2 ( OsiRowCut & aCut, const double * solCut, const OsiSolverInterface & si, const CglParam & par, const double * colLower, const double * colUpper )`

Clean an OsiCut by another method.

6.71.4.3 `int LAP::Validator::operator() ( OsiRowCut & aCut, const double * solCut, const OsiSolverInterface & si, const CglParam & par, const double * colLower, const double * colUpper )` `[inline]`

Call the cut cleaner.

Definition at line 55 of file CglLandPValidator.hpp.

6.71.4.4 `void LAP::Validator::setMaxFillIn ( double value )` `[inline]`

Definition at line 62 of file CglLandPValidator.hpp.

6.71.4.5 `void LAP::Validator::setMaxRatio ( double value )` `[inline]`

Definition at line 66 of file CglLandPValidator.hpp.

6.71.4.6 `void LAP::Validator::setMinViolation ( double value )` `[inline]`

Definition at line 70 of file CglLandPValidator.hpp.

6.71.4.7 `void LAP::Validator::setRhsScale ( double v )` `[inline]`

Definition at line 75 of file CglLandPValidator.hpp.

6.71.4.8 `double LAP::Validator::getMaxFillIn ( )` `[inline]`

Definition at line 82 of file CglLandPValidator.hpp.

6.71.4.9 `double LAP::Validator::getMaxRatio ( )` `[inline]`

Definition at line 86 of file CglLandPValidator.hpp.

6.71.4.10 `double LAP::Validator::getMinViolation ( )` `[inline]`

Definition at line 90 of file CglLandPValidator.hpp.

6.71.4.11 `const std::string& LAP::Validator::failureString ( RejectionsReasons code ) const` `[inline]`

Definition at line 96 of file CglLandPValidator.hpp.

6.71.4.12 `const std::string& LAP::Validator::failureString ( int code ) const` `[inline]`

Definition at line 100 of file CglLandPValidator.hpp.

6.71.4.13 `int LAP::Validator::numRejected ( RejectionsReasons code ) const` `[inline]`

Definition at line 104 of file CglLandPValidator.hpp.



6.71.4.14 `int LAP::Validator::numRejected ( int code ) const` `[inline]`

Definition at line 108 of file `CglLandPValidator.hpp`.

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

- [src/CglLandP/CglLandPValidator.hpp](#)

## 7 File Documentation

### 7.1 `src/CglAllDifferent/CglAllDifferent.hpp` File Reference

```
#include <string>
#include "CglCutGenerator.hpp"
```

#### Classes

- class [CglAllDifferent](#)  
*AllDifferent Cut Generator Class This has a number of sets.*

### 7.2 `src/CglClique/CglClique.hpp` File Reference

```
#include "CglCutGenerator.hpp"
```

#### Classes

- class [CglClique](#)
- class [CglFakeClique](#)

#### Functions

- void [CglCliqueUnitTest](#) (const `OsiSolverInterface` \**siP*, const `std::string` *mpdDir*)  
*A function that tests the methods in the [CglClique](#) class.*

#### 7.2.1 Function Documentation

7.2.1.1 `void CglCliqueUnitTest ( const OsiSolverInterface * siP, const std::string mpdDir )`

A function that tests the methods in the [CglClique](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

### 7.3 src/CglConfig.h File Reference

```
#include "config_cgl_default.h"
```

### 7.4 src/CglCutGenerator.hpp File Reference

```
#include "OsiCuts.hpp"  
#include "OsiSolverInterface.hpp"  
#include "CglTreeInfo.hpp"
```

#### Classes

- class [CglCutGenerator](#)  
*Cut Generator Base Class.*

### 7.5 src/CglDuplicateRow/CglDuplicateRow.hpp File Reference

```
#include <string>  
#include "CglCutGenerator.hpp"
```

#### Classes

- class [CglDuplicateRow](#)  
*DuplicateRow Cut Generator Class.*

### 7.6 src/CglFlowCover/CglFlowCover.hpp File Reference

```
#include <iostream>  
#include "CoinError.hpp"  
#include "CglCutGenerator.hpp"
```

#### Classes

- class [CglFlowVUB](#)  
*Variable upper bound class.*
- class [CglFlowCover](#)  
*Lifed Simple Generalized Flow Cover Cut Generator Class.*

#### Typedefs

- typedef [CglFlowVUB](#) [CglFlowVLB](#)  
*Variable lower bound class, which is the same as vub.*

## Enumerations

- enum [CglFlowColType](#) { [CGLFLOW\\_COL\\_BINNEG](#) = -2, [CGLFLOW\\_COL\\_CONTNEG](#), [CGLFLOW\\_COL\\_CONTPOS](#) = 1, [CGLFLOW\\_COL\\_BINPOS](#) }

*This enumerative constant describes the various col types.*

- enum [CglFlowColStatus](#)
- enum [CglFlowColCut](#) { [CGLFLOW\\_COL\\_OUTCUT](#) = 0, [CGLFLOW\\_COL\\_INCURT](#), [CGLFLOW\\_COL\\_INCURTDONE](#), [CGLFLOW\\_COL\\_INLMIN](#), [CGLFLOW\\_COL\\_INLMINDONE](#), [CGLFLOW\\_COL\\_INLMINMIN](#), [CGLFLOW\\_COL\\_PRIME](#), [CGLFLOW\\_COL\\_SECONDRARY](#) }

*This enumerative constant describes the various stati of vars in a cut or not.*

- enum [CglFlowRowType](#) { [CGLFLOW\\_ROW\\_UNDEFINED](#), [CGLFLOW\\_ROW\\_VARUB](#), [CGLFLOW\\_ROW\\_VARLB](#), [CGLFLOW\\_ROW\\_VAREQ](#), [CGLFLOW\\_ROW\\_MIXUB](#), [CGLFLOW\\_ROW\\_MIXEQ](#), [CGLFLOW\\_ROW\\_NOBINUB](#), [CGLFLOW\\_ROW\\_NOBI-NEQ](#), [CGLFLOW\\_ROW\\_SUMVARUB](#), [CGLFLOW\\_ROW\\_SUMVAREQ](#), [CGLFLOW\\_ROW\\_UNINTERSTED](#) }

*This enumerative constant describes the various row types.*

## Functions

- `std::ostream & operator<< (std::ostream &os, const CglFlowVUB &v)`  
*Overloaded operator<< for printing VUB and VLB.*
- `void CglFlowCoverUnitTest (const OsiSolverInterface *siP, const std::string mpdDir)`  
*A function that tests the methods in the [CglFlowCover](#) class.*

## 7.6.1 Typedef Documentation

### 7.6.1.1 typedef [CglFlowVUB](#) [CglFlowVLB](#)

Variable lower bound class, which is the same as vub.

Definition at line 138 of file [CglFlowCover.hpp](#).

## 7.6.2 Enumeration Type Documentation

### 7.6.2.1 enum [CglFlowColType](#)

This enumerative constant describes the various col types.

## Enumerator

**[CGLFLOW\\_COL\\_BINNEG](#)** The column(variable) is a negative binary variable.

**[CGLFLOW\\_COL\\_CONTNEG](#)** The column is a negative continous variable.

**[CGLFLOW\\_COL\\_CONTPOS](#)** The column is a positive continous variable.

**[CGLFLOW\\_COL\\_BINPOS](#)** The column is a positive binary variable.

Definition at line 28 of file [CglFlowCover.hpp](#).

## 7.6.2.2 enum CglFlowColStatus

Definition at line 39 of file CglFlowCover.hpp.

## 7.6.2.3 enum CglFlowColCut

This enumerative constant describes the various stati of vars in a cut or not.

Enumerator

**CGLFLOW\_COL\_OUTCUT** The column is NOT in cover.  
**CGLFLOW\_COL\_INCUT** The column is in cover now.  
**CGLFLOW\_COL\_INCUTDONE** The column is decided to be in cover.  
**CGLFLOW\_COL\_INLMIN** The column is in L-.  
**CGLFLOW\_COL\_INLMINDONE** The column is decided to be in L-.  
**CGLFLOW\_COL\_INLMINMIN** The column is in L-.  
**CGLFLOW\_COL\_PRIME** This enumerative constant describes the various stati of vars in determining the cover.  
The column is a prime candidate.  
**CGLFLOW\_COL\_SECONDARY** The column is a secondary candidate.

Definition at line 44 of file CglFlowCover.hpp.

## 7.6.2.4 enum CglFlowRowType

This enumerative constant describes the various row types.

Enumerator

**CGLFLOW\_ROW\_UNDEFINED** The row type of this row is NOT defined yet.  
**CGLFLOW\_ROW\_VARUB** After the row is flipped to 'L', the row has exactly two variables: one is negative binary and the other is continous, and the RHS is zero.  
**CGLFLOW\_ROW\_VARLB** After the row is flipped to 'L', the row has exactlytwo variables: one is positive binary and the other is continous, and the RHS is zero.  
**CGLFLOW\_ROW\_VAREQ** The row sense is 'E', the row has exactly two variables: one is binary and the other is a continous, and the RHS is zero.  
**CGLFLOW\_ROW\_MIXUB** Rows can not be classfied into other types and the row sense is NOT 'E'.  
**CGLFLOW\_ROW\_MIXEQ** Rows can not be classified into other types and the row sense is 'E'.  
**CGLFLOW\_ROW\_NOBINUB** All variables are NOT binary and the row sense is NOT 'E'.  
**CGLFLOW\_ROW\_NOBINEQ** All variables are NOT binary and the row sense is 'E'.  
**CGLFLOW\_ROW\_SUMVARUB** The row has one binary and 2 or more other types of variables and the row sense is NOT 'E'.  
**CGLFLOW\_ROW\_SUMVAREQ** The row has one binary and 2 or more other types of variables and the row sense is 'E'.  
**CGLFLOW\_ROW\_UNINTERSTED** All variables are binary.

Definition at line 66 of file CglFlowCover.hpp.

## 7.6.3 Function Documentation

## 7.6.3.1 std::ostream&amp; operator&lt;&lt; ( std::ostream &amp; os, const CglFlowVUB &amp; v )

Overloaded operator<< for printing VUB and VLB.

### 7.6.3.2 void CglFlowCoverUnitTest ( const OsiSolverInterface \* siP, const std::string mpdDir )

A function that tests the methods in the [CglFlowCover](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

## 7.7 src/CglGMI/CglGMI.hpp File Reference

```
#include "CglCutGenerator.hpp"
#include "CglGMIParam.hpp"
#include "CoinWarmStartBasis.hpp"
#include "CoinFactorization.hpp"
```

### Classes

- class [CglGMI](#)

*Gomory cut generator with several cleaning procedures, used to test the numerical safety of the resulting cuts.*

### Functions

- void [CglGMIUnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)

*A function that tests the methods in the [CglGMI](#) class.*

### 7.7.1 Function Documentation

#### 7.7.1.1 void CglGMIUnitTest ( const OsiSolverInterface \* siP, const std::string mpdDir )

A function that tests the methods in the [CglGMI](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

## 7.8 src/CglGMI/CglGMIParam.hpp File Reference

```
#include "CglParam.hpp"
```

### Classes

- class [CglGMIParam](#)

*Class collecting parameters for the GMI cut generator.*

## 7.9 src/CglGomory/CglGomory.hpp File Reference

```
#include <string>
#include "CglCutGenerator.hpp"
```

### Classes

- class [CglGomory](#)  
*Gomory Cut Generator Class.*

### Functions

- void [CglGomoryUnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)  
*A function that tests the methods in the [CglGomory](#) class.*

#### 7.9.1 Function Documentation

##### 7.9.1.1 void CglGomoryUnitTest ( const OsiSolverInterface \* siP, const std::string mpdDir )

A function that tests the methods in the [CglGomory](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

## 7.10 src/CglKnapsackCover/CglKnapsackCover.hpp File Reference

```
#include <string>
#include "CglCutGenerator.hpp"
#include "CglTreeInfo.hpp"
```

### Classes

- class [CglKnapsackCover](#)  
*Knapsack Cover Cut Generator Class.*

### Functions

- void [CglKnapsackCoverUnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)  
*A function that tests the methods in the [CglKnapsackCover](#) class.*

#### 7.10.1 Function Documentation

##### 7.10.1.1 void CglKnapsackCoverUnitTest ( const OsiSolverInterface \* siP, const std::string mpdDir )

A function that tests the methods in the [CglKnapsackCover](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

## 7.11 src/CglLandP/CglLandP.hpp File Reference

```
#include "CglLandPValidator.hpp"
#include "CglCutGenerator.hpp"
#include "CglParam.hpp"
#include <iostream>
```

### Classes

- class [LAP::LapMessages](#)  
*Output messages for Cgl.*
- class [CglLandP](#)
- class [CglLandP::Parameters](#)  
*Class storing parameters.*
- class [CglLandP::NoBasisError](#)
- class [CglLandP::SimplexInterfaceError](#)

### Namespaces

- [LAP](#)  
*Performs one round of Lift & Project using [CglLandPSimplex](#) to build cuts.*

### Enumerations

- enum [LAP::LapMessagesTypes](#) {  
[LAP::BEGIN\\_ROUND](#), [LAP::END\\_ROUND](#), [LAP::DURING\\_SEP](#), [LAP::CUT\\_REJECTED](#),  
[LAP::CUT\\_FAILED](#), [LAP::CUT\\_GAP](#), [LAP::LAP\\_CUT\\_FAILED\\_DO\\_MIG](#), [LAP::LAP\\_MESSAGES\\_DUMMY\\_END](#) }

### Functions

- void [CglLandPUnitTest](#) (OsiSolverInterface \*si, const std::string &mpsDir)

#### 7.11.1 Function Documentation

7.11.1.1 void [CglLandPUnitTest](#) ( OsiSolverInterface \* *si*, const std::string & *mpsDir* )

## 7.12 src/CglLandP/CglLandPMessages.hpp File Reference

```
#include "CoinMessage.hpp"
#include "CoinMessageHandler.hpp"
```

## Classes

- class [LAP::LandPMessages](#)  
*Message handler for lift-and-project simplex.*

## Namespaces

- [LAP](#)  
*Performs one round of Lift & Project using [CgILandPSimplex](#) to build cuts.*

## Enumerations

- enum [LAP::LAP\\_messages](#) {  
[LAP::Separating](#), [LAP::FoundImprovingRow](#), [LAP::FoundBestImprovingCol](#), [LAP::WarnFailedBestImprovingCol](#),  
[LAP::LogHead](#), [LAP::PivotLog](#), [LAP::FinishedOptimal](#), [LAP::HitLimit](#),  
[LAP::NumberNegRc](#), [LAP::NumberZeroRc](#), [LAP::NumberPosRc](#), [LAP::WeightsStats](#),  
[LAP::WarnBadSigmaComputation](#), [LAP::WarnBadRowComputation](#), [LAP::WarnGiveUpRow](#), [LAP::PivotFailed-](#)  
[SigmaUnchanged](#),  
[LAP::PivotFailedSigmaIncreased](#), [LAP::FailedSigmaIncreased](#), [LAP::WarnBadRhsComputation](#), [LAP::Warn-](#)  
[FailedPivotTol](#),  
[LAP::WarnFailedPivotIf](#), [LAP::RoundStats](#), [LAP::CutStat](#), [LAP::DUMMY\\_END](#) }  
*Types of messages for lift-and-project simplex.*

## 7.13 src/CgILandP/CgILandPSimplex.hpp File Reference

```
#include <iostream>
#include <vector>
#include "CglConfig.h"
#include "CgILandP.hpp"
#include "OsiSolverInterface.hpp"
#include "CoinMessage.hpp"
#include "CoinMessageHandler.hpp"
#include "CoinWarmStartBasis.hpp"
#include "CoinPackedMatrix.hpp"
#include "CgILandPTabRow.hpp"
#include "CgILandPUtils.hpp"
#include "CgILandPMessages.hpp"
```

## Classes

- class [LAP::CgILandPSimplex](#)

## Namespaces

- [LAP](#)  
*Performs one round of Lift & Project using [CgILandPSimplex](#) to build cuts.*



## Macros

- `#define` [OLD\\_COMPUTATION](#)

### 7.13.1 Macro Definition Documentation

#### 7.13.1.1 `#define` OLD\_COMPUTATION

Definition at line 35 of file `CglLandPSimplex.hpp`.

## 7.14 `src/CglLandP/CglLandPTabRow.hpp` File Reference

```
#include "CoinIndexedVector.hpp"
#include <iostream>
```

## Classes

- struct [LAP::TabRow](#)

## Namespaces

- [LAP](#)

*Performs one round of Lift & Project using [CglLandPSimplex](#) to build cuts.*

## 7.15 `src/CglLandP/CglLandPUtls.hpp` File Reference

```
#include "CglLandPTabRow.hpp"
#include <vector>
#include <cmath>
```

## Classes

- struct [LAP::Cuts](#)

*To store extra cuts generated by columns from which they origin.*

## Namespaces

- [LAP](#)

*Performs one round of Lift & Project using [CglLandPSimplex](#) to build cuts.*

## Functions

- double [LAP::normCoef](#) (TabRow &row, int ncols, const int \*nonBasics)  
*Compute  $\sum_{j=1}^n |a_{ij}| |1 - a_{i0}|$  for row passed as argument.*
- void [LAP::scale](#) (OsiRowCut &cut)

- scale the cut passed as argument*
  - void [LAP::scale](#) (OsiRowCut &cut, double norma)
    - scale the cut passed as argument using provided normalization factor*
  - void [LAP::modularizeRow](#) (TabRow &row, const bool \*integerVar)
    - Modularize row.*
  - double [LAP::intersectionCutCoef](#) (double alpha\_i, double beta)
    - return the coefficients of the intersection cut*
  - double [LAP::modularizedCoef](#) (double alpha, double beta)
    - compute the modularized row coefficient for an integer variable*
  - bool [LAP::int\\_val](#) (double value, double tol)
    - Says is value is integer.*

## 7.16 src/CglLandP/CglLandPValidator.hpp File Reference

```
#include "OsiSolverInterface.hpp"
#include "CglParam.hpp"
#include <vector>
```

### Classes

- class [LAP::Validator](#)
  - Class to validate or reject a cut.*

### Namespaces

- [LAP](#)
  - Performs one round of Lift & Project using [CglLandPSimplex](#) to build cuts.*

## 7.17 src/CglLiftAndProject/CglLiftAndProject.hpp File Reference

```
#include <string>
#include "CglCutGenerator.hpp"
```

### Classes

- class [CglLiftAndProject](#)
  - Lift And Project Cut Generator Class.*

### Functions

- void [CglLiftAndProjectUnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)
  - A function that tests the methods in the [CglLiftAndProject](#) class.*

### 7.17.1 Function Documentation

#### 7.17.1.1 void CglLiftAndProjectUnitTest ( const OsiSolverInterface \* *siP*, const std::string *mpdDir* )

A function that tests the methods in the [CglLiftAndProject](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

## 7.18 src/CglMessage.hpp File Reference

```
#include "CoinPragma.hpp"
#include "CoinMessageHandler.hpp"
```

### Classes

- class [CglMessage](#)  
*This deals with Cgl messages (as against Osi messages etc)*

### Enumerations

- enum [CGL\\_Message](#) {  
[CGL\\_INFEASIBLE](#), [CGL\\_CLIQUES](#), [CGL\\_FIXED](#), [CGL\\_PROCESS\\_STATS](#),  
[CGL\\_SLACKS](#), [CGL\\_PROCESS\\_STATS2](#), [CGL\\_PROCESS\\_SOS1](#), [CGL\\_PROCESS\\_SOS2](#),  
[CGL\\_UNBOUNDED](#), [CGL\\_ELEMENTS\\_CHANGED1](#), [CGL\\_ELEMENTS\\_CHANGED2](#), [CGL\\_MADE\\_INTEGER](#),  
[CGL\\_ADDED\\_INTEGERS](#), [CGL\\_POST\\_INFEASIBLE](#), [CGL\\_POST\\_CHANGED](#), [CGL\\_GENERAL](#),  
[CGL\\_DUMMY\\_END](#) }

### 7.18.1 Enumeration Type Documentation

#### 7.18.1.1 enum CGL\_Message

#### Enumerator

***CGL\_INFEASIBLE***  
***CGL\_CLIQUES***  
***CGL\_FIXED***  
***CGL\_PROCESS\_STATS***  
***CGL\_SLACKS***  
***CGL\_PROCESS\_STATS2***  
***CGL\_PROCESS\_SOS1***  
***CGL\_PROCESS\_SOS2***  
***CGL\_UNBOUNDED***  
***CGL\_ELEMENTS\_CHANGED1***  
***CGL\_ELEMENTS\_CHANGED2***  
***CGL\_MADE\_INTEGER***  
***CGL\_ADDED\_INTEGERS***

***CGL\_POST\_INFEASIBLE***

***CGL\_POST\_CHANGED***

***CGL\_GENERAL***

***CGL\_DUMMY\_END***

Definition at line 15 of file CglMessage.hpp.

## 7.19 src/CglMixedIntegerRounding/CglMixedIntegerRounding.hpp File Reference

```
#include <iostream>
#include <fstream>
#include "CoinError.hpp"
#include "CglCutGenerator.hpp"
```

### Classes

- class [CglMixIntRoundVUB](#)
- class [CglMixedIntegerRounding](#)  
*Mixed Integer Rounding Cut Generator Class.*

### Macros

- #define [CGL\\_DEBUG](#) 0

### Typedefs

- typedef [CglMixIntRoundVUB](#) [CglMixIntRoundVLB](#)

### Functions

- void [CglMixedIntegerRoundingUnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)

#### 7.19.1 Macro Definition Documentation

##### 7.19.1.1 #define CGL\_DEBUG 0

Definition at line 26 of file CglMixedIntegerRounding.hpp.

#### 7.19.2 Typedef Documentation

##### 7.19.2.1 typedef CglMixIntRoundVUB CglMixIntRoundVLB

Definition at line 74 of file CglMixedIntegerRounding.hpp.

### 7.19.3 Function Documentation

7.19.3.1 void CglMixedIntegerRoundingUnitTest ( const OsiSolverInterface \* *siP*, const std::string *mpdDir* )

## 7.20 src/CglMixedIntegerRounding2/CglMixedIntegerRounding2.hpp File Reference

```
#include <iostream>
#include <fstream>
#include "CoinError.hpp"
#include "CglCutGenerator.hpp"
#include "CoinIndexedVector.hpp"
```

### Classes

- class [CglMixIntRoundVUB2](#)
- class [CglMixedIntegerRounding2](#)  
*Mixed Integer Rounding Cut Generator Class.*

### Macros

- #define [CGL\\_DEBUG](#) 0

### Typedefs

- typedef [CglMixIntRoundVUB2](#) [CglMixIntRoundVLB2](#)

### Functions

- void [CglMixedIntegerRounding2UnitTest](#) (const OsiSolverInterface \**siP*, const std::string *mpdDir*)

### 7.20.1 Macro Definition Documentation

7.20.1.1 #define [CGL\\_DEBUG](#) 0

Definition at line 27 of file CglMixedIntegerRounding2.hpp.

### 7.20.2 Typedef Documentation

7.20.2.1 typedef [CglMixIntRoundVUB2](#) [CglMixIntRoundVLB2](#)

Definition at line 75 of file CglMixedIntegerRounding2.hpp.

### 7.20.3 Function Documentation

7.20.3.1 void [CglMixedIntegerRounding2UnitTest](#) ( const OsiSolverInterface \* *siP*, const std::string *mpdDir* )

## 7.21 src/CglOddHole/CglOddHole.hpp File Reference

```
#include <string>
#include "CglCutGenerator.hpp"
```

### Classes

- class [CglOddHole](#)  
*Odd Hole Cut Generator Class.*

### Functions

- void [CglOddHoleUnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)  
*A function that tests the methods in the [CglOddHole](#) class.*

#### 7.21.1 Function Documentation

##### 7.21.1.1 void CglOddHoleUnitTest ( const OsiSolverInterface \* siP, const std::string mpdDir )

A function that tests the methods in the [CglOddHole](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

## 7.22 src/CglParam.hpp File Reference

```
#include "CglConfig.h"
#include "CoinFinite.hpp"
```

### Classes

- class [CglParam](#)  
*Class collecting parameters for all cut generators.*

## 7.23 src/CglPreProcess/CglPreProcess.hpp File Reference

```
#include <string>
#include <vector>
#include "CoinMessageHandler.hpp"
#include "OsiSolverInterface.hpp"
#include "CglStored.hpp"
#include "OsiPresolve.hpp"
#include "CglCutGenerator.hpp"
```

## Classes

- class [CglPreProcess](#)  
*Class for preProcessing and postProcessing.*
- class [CglBK](#)  
*For Bron-Kerbosch.*
- struct [CglHashLink](#)  
*Only store unique row cuts.*
- class [CglUniqueRowCuts](#)

## 7.24 src/CglProbing/CglProbing.hpp File Reference

```
#include <string>
#include "CglCutGenerator.hpp"
```

## Classes

- struct [disaggregationAction](#)  
*Only useful type of disaggregation is most normal For now just done for 0-1 variables Can be used for building cliques.*
- class [CglProbing](#)  
*Probing Cut Generator Class.*
- class [CglImplication](#)  
*This just uses implication info.*

## Functions

- int [affectedInDisaggregation](#) (const [disaggregationAction](#) &dis)
- void [setAffectedInDisaggregation](#) ([disaggregationAction](#) &dis, int affected)
- bool [zeroOneInDisaggregation](#) (const [disaggregationAction](#) &dis)
- void [setZeroOneInDisaggregation](#) ([disaggregationAction](#) &dis, bool zeroOne)
- bool [whenAtUBInDisaggregation](#) (const [disaggregationAction](#) &dis)
- void [setWhenAtUBInDisaggregation](#) ([disaggregationAction](#) &dis, bool whenAtUB)
- bool [affectedToUBInDisaggregation](#) (const [disaggregationAction](#) &dis)
- void [setAffectedToUBInDisaggregation](#) ([disaggregationAction](#) &dis, bool affectedToUB)
- void [CglProbingUnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)  
*A function that tests the methods in the [CglProbing](#) class.*

## 7.24.1 Function Documentation

7.24.1.1 int [affectedInDisaggregation](#) ( const [disaggregationAction](#) & *dis* ) `[inline]`

Definition at line 435 of file CglProbing.hpp.

7.24.1.2 void [setAffectedInDisaggregation](#) ( [disaggregationAction](#) & *dis*, int *affected* ) `[inline]`

Definition at line 437 of file CglProbing.hpp.

7.24.1.3 `bool zeroOneInDisaggregation ( const disaggregationAction & dis )` `[inline]`

Definition at line 444 of file CglProbing.hpp.

7.24.1.4 `void setZeroOneInDisaggregation ( disaggregationAction & dis, bool zeroOne )` `[inline]`

Definition at line 448 of file CglProbing.hpp.

7.24.1.5 `bool whenAtUBInDisaggregation ( const disaggregationAction & dis )` `[inline]`

Definition at line 450 of file CglProbing.hpp.

7.24.1.6 `void setWhenAtUBInDisaggregation ( disaggregationAction & dis, bool whenAtUB )` `[inline]`

Definition at line 452 of file CglProbing.hpp.

7.24.1.7 `bool affectedToUBInDisaggregation ( const disaggregationAction & dis )` `[inline]`

Definition at line 454 of file CglProbing.hpp.

7.24.1.8 `void setAffectedToUBInDisaggregation ( disaggregationAction & dis, bool affectedToUB )` `[inline]`

Definition at line 456 of file CglProbing.hpp.

7.24.1.9 `void CglProbingUnitTest ( const OsiSolverInterface * siP, const std::string mpdDir )`

A function that tests the methods in the [CglProbing](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

## 7.25 src/CglRedSplit/CglRedSplit.hpp File Reference

```
#include "CglCutGenerator.hpp"
#include "CglRedSplitParam.hpp"
```

### Classes

- class [CglRedSplit](#)  
*Gomory Reduce-and-Split Cut Generator Class; See method [generateCuts\(\)](#).*

### Functions

- void [CglRedSplitUnitTest](#) (const OsiSolverInterface \**siP*, const std::string *mpdDir*)  
*A function that tests the methods in the [CglRedSplit](#) class.*

### 7.25.1 Function Documentation

7.25.1.1 `void CglRedSplitUnitTest ( const OsiSolverInterface * siP, const std::string mpdDir )`

A function that tests the methods in the [CglRedSplit](#) class.



The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

## 7.26 src/CglRedSplit/CglRedSplitParam.hpp File Reference

```
#include "CglParam.hpp"
```

### Classes

- class [CglRedSplitParam](#)  
*Class collecting parameters the Reduced-and-split cut generator.*

## 7.27 src/CglRedSplit2/CglRedSplit2.hpp File Reference

```
#include "CglCutGenerator.hpp"
#include "CglRedSplit2Param.hpp"
#include "CoinWarmStartBasis.hpp"
#include "CoinHelperFunctions.hpp"
#include "CoinTime.hpp"
```

### Classes

- class [CglRedSplit2](#)  
*Reduce-and-Split Cut Generator Class; See method [generateCuts\(\)](#).*

### Functions

- void [CglRedSplit2UnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)  
*A function that tests some of the methods in the [CglRedSplit2](#) class.*

### 7.27.1 Function Documentation

#### 7.27.1.1 void CglRedSplit2UnitTest ( const OsiSolverInterface \* siP, const std::string mpdDir )

A function that tests some of the methods in the [CglRedSplit2](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

## 7.28 src/CglRedSplit2/CglRedSplit2Param.hpp File Reference

```
#include "CglParam.hpp"
#include <vector>
```

## Classes

- class [CglRedSplit2Param](#)

*Class collecting parameters the Reduced-and-split cut generator.*

## 7.29 src/CglResidualCapacity/CglResidualCapacity.hpp File Reference

```
#include <iostream>
#include <fstream>
#include "CoinError.hpp"
#include "CglCutGenerator.hpp"
```

## Classes

- class [CglResidualCapacity](#)

*Residual Capacity Inequalities Cut Generator Class.*

## Macros

- #define [CGL\\_DEBUG](#) 0

## Functions

- void [CglResidualCapacityUnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)

*A function that tests the methods in the [CglResidualCapacity](#) class.*

### 7.29.1 Macro Definition Documentation

#### 7.29.1.1 #define CGL\_DEBUG 0

Definition at line 26 of file CglResidualCapacity.hpp.

### 7.29.2 Function Documentation

#### 7.29.2.1 void CglResidualCapacityUnitTest ( const OsiSolverInterface \* siP, const std::string mpdDir )

A function that tests the methods in the [CglResidualCapacity](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

## 7.30 src/CglSimpleRounding/CglSimpleRounding.hpp File Reference

```
#include <string>
#include "CglCutGenerator.hpp"
#include "CoinPackedMatrix.hpp"
```

## Classes

- class [CglSimpleRounding](#)  
*Simple Rounding Cut Generator Class.*

## Functions

- void [CglSimpleRoundingUnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)  
*A function that tests the methods in the [CglSimpleRounding](#) class.*

### 7.30.1 Function Documentation

#### 7.30.1.1 void CglSimpleRoundingUnitTest ( const OsiSolverInterface \* siP, const std::string mpdDir )

A function that tests the methods in the [CglSimpleRounding](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

### 7.31 src/CglStored.hpp File Reference

```
#include <string>
#include "CglCutGenerator.hpp"
```

## Classes

- class [CglStored](#)  
*Stored Cut Generator Class.*

### 7.32 src/CglTreeInfo.hpp File Reference

```
#include "OsiCuts.hpp"
#include "OsiSolverInterface.hpp"
#include "CoinHelperFunctions.hpp"
```

## Classes

- class [CglTreeInfo](#)  
*Information about where the cut generator is invoked from.*
- struct [cliqueEntry](#)  
*Derived class to pick up probing info.*
- class [CglTreeProbingInfo](#)

## Functions

- int [sequenceInCliqueEntry](#) (const [cliqueEntry](#) &cEntry)
- void [setSequenceInCliqueEntry](#) ([cliqueEntry](#) &cEntry, int sequence)
- bool [oneFixesInCliqueEntry](#) (const [cliqueEntry](#) &cEntry)
- void [setOneFixesInCliqueEntry](#) ([cliqueEntry](#) &cEntry, bool oneFixes)

## 7.32.1 Function Documentation

7.32.1.1 int [sequenceInCliqueEntry](#) ( const [cliqueEntry](#) & *cEntry* ) [inline]

Definition at line 169 of file CglTreeInfo.hpp.

7.32.1.2 void [setSequenceInCliqueEntry](#) ( [cliqueEntry](#) & *cEntry*, int *sequence* ) [inline]

Definition at line 171 of file CglTreeInfo.hpp.

7.32.1.3 bool [oneFixesInCliqueEntry](#) ( const [cliqueEntry](#) & *cEntry* ) [inline]

Definition at line 173 of file CglTreeInfo.hpp.

7.32.1.4 void [setOneFixesInCliqueEntry](#) ( [cliqueEntry](#) & *cEntry*, bool *oneFixes* ) [inline]

Definition at line 175 of file CglTreeInfo.hpp.

## 7.33 src/CglTwomir/CglTwomir.hpp File Reference

```
#include <string>
#include "CglCutGenerator.hpp"
#include "CoinFactorization.hpp"
```

## Classes

- struct [DGG\\_constraint\\_t](#)
- struct [DGG\\_list\\_t](#)
- struct [cutParams](#)
- struct [DGG\\_data\\_t](#)
- class [CglTwomir](#)

*Twostep MIR Cut Generator Class.*

## Macros

- #define [DGG\\_isBasic](#)(data, idx) ((data->info[idx])&1)
- #define [DGG\\_isInteger](#)(data, idx) ((data->info[idx] >> 1)&1)
- #define [DGG\\_isStructural](#)(data, idx) ((data->info[idx] >> 2)&1)
- #define [DGG\\_isEqualityConstraint](#)(data, idx) ((data->info[idx] >> 3)&1)
- #define [DGG\\_isNonBasicAtUB](#)(data, idx) ((data->info[idx] >> 4)&1)
- #define [DGG\\_isNonBasicAtLB](#)(data, idx) ((data->info[idx] >> 5)&1)
- #define [DGG\\_isConstraintBoundedAbove](#)(data, idx) ((data->info[idx] >> 6)&1)
- #define [DGG\\_isConstraintBoundedBelow](#)(data, idx) ((data->info[idx] >> 7)&1)

- #define DGG\_setIsBasic(data, idx) ((data->info[idx]) != 1)
- #define DGG\_setIsInteger(data, idx) ((data->info[idx]) != (1 < 1))
- #define DGG\_setIsStructural(data, idx) ((data->info[idx]) != (1 < 2))
- #define DGG\_setEqualityConstraint(data, idx) ((data->info[idx]) != (1 < 3))
- #define DGG\_setIsNonBasicAtUB(data, idx) ((data->info[idx]) != (1 < 4))
- #define DGG\_setIsNonBasicAtLB(data, idx) ((data->info[idx]) != (1 < 5))
- #define DGG\_setIsConstraintBoundedAbove(data, idx) ((data->info[idx]) != (1 < 6))
- #define DGG\_setIsConstraintBoundedBelow(data, idx) ((data->info[idx]) != (1 < 7))
- #define DGG\_DEBUG\_DGG 1
- #define DGG\_TRACE\_ERRORS 0
- #define DGG\_DISPLAY 0
- #define DGG\_AUTO\_CHECK\_CUT\_OFF\_OPTIMAL 1
- #define DGG\_DEFAULT\_METHOD 2
- #define DGG\_DEFAULT\_TMIN 1
- #define DGG\_DEFAULT\_TMAX 1
- #define DGG\_DEFAULT\_TAUMIN 2
- #define DGG\_DEFAULT\_TAUMAX 6
- #define DGG\_DEFAULT\_MAX\_CUTS 500
- #define DGG\_DEFAULT\_IMPROVEMENT\_THRESH 0.001
- #define DGG\_DEFAULT\_NBELOW\_THRESH INT\_MAX
- #define DGG\_DEFAULT\_NROOT\_ROUNDS 2
- #define DGG\_DEFAULT\_NEGATIVE\_SCALED\_TWOSTEPS 0
- #define DGG\_DEFAULT\_ALPHA\_RULE 0
- #define DGG\_DEFAULT\_CUT\_INC 250
- #define DGG\_DEFAULT\_CUT\_FORM 0
- #define DGG\_DEFAULT\_NICEFY 0
- #define DGG\_DEFAULT\_ONLY\_DELAYED 0
- #define DGG\_DEFAULT\_DELAYED\_FREQ 9999999
- #define DGG\_DEFAULT\_LPROWS\_FREQ 9999999
- #define DGG\_DEFAULT\_WHICH\_FORMULATION\_CUTS 2
- #define DGG\_OSI 0
- #define DGG\_CPX 1
- #define DGG\_QSO 2
- #define DGG\_SOLVER DGG\_OSI
- #define DGG\_DEBUG\_SOLVER 0
- #define DGG\_SOLVER\_SCREEN\_FLAG 0
- #define DGG\_TMIR\_CUT 1
- #define DGG\_2STEP\_CUT 2
- #define DGG\_ALPHA\_MIN\_SUM 0
- #define DGG\_ALPHA\_RANDOM\_01 1
- #define DGG\_ALPHA\_RANDOM\_COEFF 2
- #define DGG\_ALPHA\_ALL 3
- #define DGG\_ALPHA\_MAX\_STEEP 5
- #define DGG\_MIN\_STEEPNESS 1.0e-4
- #define DGG\_MAX\_L2NORM 1.0e7
- #define DGG\_NORM\_CRITERIA 1
- #define UB\_MAX DBL\_MAX
- #define DGG\_GOMORY\_THRESH 0.005
- #define DGG\_RHS\_THRESH 0.005
- #define DGG\_BOUND\_THRESH 1.0e-6
- #define DGG\_EQUALITY\_THRESH 1.0e-5

- `#define DGG_SHIFT_THRESH 1.0e-6`
- `#define DGG_INTEGRALITY_THRESH 1.0e-10`
- `#define CBC_CHECK_CUT`
- `#define DGG_MIN_TABLEAU_COEFFICIENT 1.0e-12`
- `#define DGG_MIN_RHO 1.0e-7`
- `#define DGG_MIN_ALPHA 1.0e-7`
- `#define DGG_NULL_SLACK 1.0e-5`
- `#define DGG_NICEFY_MIN_ABSVALUE 1.0e-13`
- `#define DGG_NICEFY_MIN_FIX 1.0e-7`
- `#define DGG_NICEFY_MAX_PADDING 1.0e-6`
- `#define DGG_NICEFY_MAX_RATIO 1.0e9`
- `#define DGG_IF_EXIT(A, B, REST) {if(A) {fprintf(stdout, REST);exit(B);}}`
- `#define DGG_THROW(A, B) return(A)`
- `#define DGG_CHECKRVAL(A, B) { if(A) return(B); }`
- `#define DGG_CHECKRVAL1(A, B) { if(A) { rval = B; goto CLEANUP; } }`
- `#define DGG_TEST(A, B, REST) { if(A) return(B); }`
- `#define DGG_TEST2(A, B, REST, C) { DGG_TEST(A,B,REST) }`
- `#define DGG_TEST3(A, B, REST, C, D) { DGG_TEST(A,B,REST) }`
- `#define DGG_MIN(a, b) ( (a<b)?a:b )`
- `#define DGG_MAX(a, b) ( (a>b)?a:b )`
- `#define KREM(vht, alpha, tau) (DGG_MIN( ceil(vht / alpha), tau ) - 1)`
- `#define LMIN(vht, d, bht) (DGG_MIN( floor(d*bht/bht), d))`
- `#define ABOV(v) (v - floor(v))`
- `#define QINT(vht, bht, tau) ( (int)floor( (vht*(tau-1))/bht ) )`
- `#define V2I(bht, tau, i) ( ((i+1)*bht / tau) )`

## Functions

- `int DGG_is_even (double vht, double bht, int tau, int q)`
- `double frac_part (double value)`
- `int DGG_is_a_multiple_of_b (double a, double b)`
- `int DGG_freeData (DGG_data_t *data)`
- `DGG_constraint_t * DGG_newConstraint (int max_arrays)`
- `void DGG_freeConstraint (DGG_constraint_t *c)`
- `DGG_constraint_t * DGG_copyConstraint (DGG_constraint_t *c)`
- `void DGG_scaleConstraint (DGG_constraint_t *c, int t)`
- `void DGG_list_init (DGG_list_t *l)`
- `int DGG_list_addcut (DGG_list_t *l, DGG_constraint_t *cut, int ctype, double alpha)`
- `void DGG_list_delcut (DGG_list_t *l, int i)`
- `void DGG_list_free (DGG_list_t *l)`
- `DGG_data_t * DGG_getData (const void *solver_ptr)`
- `int DGG_transformConstraint (DGG_data_t *data, double **x_out, double **rc_out, char **isint_out, DGG_constraint_t *constraint)`
- `int DGG_unTransformConstraint (DGG_data_t *data, DGG_constraint_t *constraint)`
- `int DGG_substituteSlacks (const void *solver_ptr, DGG_data_t *data, DGG_constraint_t *cut)`
- `int DGG_nicefyConstraint (const void *solver_ptr, DGG_data_t *data, DGG_constraint_t *cut)`
- `int DGG_getFormulaConstraint (int row_idx, const void *solver_ptr, DGG_data_t *data, DGG_constraint_t *row)`
- `int DGG_getTableauConstraint (int index, const void *solver_ptr, DGG_data_t *data, DGG_constraint_t *tabrow, const int *collsBasic, const int *rowsBasic, CoinFactorization &factorization, int mode)`
- `DGG_constraint_t * DGG_getSlackExpression (const void *solver_ptr, DGG_data_t *data, int row_index)`

- `int DGG_generateTabRowCuts (DGG_list_t *list, DGG_data_t *data, const void *solver_ptr)`
- `int DGG_generateFormulationCuts (DGG_list_t *list, DGG_data_t *data, const void *solver_ptr, int nrows, CoinThreadRandom &generator)`
- `int DGG_generateFormulationCutsFromBase (DGG_constraint_t *base, double slack, DGG_list_t *list, DGG_data_t *data, const void *solver_ptr, CoinThreadRandom &generator)`
- `int DGG_generateCutsFromBase (DGG_constraint_t *base, DGG_list_t *list, DGG_data_t *data, const void *solver_ptr)`
- `int DGG_buildMir (char *isint, DGG_constraint_t *base, DGG_constraint_t **cut_out)`
- `int DGG_build2step (double alpha, char *isint, DGG_constraint_t *base, DGG_constraint_t **cut_out)`
- `int DGG_addMirToList (DGG_constraint_t *base, char *isint, double *x, DGG_list_t *list, DGG_data_t *data, DGG_constraint_t *orig_base)`
- `int DGG_add2stepToList (DGG_constraint_t *base, char *isint, double *x, double *rc, DGG_list_t *list, DGG_data_t *data, DGG_constraint_t *orig_base)`
- `double DGG_cutLHS (DGG_constraint_t *c, double *x)`
- `int DGG_isCutDesirable (DGG_constraint_t *c, DGG_data_t *d)`
- `int DGG_isConstraintViolated (DGG_data_t *d, DGG_constraint_t *c)`
- `int DGG_isBaseTrivial (DGG_data_t *d, DGG_constraint_t *c)`
- `int DGG_is2stepValid (double alpha, double bht)`
- `int DGG_cutsOffPoint (double *x, DGG_constraint_t *cut)`
- `void CglTwomirUnitTest (const OsiSolverInterface *siP, const std::string mpdDir)`

*A function that tests the methods in the `CglTwomir` class.*

### 7.33.1 Macro Definition Documentation

#### 7.33.1.1 `#define DGG_isBasic( data, idx ) ((data->info[idx])&1)`

Definition at line 71 of file `CglTwomir.hpp`.

#### 7.33.1.2 `#define DGG_isInteger( data, idx ) ((data->info[idx] >> 1)&1)`

Definition at line 72 of file `CglTwomir.hpp`.

#### 7.33.1.3 `#define DGG_isStructural( data, idx ) ((data->info[idx] >> 2)&1)`

Definition at line 73 of file `CglTwomir.hpp`.

#### 7.33.1.4 `#define DGG_isEqualityConstraint( data, idx ) ((data->info[idx] >> 3)&1)`

Definition at line 74 of file `CglTwomir.hpp`.

#### 7.33.1.5 `#define DGG_isNonBasicAtUB( data, idx ) ((data->info[idx] >> 4)&1)`

Definition at line 75 of file `CglTwomir.hpp`.

#### 7.33.1.6 `#define DGG_isNonBasicAtLB( data, idx ) ((data->info[idx] >> 5)&1)`

Definition at line 76 of file `CglTwomir.hpp`.

#### 7.33.1.7 `#define DGG_isConstraintBoundedAbove( data, idx ) ((data->info[idx] >> 6)&1)`

Definition at line 77 of file `CglTwomir.hpp`.

#### 7.33.1.8 `#define DGG_isConstraintBoundedBelow( data, idx ) ((data->info[idx] >> 7)&1)`

Definition at line 78 of file `CglTwomir.hpp`.

7.33.1.9 `#define DGG_setIsBasic( data, idx ) ((data->info[idx]) != 1)`

Definition at line 80 of file CglTwomir.hpp.

7.33.1.10 `#define DGG_setIsInteger( data, idx ) ((data->info[idx]) != (1 < 1))`

Definition at line 81 of file CglTwomir.hpp.

7.33.1.11 `#define DGG_setIsStructural( data, idx ) ((data->info[idx]) != (1 < 2))`

Definition at line 82 of file CglTwomir.hpp.

7.33.1.12 `#define DGG_setEqualityConstraint( data, idx ) ((data->info[idx]) != (1 < 3))`

Definition at line 83 of file CglTwomir.hpp.

7.33.1.13 `#define DGG_setIsNonBasicAtUB( data, idx ) ((data->info[idx]) != (1 < 4))`

Definition at line 84 of file CglTwomir.hpp.

7.33.1.14 `#define DGG_setIsNonBasicAtLB( data, idx ) ((data->info[idx]) != (1 < 5))`

Definition at line 85 of file CglTwomir.hpp.

7.33.1.15 `#define DGG_setIsConstraintBoundedAbove( data, idx ) ((data->info[idx]) != (1 < 6))`

Definition at line 86 of file CglTwomir.hpp.

7.33.1.16 `#define DGG_setIsConstraintBoundedBelow( data, idx ) ((data->info[idx]) != (1 < 7))`

Definition at line 87 of file CglTwomir.hpp.

7.33.1.17 `#define DGG_DEBUG_DGG 1`

Definition at line 236 of file CglTwomir.hpp.

7.33.1.18 `#define DGG_TRACE_ERRORS 0`

Definition at line 237 of file CglTwomir.hpp.

7.33.1.19 `#define DGG_DISPLAY 0`

Definition at line 238 of file CglTwomir.hpp.

7.33.1.20 `#define DGG_AUTO_CHECK_CUT_OFF_OPTIMAL 1`

Definition at line 239 of file CglTwomir.hpp.

7.33.1.21 `#define DGG_DEFAULT_METHOD 2`

Definition at line 243 of file CglTwomir.hpp.

7.33.1.22 `#define DGG_DEFAULT_TMIN 1`

Definition at line 244 of file CglTwomir.hpp.



**7.33.1.23 #define DGG\_DEFAULT\_TMAX 1**

Definition at line 245 of file CglTwomir.hpp.

**7.33.1.24 #define DGG\_DEFAULT\_TAUMIN 2**

Definition at line 246 of file CglTwomir.hpp.

**7.33.1.25 #define DGG\_DEFAULT\_TAUMAX 6**

Definition at line 247 of file CglTwomir.hpp.

**7.33.1.26 #define DGG\_DEFAULT\_MAX\_CUTS 500**

Definition at line 248 of file CglTwomir.hpp.

**7.33.1.27 #define DGG\_DEFAULT\_IMPROVEMENT\_THRESH 0.001**

Definition at line 249 of file CglTwomir.hpp.

**7.33.1.28 #define DGG\_DEFAULT\_NBELOW\_THRESH INT\_MAX**

Definition at line 250 of file CglTwomir.hpp.

**7.33.1.29 #define DGG\_DEFAULT\_NROOT\_ROUNDS 2**

Definition at line 251 of file CglTwomir.hpp.

**7.33.1.30 #define DGG\_DEFAULT\_NEGATIVE\_SCALED\_TWOSTEPS 0**

Definition at line 252 of file CglTwomir.hpp.

**7.33.1.31 #define DGG\_DEFAULT\_ALPHA\_RULE 0**

Definition at line 253 of file CglTwomir.hpp.

**7.33.1.32 #define DGG\_DEFAULT\_CUT\_INC 250**

Definition at line 254 of file CglTwomir.hpp.

**7.33.1.33 #define DGG\_DEFAULT\_CUT\_FORM 0**

Definition at line 255 of file CglTwomir.hpp.

**7.33.1.34 #define DGG\_DEFAULT\_NICEFY 0**

Definition at line 256 of file CglTwomir.hpp.

**7.33.1.35 #define DGG\_DEFAULT\_ONLY\_DELAYED 0**

Definition at line 257 of file CglTwomir.hpp.

**7.33.1.36 #define DGG\_DEFAULT\_DELAYED\_FREQ 9999999**

Definition at line 258 of file CglTwomir.hpp.

7.33.1.37 `#define DGG_DEFAULT_LPROWS_FREQ 9999999`

Definition at line 259 of file CglTwomir.hpp.

7.33.1.38 `#define DGG_DEFAULT_WHICH_FORMULATION_CUTS 2`

Definition at line 260 of file CglTwomir.hpp.

7.33.1.39 `#define DGG_OSI 0`

Definition at line 264 of file CglTwomir.hpp.

7.33.1.40 `#define DGG_CPX 1`

Definition at line 265 of file CglTwomir.hpp.

7.33.1.41 `#define DGG_QSO 2`

Definition at line 266 of file CglTwomir.hpp.

7.33.1.42 `#define DGG_SOLVER DGG_OSI`

Definition at line 269 of file CglTwomir.hpp.

7.33.1.43 `#define DGG_DEBUG_SOLVER 0`

Definition at line 272 of file CglTwomir.hpp.

7.33.1.44 `#define DGG_SOLVER_SCREEN_FLAG 0`

Definition at line 275 of file CglTwomir.hpp.

7.33.1.45 `#define DGG_TMIR_CUT 1`

Definition at line 280 of file CglTwomir.hpp.

7.33.1.46 `#define DGG_2STEP_CUT 2`

Definition at line 281 of file CglTwomir.hpp.

7.33.1.47 `#define DGG_ALPHA_MIN_SUM 0`

Definition at line 284 of file CglTwomir.hpp.

7.33.1.48 `#define DGG_ALPHA_RANDOM_01 1`

Definition at line 285 of file CglTwomir.hpp.

7.33.1.49 `#define DGG_ALPHA_RANDOM_COEFF 2`

Definition at line 286 of file CglTwomir.hpp.

7.33.1.50 `#define DGG_ALPHA_ALL 3`

Definition at line 287 of file CglTwomir.hpp.

7.33.1.51 `#define DGG_ALPHA_MAX_STEEP 5`

Definition at line 288 of file CglTwomir.hpp.

7.33.1.52 `#define DGG_MIN_STEEPNESS 1.0e-4`

Definition at line 293 of file CglTwomir.hpp.

7.33.1.53 `#define DGG_MAX_L2NORM 1.0e7`

Definition at line 294 of file CglTwomir.hpp.

7.33.1.54 `#define DGG_NORM_CRITERIA 1`

Definition at line 297 of file CglTwomir.hpp.

7.33.1.55 `#define UB_MAX DBL_MAX`

Definition at line 300 of file CglTwomir.hpp.

7.33.1.56 `#define DGG_GOMORY_THRESH 0.005`

Definition at line 305 of file CglTwomir.hpp.

7.33.1.57 `#define DGG_RHS_THRESH 0.005`

Definition at line 307 of file CglTwomir.hpp.

7.33.1.58 `#define DGG_BOUND_THRESH 1.0e-6`

Definition at line 313 of file CglTwomir.hpp.

7.33.1.59 `#define DGG_EQUALITY_THRESH 1.0e-5`

Definition at line 317 of file CglTwomir.hpp.

7.33.1.60 `#define DGG_SHIFT_THRESH 1.0e-6`

Definition at line 321 of file CglTwomir.hpp.

7.33.1.61 `#define DGG_INTEGRALITY_THRESH 1.0e-10`

Definition at line 326 of file CglTwomir.hpp.

7.33.1.62 `#define CBC_CHECK_CUT`

Definition at line 330 of file CglTwomir.hpp.

7.33.1.63 `#define DGG_MIN_TABLEAU_COEFFICIENT 1.0e-12`

Definition at line 334 of file CglTwomir.hpp.

7.33.1.64 `#define DGG_MIN_RHO 1.0e-7`

Definition at line 339 of file CglTwomir.hpp.

7.33.1.65 `#define DGG_MIN_ALPHA 1.0e-7`

Definition at line 340 of file CglTwomir.hpp.

7.33.1.66 `#define DGG_NULL_SLACK 1.0e-5`

Definition at line 343 of file CglTwomir.hpp.

7.33.1.67 `#define DGG_NICEFY_MIN_ABSVALUE 1.0e-13`

Definition at line 346 of file CglTwomir.hpp.

7.33.1.68 `#define DGG_NICEFY_MIN_FIX 1.0e-7`

Definition at line 347 of file CglTwomir.hpp.

7.33.1.69 `#define DGG_NICEFY_MAX_PADDING 1.0e-6`

Definition at line 348 of file CglTwomir.hpp.

7.33.1.70 `#define DGG_NICEFY_MAX_RATIO 1.0e9`

Definition at line 349 of file CglTwomir.hpp.

7.33.1.71 `#define DGG_IF_EXIT( A, B, REST ) {if(A) {fprintf(stdout, REST);exit(B);}}`

Definition at line 392 of file CglTwomir.hpp.

7.33.1.72 `#define DGG_THROW( A, B ) return(A)`

Definition at line 394 of file CglTwomir.hpp.

7.33.1.73 `#define DGG_CHECKRVAL( A, B ) { if(A) return(B); }`

Definition at line 396 of file CglTwomir.hpp.

7.33.1.74 `#define DGG_CHECKRVAL1( A, B ) { if(A) { rval = B; goto CLEANUP; } }`

Definition at line 397 of file CglTwomir.hpp.

7.33.1.75 `#define DGG_TEST( A, B, REST ) { if(A) return(B); }`

Definition at line 399 of file CglTwomir.hpp.

7.33.1.76 `#define DGG_TEST2( A, B, REST, C ) { DGG_TEST(A,B,REST) }`

Definition at line 400 of file CglTwomir.hpp.

7.33.1.77 `#define DGG_TEST3( A, B, REST, C, D ) { DGG_TEST(A,B,REST) }`

Definition at line 401 of file CglTwomir.hpp.

7.33.1.78 `#define DGG_MIN( a, b ) ( (a<b)?a:b )`

Definition at line 407 of file CglTwomir.hpp.

7.33.1.79 `#define DGG_MAX( a, b ) ((a>b)?a:b)`

Definition at line 408 of file CglTwomir.hpp.

7.33.1.80 `#define KREM( vht, alpha, tau ) (DGG_MIN( ceil(vht / alpha), tau ) - 1)`

Definition at line 409 of file CglTwomir.hpp.

7.33.1.81 `#define LMIN( vht, d, bht ) (DGG_MIN( floor(d*bht/bht), d))`

Definition at line 410 of file CglTwomir.hpp.

7.33.1.82 `#define ABOV( v ) (v - floor(v))`

Definition at line 411 of file CglTwomir.hpp.

7.33.1.83 `#define QINT( vht, bht, tau ) ((int)floor( (vht*(tau-1))/bht ))`

Definition at line 412 of file CglTwomir.hpp.

7.33.1.84 `#define V2I( bht, tau, i ) ((i+1)*bht / tau)`

Definition at line 413 of file CglTwomir.hpp.

## 7.33.2 Function Documentation

7.33.2.1 `int DGG_is_even ( double vht, double bht, int tau, int q )`

7.33.2.2 `double frac_part ( double value )`

7.33.2.3 `int DGG_is_a_multiple_of_b ( double a, double b )`

7.33.2.4 `int DGG_freeData ( DGG_data_t * data )`

7.33.2.5 `DGG_constraint_t* DGG_newConstraint ( int max_arrays )`

7.33.2.6 `void DGG_freeConstraint ( DGG_constraint_t * c )`

7.33.2.7 `DGG_constraint_t* DGG_copyConstraint ( DGG_constraint_t * c )`

7.33.2.8 `void DGG_scaleConstraint ( DGG_constraint_t * c, int t )`

7.33.2.9 `void DGG_list_init ( DGG_list_t * l )`

7.33.2.10 `int DGG_list_addcut ( DGG_list_t * l, DGG_constraint_t * cut, int ctype, double alpha )`

7.33.2.11 `void DGG_list_delcut ( DGG_list_t * l, int i )`

7.33.2.12 `void DGG_list_free ( DGG_list_t * l )`

7.33.2.13 `DGG_data_t* DGG_getData ( const void * solver_ptr )`

7.33.2.14 `int DGG_transformConstraint ( DGG_data_t * data, double ** x_out, double ** rc_out, char ** isint_out, DGG_constraint_t * constraint )`

7.33.2.15 `int DGG_unTransformConstraint ( DGG_data_t * data, DGG_constraint_t * constraint )`

- 7.33.2.16 int DGG\_substituteSlacks ( const void \* *solver\_ptr*, DGG\_data\_t \* *data*, DGG\_constraint\_t \* *cut* )
- 7.33.2.17 int DGG\_nicifyConstraint ( const void \* *solver\_ptr*, DGG\_data\_t \* *data*, DGG\_constraint\_t \* *cut* )
- 7.33.2.18 int DGG\_getFormulaConstraint ( int *row\_idx*, const void \* *solver\_ptr*, DGG\_data\_t \* *data*, DGG\_constraint\_t \* *row* )
- 7.33.2.19 int DGG\_getTableauConstraint ( int *index*, const void \* *solver\_ptr*, DGG\_data\_t \* *data*, DGG\_constraint\_t \* *tabrow*, const int \* *colsBasic*, const int \* *rowsBasic*, CoinFactorization & *factorization*, int *mode* )
- 7.33.2.20 DGG\_constraint\_t\* DGG\_getSlackExpression ( const void \* *solver\_ptr*, DGG\_data\_t \* *data*, int *row\_index* )
- 7.33.2.21 int DGG\_generateTabRowCuts ( DGG\_list\_t \* *list*, DGG\_data\_t \* *data*, const void \* *solver\_ptr* )
- 7.33.2.22 int DGG\_generateFormulationCuts ( DGG\_list\_t \* *list*, DGG\_data\_t \* *data*, const void \* *solver\_ptr*, int *nrows*, CoinThreadRandom & *generator* )
- 7.33.2.23 int DGG\_generateFormulationCutsFromBase ( DGG\_constraint\_t \* *base*, double *slack*, DGG\_list\_t \* *list*, DGG\_data\_t \* *data*, const void \* *solver\_ptr*, CoinThreadRandom & *generator* )
- 7.33.2.24 int DGG\_generateCutsFromBase ( DGG\_constraint\_t \* *base*, DGG\_list\_t \* *list*, DGG\_data\_t \* *data*, const void \* *solver\_ptr* )
- 7.33.2.25 int DGG\_buildMir ( char \* *isint*, DGG\_constraint\_t \* *base*, DGG\_constraint\_t \*\* *cut\_out* )
- 7.33.2.26 int DGG\_build2step ( double *alpha*, char \* *isint*, DGG\_constraint\_t \* *base*, DGG\_constraint\_t \*\* *cut\_out* )
- 7.33.2.27 int DGG\_addMirToList ( DGG\_constraint\_t \* *base*, char \* *isint*, double \* *x*, DGG\_list\_t \* *list*, DGG\_data\_t \* *data*, DGG\_constraint\_t \* *orig\_base* )
- 7.33.2.28 int DGG\_add2stepToList ( DGG\_constraint\_t \* *base*, char \* *isint*, double \* *x*, double \* *rc*, DGG\_list\_t \* *list*, DGG\_data\_t \* *data*, DGG\_constraint\_t \* *orig\_base* )
- 7.33.2.29 double DGG\_cutLHS ( DGG\_constraint\_t \* *c*, double \* *x* )
- 7.33.2.30 int DGG\_isCutDesirable ( DGG\_constraint\_t \* *c*, DGG\_data\_t \* *d* )
- 7.33.2.31 int DGG\_isConstraintViolated ( DGG\_data\_t \* *d*, DGG\_constraint\_t \* *c* )
- 7.33.2.32 int DGG\_isBaseTrivial ( DGG\_data\_t \* *d*, DGG\_constraint\_t \* *c* )
- 7.33.2.33 int DGG\_is2stepValid ( double *alpha*, double *bht* )
- 7.33.2.34 int DGG\_cutsOffPoint ( double \* *x*, DGG\_constraint\_t \* *cut* )
- 7.33.2.35 void CglTwomirUnitTest ( const OsiSolverInterface \* *siP*, const std::string *mpdDir* )

A function that tests the methods in the [CglTwomir](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

## 7.34 src/CglZeroHalf/Cgl012cut.hpp File Reference

```
#include <cstdio>
```

```
#include <cstdlib>
#include <cmath>
```

## Classes

- struct [cgl\\_arc](#)
  - struct [cgl\\_node](#)
  - struct [cgl\\_graph](#)
  - struct [ilp](#)
  - struct [parity\\_ilp](#)
  - struct [info\\_weak](#)
  - struct [edge](#)
  - struct [separation\\_graph](#)
  - struct [auxiliary\\_graph](#)
  - struct [short\\_path\\_node](#)
  - struct [cycle](#)
  - struct [cycle\\_list](#)
  - struct [cut](#)
  - struct [cut\\_list](#)
  - struct [pool\\_cut](#)
  - struct [pool\\_cut\\_list](#)
  - struct [select\\_cut](#)
  - struct [log\\_var](#)
  - class [Cgl012Cut](#)
- 012Cut Generator Class*

## Macros

- `#define` [CGL\\_NEW\\_SHORT](#)
- `#define` [REDUCTION](#)

### 7.34.1 Macro Definition Documentation

#### 7.34.1.1 `#define` CGL\_NEW\_SHORT

Definition at line 12 of file Cgl012cut.hpp.

#### 7.34.1.2 `#define` REDUCTION

Definition at line 58 of file Cgl012cut.hpp.

## 7.35 src/CglZeroHalf/CglZeroHalf.hpp File Reference

```
#include <string>
#include "CglCutGenerator.hpp"
#include "CoinPackedMatrix.hpp"
#include "Cgl012cut.hpp"
```

## Classes

- class [CglZeroHalf](#)  
*Zero Half Cut Generator Class.*

## Functions

- void [cglShortestPath](#) ([auxiliary\\_graph](#) \*graph, int source, int maximumLength)  
*A simple Dijkstra shortest path - make better later.*
- void [CglZeroHalfUnitTest](#) (const OsiSolverInterface \*siP, const std::string mpdDir)  
*A function that tests the methods in the [CglZeroHalf](#) class.*

## 7.35.1 Function Documentation

7.35.1.1 void cglShortestPath ( [auxiliary\\_graph](#) \* graph, int source, int maximumLength )

A simple Dijkstra shortest path - make better later.

## 7.35.1.2 void CglZeroHalfUnitTest ( const OsiSolverInterface \* siP, const std::string mpdDir )

A function that tests the methods in the [CglZeroHalf](#) class.

The only reason for it not to be a member method is that this way it doesn't have to be compiled into the library. And that's a gain, because the library should be compiled with optimization on, but this method should be compiled with debugging.

## 7.36 src/config\_cgl\_default.h File Reference

## Macros

- #define [CGL\\_VERSION](#) "trunk"
- #define [CGL\\_VERSION\\_MAJOR](#) 9999
- #define [CGL\\_VERSION\\_MINOR](#) 9999
- #define [CGL\\_VERSION\\_RELEASE](#) 9999

## 7.36.1 Macro Definition Documentation

## 7.36.1.1 #define CGL\_VERSION "trunk"

Definition at line 8 of file config\_cgl\_default.h.

## 7.36.1.2 #define CGL\_VERSION\_MAJOR 9999

Definition at line 11 of file config\_cgl\_default.h.

## 7.36.1.3 #define CGL\_VERSION\_MINOR 9999

Definition at line 14 of file config\_cgl\_default.h.

## 7.36.1.4 #define CGL\_VERSION\_RELEASE 9999

Definition at line 17 of file config\_cgl\_default.h.



### 7.37 src/config\_default.h File Reference

```
#include "configall_system.h"  
#include "config_cgl_default.h"
```

#### Macros

- `#define COIN_CGL_CHECKLEVEL 0`
- `#define COIN_CGL_VERBOSITY 0`
- `#define COIN_HAS_OSICLP 1`
- `#define COIN_HAS_COINUTILS 1`
- `#define COIN_HAS_OSI 1`
- `#define COIN_HAS_VOL 1`

#### 7.37.1 Macro Definition Documentation

##### 7.37.1.1 `#define COIN_CGL_CHECKLEVEL 0`

Definition at line 14 of file config\_default.h.

##### 7.37.1.2 `#define COIN_CGL_VERBOSITY 0`

Definition at line 17 of file config\_default.h.

##### 7.37.1.3 `#define COIN_HAS_OSICLP 1`

Definition at line 20 of file config\_default.h.

##### 7.37.1.4 `#define COIN_HAS_COINUTILS 1`

Definition at line 23 of file config\_default.h.

##### 7.37.1.5 `#define COIN_HAS_OSI 1`

Definition at line 26 of file config\_default.h.

##### 7.37.1.6 `#define COIN_HAS_VOL 1`

Definition at line 29 of file config\_default.h.

## Index

- ~Cgl012Cut
  - Cgl012Cut, [16](#)
- ~CglAllDifferent
  - CglAllDifferent, [20](#)
- ~CglIBK
  - CglIBK, [22](#)
- ~CglClique
  - CglClique, [26](#)
- ~CglCutGenerator
  - CglCutGenerator, [32](#)
- ~CglDuplicateRow
  - CglDuplicateRow, [37](#)
- ~CglFakeClique
  - CglFakeClique, [41](#)
- ~CglFlowCover
  - CglFlowCover, [44](#)
- ~CglGMI
  - CglGMI, [49](#)
- ~CglGMIParam
  - CglGMIParam, [55](#)
- ~CglGomory
  - CglGomory, [65](#)
- ~CglImplication
  - CglImplication, [69](#)
- ~CglKnapsackCover
  - CglKnapsackCover, [71](#)
- ~CglLandP
  - CglLandP, [76](#)
- ~CglLandPSimplex
  - LAP::CglLandPSimplex, [80](#)
- ~CglLiftAndProject
  - CglLiftAndProject, [86](#)
- ~CglMixIntRoundVUB
  - CglMixIntRoundVUB, [95](#)
- ~CglMixIntRoundVUB2
  - CglMixIntRoundVUB2, [96](#)
- ~CglMixedIntegerRounding
  - CglMixedIntegerRounding, [90](#)
- ~CglMixedIntegerRounding2
  - CglMixedIntegerRounding2, [93](#)
- ~CglOddHole
  - CglOddHole, [99](#)
- ~CglParam
  - CglParam, [102](#)
- ~CglPreProcess
  - CglPreProcess, [106](#)
- ~CglProbing
  - CglProbing, [114](#)
- ~CglRedSplit
  - CglRedSplit, [120](#)
- ~CglRedSplit2
  - CglRedSplit2, [125](#)
- ~CglRedSplit2Param
  - CglRedSplit2Param, [134](#)
- ~CglRedSplitParam
  - CglRedSplitParam, [146](#)
- ~CglResidualCapacity
  - CglResidualCapacity, [152](#)
- ~CglSimpleRounding
  - CglSimpleRounding, [154](#)
- ~CglStored
  - CglStored, [157](#)
- ~CglTreeInfo
  - CglTreeInfo, [161](#)
- ~CglTreeProbingInfo
  - CglTreeProbingInfo, [164](#)
- ~CglTwomir
  - CglTwomir, [169](#)
- ~CglUniqueRowCuts
  - CglUniqueRowCuts, [173](#)
- ~CglZeroHalf
  - CglZeroHalf, [175](#)
- ~Cuts
  - LAP::Cuts, [179](#)
- ~LapMessages
  - LAP::LapMessages, [189](#)
- ~TabRow
  - LAP::TabRow, [202](#)
- a\_max
  - cutParams, [178](#)
- ABOV
  - CglTwomir.hpp, [234](#)
- AWAY
  - CglGMIParam, [61](#)
- addColumnSelectionStrategy
  - CglRedSplit2Param, [137](#)
- addColumnSelectionStrategyLAP
  - CglRedSplit2Param, [137](#)
- addCut
  - CglStored, [157](#), [158](#)
- addCutGenerator
  - CglPreProcess, [110](#)
- addCuts
  - CglUniqueRowCuts, [173](#)
- addNumRowsReduction
  - CglRedSplit2Param, [136](#)
- addNumRowsReductionLAP
  - CglRedSplit2Param, [137](#)
- addRowSelectionStrategy
  - CglRedSplit2Param, [137](#)
- addRowSelectionStrategyLAP

- CglRedSplit2Param, 138
- adjustTableauRow
  - LAP::CglLandPSimplex, 83
- affected
  - disaggregationAction, 185
- affectedInDisaggregation
  - CglProbing.hpp, 220
- affectedToUBInDisaggregation
  - CglProbing.hpp, 221
- aggressive\_
  - CglCutGenerator, 34
- AllViolatedMigs
  - CglLandP, 75
- alloc\_parity\_ilp
  - Cgl012Cut, 16
- alpha
  - DGG\_list\_t, 184
- alternativeFactorization
  - CglGomory, 67
- analyze
  - CglTreeProbingInfo, 164
- arcs
  - auxiliary\_graph, 15
  - cgl\_graph, 18
- areEqual
  - CglGMI, 49
- assignSolver
  - CglFakeClique, 42
- AtOptimalBasis
  - CglLandP, 75
- auxiliary\_graph, 14
  - arcs, 15
  - narcs, 15
  - nnodes, 15
  - nodes, 15
- Average
  - CglLandP, 75
- away
  - CglLandP::Parameters, 193
- away\_
  - CglRedSplit2Param, 141
  - CglRedSplitParam, 150
- BEGIN\_ROUND
  - LAP, 13
- backward
  - CglTreeProbingInfo, 165
- backward\_
  - CglTreeProbingInfo, 166
- bestPivot
  - CglLandP, 74
- bestObjective
  - CglStored, 158
- bestSolution
  - CglStored, 158
- bestSolution\_
  - CglStored, 159
- BigDynamic
  - LAP::Validator, 204
- bounds\_
  - CglStored, 159
- bronKerbosch
  - CglBK, 22
- c
  - DGG\_list\_t, 184
- CGL\_ADDED\_INTEGERS
  - CglMessage.hpp, 216
- CGL\_CLIQUES
  - CglMessage.hpp, 216
- CGL\_DUMMY\_END
  - CglMessage.hpp, 217
- CGL\_ELEMENTS\_CHANGED1
  - CglMessage.hpp, 216
- CGL\_ELEMENTS\_CHANGED2
  - CglMessage.hpp, 216
- CGL\_FIXED
  - CglMessage.hpp, 216
- CGL\_GENERAL
  - CglMessage.hpp, 217
- CGL\_INFEASIBLE
  - CglMessage.hpp, 216
- CGL\_MADE\_INTEGER
  - CglMessage.hpp, 216
- CGL\_POST\_CHANGED
  - CglMessage.hpp, 217
- CGL\_POST\_INFEASIBLE
  - CglMessage.hpp, 216
- CGL\_PROCESS\_SOS1
  - CglMessage.hpp, 216
- CGL\_PROCESS\_SOS2
  - CglMessage.hpp, 216
- CGL\_PROCESS\_STATS
  - CglMessage.hpp, 216
- CGL\_PROCESS\_STATS2
  - CglMessage.hpp, 216
- CGL\_SLACKS
  - CglMessage.hpp, 216
- CGL\_UNBOUNDED
  - CglMessage.hpp, 216
- CGLFLOW\_COL\_BINNEG
  - CglFlowCover.hpp, 208
- CGLFLOW\_COL\_BINPOS
  - CglFlowCover.hpp, 208
- CGLFLOW\_COL\_CONTNEG
  - CglFlowCover.hpp, 208
- CGLFLOW\_COL\_CONTPOS
  - CglFlowCover.hpp, 208

CGLFLOW\_COL\_INCURT  
    CglFlowCover.hpp, [209](#)  
CGLFLOW\_COL\_INCURTDONE  
    CglFlowCover.hpp, [209](#)  
CGLFLOW\_COL\_INLMIN  
    CglFlowCover.hpp, [209](#)  
CGLFLOW\_COL\_INLMINDONE  
    CglFlowCover.hpp, [209](#)  
CGLFLOW\_COL\_INLMINMIN  
    CglFlowCover.hpp, [209](#)  
CGLFLOW\_COL\_OUTCUT  
    CglFlowCover.hpp, [209](#)  
CGLFLOW\_COL\_PRIME  
    CglFlowCover.hpp, [209](#)  
CGLFLOW\_COL\_SECONDARY  
    CglFlowCover.hpp, [209](#)  
CGLFLOW\_ROW\_MIXEQ  
    CglFlowCover.hpp, [209](#)  
CGLFLOW\_ROW\_MIXUB  
    CglFlowCover.hpp, [209](#)  
CGLFLOW\_ROW\_NOBINEQ  
    CglFlowCover.hpp, [209](#)  
CGLFLOW\_ROW\_NOBINUB  
    CglFlowCover.hpp, [209](#)  
CGLFLOW\_ROW\_SUMVAREQ  
    CglFlowCover.hpp, [209](#)  
CGLFLOW\_ROW\_SUMVARUB  
    CglFlowCover.hpp, [209](#)  
CGLFLOW\_ROW\_UNDEFINED  
    CglFlowCover.hpp, [209](#)  
CGLFLOW\_ROW\_UNINTERSTED  
    CglFlowCover.hpp, [209](#)  
CGLFLOW\_ROW\_VAREQ  
    CglFlowCover.hpp, [209](#)  
CGLFLOW\_ROW\_VARLB  
    CglFlowCover.hpp, [209](#)  
CGLFLOW\_ROW\_VARUB  
    CglFlowCover.hpp, [209](#)  
CP\_CGLLANDP1  
    CglGMIParam, [55](#)  
CP\_CGLLANDP1\_INT  
    CglGMIParam, [55](#)  
CP\_CGLLANDP1\_SCALEMAX  
    CglGMIParam, [55](#)  
CP\_CGLLANDP1\_SCALERHS  
    CglGMIParam, [55](#)  
CP\_CGLLANDP2  
    CglGMIParam, [55](#)  
CP\_CGLREDSPLIT  
    CglGMIParam, [55](#)  
CP\_INTEGRAL\_CUTS  
    CglGMIParam, [55](#)  
CS1  
    CglRedSplit2Param, [133](#)  
CS10  
    CglRedSplit2Param, [133](#)  
CS11  
    CglRedSplit2Param, [133](#)  
CS12  
    CglRedSplit2Param, [133](#)  
CS13  
    CglRedSplit2Param, [133](#)  
CS14  
    CglRedSplit2Param, [133](#)  
CS15  
    CglRedSplit2Param, [133](#)  
CS16  
    CglRedSplit2Param, [133](#)  
CS17  
    CglRedSplit2Param, [133](#)  
CS18  
    CglRedSplit2Param, [133](#)  
CS19  
    CglRedSplit2Param, [133](#)  
CS2  
    CglRedSplit2Param, [133](#)  
CS20  
    CglRedSplit2Param, [133](#)  
CS21  
    CglRedSplit2Param, [133](#)  
CS3  
    CglRedSplit2Param, [133](#)  
CS4  
    CglRedSplit2Param, [133](#)  
CS5  
    CglRedSplit2Param, [133](#)  
CS6  
    CglRedSplit2Param, [133](#)  
CS7  
    CglRedSplit2Param, [133](#)  
CS8  
    CglRedSplit2Param, [133](#)  
CS9  
    CglRedSplit2Param, [133](#)  
CS\_ALL  
    CglRedSplit2Param, [133](#)  
CS\_ALLCONT  
    CglRedSplit2Param, [133](#)  
CS\_BEST  
    CglRedSplit2Param, [133](#)  
CS\_LAP\_NONBASICS  
    CglRedSplit2Param, [133](#)  
CUT\_FAILED  
    LAP, [13](#)  
CUT\_GAP  
    LAP, [13](#)  
CUT\_REJECTED  
    LAP, [13](#)

- CBC\_CHECK\_CUT
  - CglTwomir.hpp, 232
- CGL\_DEBUG
  - CglMixedIntegerRounding.hpp, 217
  - CglMixedIntegerRounding2.hpp, 218
  - CglResidualCapacity.hpp, 223
- CGL\_Message
  - CglMessage.hpp, 216
- CGL\_NEW\_SHORT
  - Cgl012cut.hpp, 236
- CGL\_VERSION
  - config\_cgl\_default.h, 237
- CGL\_VERSION\_MAJOR
  - config\_cgl\_default.h, 237
- CGL\_VERSION\_MINOR
  - config\_cgl\_default.h, 237
- CHECK\_DUPLICATES
  - CglGMIParam, 62
- CLEAN\_PROC
  - CglGMIParam, 62
- COIN\_CGL\_VERBOSITY
  - config\_default.h, 238
- COIN\_HAS\_COINUTILS
  - config\_default.h, 238
- COIN\_HAS\_OSI
  - config\_default.h, 238
- COIN\_HAS\_OSICLP
  - config\_default.h, 238
- COIN\_HAS\_VOL
  - config\_default.h, 238
- cacheUpdate
  - LAP::CglLandPSimplex, 80
- canDoGlobalCuts
  - CglCutGenerator, 34
- canDoGlobalCuts\_
  - CglCutGenerator, 34
- ccoef
  - select\_cut, 199
- CftCglp
  - CglLandP, 77
- Cgl012Cut, 15
  - ~Cgl012Cut, 16
  - alloc\_parity\_ilp, 16
  - Cgl012Cut, 16
  - Cgl012Cut, 16
  - free\_ilp, 16
  - free\_log\_var, 16
  - free\_parity\_ilp, 16
  - ilp\_load, 16
  - initialize\_log\_var, 16
  - operator=, 16
  - sep\_012\_cut, 16
- Cgl012cut.hpp
  - CGL\_NEW\_SHORT, 236
- REDUCTION, 236
- CglClique
  - SCL\_MAX\_DEGREE, 25
  - SCL\_MAX\_XJ\_MAX\_DEG, 25
  - SCL\_MIN\_DEGREE, 25
- CglFlowCover.hpp
  - CGLFLOW\_COL\_BINNEG, 208
  - CGLFLOW\_COL\_BINPOS, 208
  - CGLFLOW\_COL\_CONTNEG, 208
  - CGLFLOW\_COL\_CONTPOS, 208
  - CGLFLOW\_COL\_INCUT, 209
  - CGLFLOW\_COL\_INCUTDONE, 209
  - CGLFLOW\_COL\_INLMIN, 209
  - CGLFLOW\_COL\_INLMINDONE, 209
  - CGLFLOW\_COL\_INLMINMIN, 209
  - CGLFLOW\_COL\_OUTCUT, 209
  - CGLFLOW\_COL\_PRIME, 209
  - CGLFLOW\_COL\_SECONDARY, 209
  - CGLFLOW\_ROW\_MIXEQ, 209
  - CGLFLOW\_ROW\_MIXUB, 209
  - CGLFLOW\_ROW\_NOBINEQ, 209
  - CGLFLOW\_ROW\_NOBINUB, 209
  - CGLFLOW\_ROW\_SUMVAREQ, 209
  - CGLFLOW\_ROW\_SUMVARUB, 209
  - CGLFLOW\_ROW\_UNDEFINED, 209
  - CGLFLOW\_ROW\_UNINTERSTED, 209
  - CGLFLOW\_ROW\_VAREQ, 209
  - CGLFLOW\_ROW\_VARLB, 209
  - CGLFLOW\_ROW\_VARUB, 209
- CglGMI
  - failureDynamism, 48
  - failureFractionality, 48
  - failureScale, 49
  - failureSupport, 49
  - failureViolation, 49
- CglGMIParam
  - CP\_CGLLANDP1, 55
  - CP\_CGLLANDP1\_INT, 55
  - CP\_CGLLANDP1\_SCALEMAX, 55
  - CP\_CGLLANDP1\_SCALERHS, 55
  - CP\_CGLLANDP2, 55
  - CP\_CGLREDSPLIT, 55
  - CP\_INTEGRAL\_CUTS, 55
- CglLandP
  - AllViolatedMigs, 75
  - AtOptimalBasis, 75
  - Average, 75
  - bestPivot, 74
  - Dynamic, 76
  - Fixed, 76
  - Fractional, 75
  - Fractional\_rc, 75
  - Full, 75
  - Infinity, 75

- initialReducedCosts, 74
- L1, 75
- L2, 75
- mostNegativeRc, 74
- none, 75
- SupportSize, 75
- Uniform, 75
- Unweighted, 75
- WeightBoth, 75
- WeightLHS, 75
- WeightRHS, 75
- WhenEnteringBasis, 75
- CglMessage.hpp
  - CGL\_ADDED\_INTEGERS, 216
  - CGL\_CLIQUES, 216
  - CGL\_DUMMY\_END, 217
  - CGL\_ELEMENTS\_CHANGED1, 216
  - CGL\_ELEMENTS\_CHANGED2, 216
  - CGL\_FIXED, 216
  - CGL\_GENERAL, 217
  - CGL\_INFEASIBLE, 216
  - CGL\_MADE\_INTEGER, 216
  - CGL\_POST\_CHANGED, 217
  - CGL\_POST\_INFEASIBLE, 216
  - CGL\_PROCESS\_SOS1, 216
  - CGL\_PROCESS\_SOS2, 216
  - CGL\_PROCESS\_STATS, 216
  - CGL\_PROCESS\_STATS2, 216
  - CGL\_SLACKS, 216
  - CGL\_UNBOUNDED, 216
- CglRedSplit2Param
  - CS1, 133
  - CS10, 133
  - CS11, 133
  - CS12, 133
  - CS13, 133
  - CS14, 133
  - CS15, 133
  - CS16, 133
  - CS17, 133
  - CS18, 133
  - CS19, 133
  - CS2, 133
  - CS20, 133
  - CS21, 133
  - CS3, 133
  - CS4, 133
  - CS5, 133
  - CS6, 133
  - CS7, 133
  - CS8, 133
  - CS9, 133
  - CS\_ALL, 133
  - CS\_ALLCONT, 133
  - CS\_BEST, 133
  - CS\_LAP\_NONBASICS, 133
  - RS1, 132
  - RS2, 132
  - RS3, 132
  - RS4, 132
  - RS5, 132
  - RS6, 132
  - RS7, 132
  - RS8, 132
  - RS\_ALL, 132
  - RS\_BEST, 132
  - SC\_LINEAR, 133
  - SC\_LINEAR\_BOUNDED, 133
  - SC\_LOG\_BOUNDED, 133
  - SC\_NONE, 133
  - SC\_UNIFORM, 133
  - SC\_UNIFORM\_NZ, 133
- cgl\_arc, 17
  - length, 17
  - to, 17
- cgl\_graph, 17
  - arcs, 18
  - narcs, 17
  - nnodes, 17
  - nodes, 18
- cgl\_node, 18
  - distanceBack, 18
  - firstArc, 18
  - index, 18
  - parentNode, 18
- CglAllDifferent, 19
  - ~CglAllDifferent, 20
  - CglAllDifferent, 20
  - CglAllDifferent, 20
  - clone, 20
  - generateCpp, 20
  - generateCuts, 20
  - getLogLevel, 21
  - getMaxLook, 21
  - mayGenerateRowCutsInTree, 21
  - operator=, 20
  - refreshSolver, 20
  - setLogLevel, 21
  - setMaxLook, 21
- CglBK, 21
  - ~CglBK, 22
  - bronKerbosch, 22
  - CglBK, 22
  - CglBK, 22
  - newSolver, 22
  - operator=, 22
- CglClique, 23
  - ~CglClique, 26

- CglClique, 25
- CglCliqueUnitTest, 27
- CglClique, 25
- cl\_del\_indices, 30
- cl\_del\_length, 30
- cl\_indices, 29
- cl\_length, 29
- cl\_perm\_indices, 29
- cl\_perm\_length, 29
- clone, 26
- considerRows, 26
- do\_row\_clique, 28
- do\_star\_clique, 28
- fgraph, 28
- frac\_graph, 27
- generateCpp, 26
- generateCuts, 26
- getMinViolation, 27
- justOriginalRows\_, 27
- node\_node, 28
- operator=, 26
- petol, 28
- rcl\_candidate\_length\_threshold, 29
- rcl\_report\_result, 29
- scl\_candidate\_length\_threshold, 29
- scl\_next\_node\_method, 25
- scl\_next\_node\_rule, 29
- scl\_report\_result, 29
- setDoRowClique, 27
- setDoStarClique, 27
- setMinViolation, 27
- setPacking\_, 27
- setRowCliqueCandidateLengthThreshold, 26
- setRowCliqueReport, 27
- setStarCliqueCandidateLengthThreshold, 26
- setStarCliqueNextNodeMethod, 26
- setStarCliqueReport, 26
- sp\_col\_ind, 28
- sp\_col\_start, 28
- sp\_colsol, 28
- sp\_numcols, 28
- sp\_numrows, 27
- sp\_orig\_col\_ind, 28
- sp\_orig\_row\_ind, 27
- sp\_row\_ind, 28
- sp\_row\_start, 28
- CglClique.hpp
  - CglCliqueUnitTest, 206
- CglCliqueUnitTest
  - CglClique, 27
  - CglClique.hpp, 206
- CglCutGenerator, 30
  - ~CglCutGenerator, 32
  - aggressive\_, 34
  - canDoGlobalCuts, 34
  - canDoGlobalCuts\_, 34
  - CglCutGenerator, 32
  - CglCutGenerator, 32
  - clone, 33
  - generateCpp, 33
  - generateCuts, 32
  - getAggressiveness, 33
  - maxLengthOfCutInTree, 34
  - mayGenerateRowCutsInTree, 34
  - needsOptimalBasis, 34
  - operator=, 33
  - refreshSolver, 33
  - setAggressiveness, 33
  - setGlobalCuts, 33
- CglDuplicateRow, 34
  - ~CglDuplicateRow, 37
  - CglDuplicateRow, 37
  - CglDuplicateRow, 37
  - clone, 39
  - duplicate, 37
  - duplicate\_, 39
  - generateCpp, 39
  - generateCuts, 37
  - logLevel, 38
  - logLevel\_, 40
  - lower\_, 39
  - matrix\_, 39
  - matrixByRow\_, 39
  - maximumDominated, 38
  - maximumDominated\_, 39
  - maximumRhs, 38
  - maximumRhs\_, 40
  - mode, 38
  - mode\_, 40
  - numberOriginalRows, 38
  - operator=, 39
  - outDuplicates, 37
  - refreshSolver, 39
  - rhs\_, 39
  - setLogLevel, 38
  - setMaximumDominated, 38
  - setMaximumRhs, 38
  - setMode, 38
  - sizeDynamic, 38
  - sizeDynamic\_, 40
  - storedCuts\_, 39
- CglFakeClique, 40
  - ~CglFakeClique, 41
  - assignSolver, 42
  - CglFakeClique, 41
  - CglFakeClique, 41
  - clone, 42
  - fakeSolver\_, 42

- generateCuts, [42](#)
- operator=, [42](#)
- probing\_, [42](#)
- CglFlowColCut
  - CglFlowCover.hpp, [209](#)
- CglFlowColStatus
  - CglFlowCover.hpp, [208](#)
- CglFlowColType
  - CglFlowCover.hpp, [208](#)
- CglFlowCover, [42](#)
  - ~CglFlowCover, [44](#)
  - CglFlowCover, [44](#)
  - CglFlowCoverUnitTest, [45](#)
  - CglFlowCover, [44](#)
  - clone, [45](#)
  - flowPreprocess, [44](#)
  - generateCpp, [45](#)
  - generateCuts, [44](#)
  - getMaxNumCuts, [44](#)
  - getNumFlowCuts, [44](#)
  - incNumFlowCuts, [45](#)
  - operator=, [45](#)
  - setMaxNumCuts, [44](#)
  - setNumFlowCuts, [44](#)
- CglFlowCover.hpp
  - CglFlowColCut, [209](#)
  - CglFlowColStatus, [208](#)
  - CglFlowColType, [208](#)
  - CglFlowCoverUnitTest, [209](#)
  - CglFlowRowType, [209](#)
  - CglFlowVLB, [208](#)
  - operator<<, [209](#)
- CglFlowCoverUnitTest
  - CglFlowCover, [45](#)
  - CglFlowCover.hpp, [209](#)
- CglFlowRowType
  - CglFlowCover.hpp, [209](#)
- CglFlowVLB
  - CglFlowCover.hpp, [208](#)
- CglFlowVUB, [45](#)
  - CglFlowVUB, [46](#)
  - CglFlowVUB, [46](#)
  - getVal, [46](#)
  - getVar, [46](#)
  - operator=, [46](#)
  - setVal, [46](#)
  - setVar, [46](#)
  - upper\_, [47](#)
  - varInd\_, [46](#)
- CglGMI, [47](#)
  - ~CglGMI, [49](#)
  - areEqual, [49](#)
  - CglGMI, [49](#)
  - CglGMIUnitTest, [51](#)
  - CglGMI, [49](#)
  - clone, [50](#)
  - computesInteger, [50](#)
  - generateCpp, [50](#)
  - generateCuts, [49](#)
  - getNumberGeneratedCuts, [50](#)
  - getNumberRejectedCuts, [50](#)
  - getParam, [50](#)
  - getTrackRejection, [50](#)
  - isIntegerValue, [50](#)
  - isZero, [49](#)
  - needsOptimalBasis, [49](#)
  - operator=, [50](#)
  - printOptTab, [50](#)
  - RejectionType, [48](#)
  - resetRejectionCounters, [50](#)
  - setParam, [50](#)
  - setTrackRejection, [50](#)
- CglGMI.hpp
  - CglGMIUnitTest, [210](#)
- CglGMIParam, [51](#)
  - ~CglGMIParam, [55](#)
  - AWAY, [61](#)
  - CHECK\_DUPLICATES, [62](#)
  - CLEAN\_PROC, [62](#)
  - CglGMIParam, [55](#)
  - CglGMIParam, [55](#)
  - CleaningProcedure, [55](#)
  - clone, [61](#)
  - ENFORCE\_SCALING, [62](#)
  - EPS\_ELIM, [61](#)
  - EPS\_RELAX\_ABS, [61](#)
  - EPS\_RELAX\_REL, [61](#)
  - getAWAY, [57](#)
  - getAway, [57](#)
  - getCHECK\_DUPLICATES, [59](#)
  - getCLEAN\_PROC, [60](#)
  - getCheckDuplicates, [59](#)
  - getCleaningProcedure, [60](#)
  - getENFORCE\_SCALING, [60](#)
  - getEPS\_ELIM, [57](#)
  - getEPS\_RELAX\_ABS, [57](#)
  - getEPS\_RELAX\_REL, [58](#)
  - getEnforcescaling, [61](#)
  - getEps, [56](#)
  - getEpsCoeff, [56](#)
  - getEpsElim, [57](#)
  - getEpsRelaxAbs, [57](#)
  - getEpsRelaxRel, [58](#)
  - getInfinity, [56](#)
  - getIntegralScaleCont, [60](#)
  - getMAX\_SUPPORT\_ABS, [56](#)
  - getMAX\_SUPPORT\_REL, [59](#)
  - getMAXDYN, [58](#)



- getMINVIOL, 58
- getMaxDyn, 58
- getMaxSupport, 56
- getMaxSupportAbs, 56
- getMaxSupportRel, 59
- getMinViol, 58
- getUSE\_INTSLACKS, 59
- getUseIntSlacks, 59
- MAX\_SUPPORT\_REL, 62
- MAXDYN, 61
- MINVIOL, 61
- operator=, 61
- setAWAY, 57
- setAway, 56
- setCHECK\_DUPLICATES, 59
- setCLEAN\_PROC, 60
- setCheckDuplicates, 59
- setCleanProc, 60
- setENFORCE\_SCALING, 60
- setEPS\_ELIM, 57
- setEPS\_RELAX\_ABS, 57
- setEPS\_RELAX\_REL, 57
- setEnforceScaling, 60
- setEps, 56
- setEpsCoeff, 56
- setEpsElim, 57
- setEpsRelaxAbs, 57
- setEpsRelaxRel, 58
- setInfinity, 55
- setIntegralScaleCont, 60
- setMAX\_SUPPORT\_REL, 58
- setMAXDYN, 58
- setMINVIOL, 58
- setMaxDyn, 58
- setMaxSupport, 56
- setMaxSupportAbs, 56
- setMaxSupportRel, 59
- setMinViol, 58
- setUSE\_INTSLACKS, 59
- setUseIntSlacks, 59
- USE\_INTSLACKS, 62
- CglGMIUnitTest
  - CglGMI, 51
  - CglGMI.hpp, 210
- CglGomory, 62
  - ~CglGomory, 65
  - alternativeFactorization, 67
  - CglGomory, 65
  - CglGomoryUnitTest, 67
  - CglGomory, 65
  - clone, 67
  - generateCpp, 67
  - generateCuts, 65
  - getAway, 66
  - getAwayAtRoot, 66
  - getConditionNumberMultiplier, 67
  - getLargestFactorMultiplier, 67
  - getLimit, 66
  - getLimitAtRoot, 66
  - gomoryType, 66
  - maxLengthOfCutInTree, 66
  - needsOptimalBasis, 65
  - operator=, 67
  - originalSolver, 66
  - passInOriginalSolver, 65
  - refreshSolver, 67
  - setAway, 66
  - setAwayAtRoot, 66
  - setConditionNumberMultiplier, 66
  - setGomoryType, 66
  - setLargestFactorMultiplier, 67
  - setLimit, 66
  - setLimitAtRoot, 66
  - useAlternativeFactorization, 67
- CglGomory.hpp
  - CglGomoryUnitTest, 211
- CglGomoryUnitTest
  - CglGomory, 67
  - CglGomory.hpp, 211
- CglHashLink, 68
  - index, 68
  - next, 68
- CglImplication, 68
  - ~CglImplication, 69
  - CglImplication, 69
  - CglImplication, 69
  - clone, 70
  - generateCpp, 70
  - generateCuts, 70
  - operator=, 70
  - setProbingInfo, 70
- CglKnapsackCover, 70
  - ~CglKnapsackCover, 71
  - CglKnapsackCover, 71
  - CglKnapsackCoverUnitTest, 73
  - CglKnapsackCover, 71
  - clone, 72
  - generateCpp, 72
  - generateCuts, 72
  - getMaxInKnapsack, 72
  - operator=, 72
  - refreshSolver, 72
  - setMaxInKnapsack, 72
  - setTestedRowIndices, 72
  - switchOffExpensive, 72
  - switchOnExpensive, 72
- CglKnapsackCover.hpp
  - CglKnapsackCoverUnitTest, 211

- CglKnapsackCoverUnitTest
  - CglKnapsackCover, [73](#)
  - CglKnapsackCover.hpp, [211](#)
- CglLandP, [73](#)
  - ~CglLandP, [76](#)
  - CftCglp, [77](#)
  - CglLandP, [76](#)
  - CglLandPUnitTest, [77](#)
  - CglLandP, [76](#)
  - clone, [76](#)
  - ExtraCutsMode, [74](#)
  - generateCuts, [76](#)
  - LAP::CglLandPSimplex, [77](#)
  - LHSnorm, [75](#)
  - needsOptimalBasis, [76](#)
  - Normalization, [75](#)
  - operator=, [76](#)
  - parameter, [77](#)
  - RhsWeightType, [75](#)
  - SelectionRules, [74](#)
  - SeparationSpaces, [75](#)
  - setLogLevel, [76](#)
  - validator, [76](#)
- CglLandP.hpp
  - CglLandPUnitTest, [212](#)
- CglLandP::NoBasisError, [190](#)
  - NoBasisError, [190](#)
- CglLandP::Parameters, [191](#)
  - away, [193](#)
  - countMistakenRc, [194](#)
  - degeneratePivotLimit, [193](#)
  - extraCutsLimit, [193](#)
  - failedPivotLimit, [193](#)
  - generateExtraCuts, [194](#)
  - lhs\_norm, [194](#)
  - maxCutPerRound, [193](#)
  - modularize, [194](#)
  - normalization, [194](#)
  - operator=, [192](#)
  - Parameters, [192](#)
  - perturb, [194](#)
  - pivotLimit, [193](#)
  - pivotLimitInTree, [193](#)
  - pivotSelection, [195](#)
  - pivotTol, [193](#)
  - rhsWeight, [194](#)
  - rhsWeightType, [194](#)
  - sepSpace, [194](#)
  - singleCutTimeLimit, [193](#)
  - strengthen, [194](#)
  - timeLimit, [193](#)
  - useTableauRow, [194](#)
- CglLandP::SimplexInterfaceError, [200](#)
  - SimplexInterfaceError, [201](#)
- CglLandPSimplex
  - LAP::CglLandPSimplex, [80](#)
- CglLandPSimplex.hpp
  - OLD\_COMPUTATION, [214](#)
- CglLandPUnitTest
  - CglLandP, [77](#)
  - CglLandP.hpp, [212](#)
- CglLiftAndProject, [85](#)
  - ~CglLiftAndProject, [86](#)
  - CglLiftAndProject, [86](#)
  - CglLiftAndProjectUnitTest, [87](#)
  - CglLiftAndProject, [86](#)
  - clone, [87](#)
  - generateCpp, [87](#)
  - generateCuts, [86](#)
  - getBeta, [87](#)
  - operator=, [87](#)
  - setBeta, [87](#)
- CglLiftAndProject.hpp
  - CglLiftAndProjectUnitTest, [216](#)
- CglLiftAndProjectUnitTest
  - CglLiftAndProject, [87](#)
  - CglLiftAndProject.hpp, [216](#)
- CglMessage, [87](#)
  - CglMessage, [88](#)
  - CglMessage, [88](#)
- CglMessage.hpp
  - CGL\_Message, [216](#)
- CglMixIntRoundVLB
  - CglMixedIntegerRounding.hpp, [217](#)
- CglMixIntRoundVLB2
  - CglMixedIntegerRounding2.hpp, [218](#)
- CglMixIntRoundVUB, [94](#)
  - ~CglMixIntRoundVUB, [95](#)
  - CglMixIntRoundVUB, [95](#)
  - CglMixIntRoundVUB, [95](#)
  - getVal, [95](#)
  - getVar, [95](#)
  - operator=, [95](#)
  - setVal, [95](#)
  - setVar, [95](#)
  - val\_, [96](#)
  - var\_, [95](#)
- CglMixIntRoundVUB2, [96](#)
  - ~CglMixIntRoundVUB2, [96](#)
  - CglMixIntRoundVUB2, [96](#)
  - CglMixIntRoundVUB2, [96](#)
  - getVal, [97](#)
  - getVar, [97](#)
  - operator=, [97](#)
  - setVal, [97](#)
  - setVar, [97](#)
  - val\_, [97](#)
  - var\_, [97](#)

CglMixedIntegerRounding, 88  
   ~CglMixedIntegerRounding, 90  
   CglMixedIntegerRounding, 89, 90  
   CglMixedIntegerRoundingUnitTest, 91  
   CglMixedIntegerRounding, 89, 90  
   clone, 90  
   generateCpp, 90  
   generateCuts, 90  
   getCRITERION\_, 91  
   getDoPreproc, 91  
   getMAXAGGR\_, 90  
   getMULTIPLY\_, 91  
   operator=, 90  
   refreshSolver, 90  
   setCRITERION\_, 91  
   setDoPreproc, 91  
   setMAXAGGR\_, 90  
   setMULTIPLY\_, 90  
 CglMixedIntegerRounding.hpp  
   CGL\_DEBUG, 217  
   CglMixIntRoundVLB, 217  
   CglMixedIntegerRoundingUnitTest, 218  
 CglMixedIntegerRounding2, 91  
   ~CglMixedIntegerRounding2, 93  
   CglMixedIntegerRounding2, 93  
   CglMixedIntegerRounding2UnitTest, 94  
   CglMixedIntegerRounding2, 93  
   clone, 93  
   generateCpp, 93  
   generateCuts, 93  
   getCRITERION\_, 94  
   getDoPreproc, 94  
   getMAXAGGR\_, 93  
   getMULTIPLY\_, 94  
   operator=, 93  
   refreshSolver, 93  
   setCRITERION\_, 94  
   setDoPreproc, 94  
   setMAXAGGR\_, 93  
   setMULTIPLY\_, 94  
 CglMixedIntegerRounding2.hpp  
   CGL\_DEBUG, 218  
   CglMixIntRoundVLB2, 218  
   CglMixedIntegerRounding2UnitTest, 218  
 CglMixedIntegerRounding2UnitTest  
   CglMixedIntegerRounding2, 94  
   CglMixedIntegerRounding2.hpp, 218  
 CglMixedIntegerRoundingUnitTest  
   CglMixedIntegerRounding, 91  
   CglMixedIntegerRounding.hpp, 218  
 CglOddHole, 97  
   ~CglOddHole, 99  
   CglOddHole, 99  
   CglOddHoleUnitTest, 100  
   CglOddHole, 99  
   clone, 100  
   createCliqueList, 99  
   createRowList, 99  
   generateCuts, 99  
   getMaximumEntries, 100  
   getMinimumViolation, 100  
   getMinimumViolationPer, 100  
   numberPossible, 99  
   operator=, 100  
   refreshSolver, 100  
   setMaximumEntries, 100  
   setMinimumViolation, 100  
   setMinimumViolationPer, 100  
 CglOddHole.hpp  
   CglOddHoleUnitTest, 219  
 CglOddHoleUnitTest  
   CglOddHole, 100  
   CglOddHole.hpp, 219  
 CglParam, 100  
   ~CglParam, 102  
   CglParam, 102  
   CglParam, 102  
   clone, 103  
   EPS, 103  
   EPS\_COEFF, 103  
   getEPS, 102  
   getEPS\_COEFF, 102  
   getINFINIT, 102  
   getMAX\_SUPPORT, 103  
   INFINIT, 103  
   MAX\_SUPPORT, 103  
   operator=, 103  
   setEPS, 102  
   setEPS\_COEFF, 102  
   setINFINIT, 102  
   setMAX\_SUPPORT, 102  
 CglPreProcess, 103  
   ~CglPreProcess, 106  
   addCutGenerator, 110  
   CglPreProcess, 106  
   CglPreProcess, 106  
   cliquelt, 107  
   cutGenerator, 110  
   cutGenerators, 109  
   cuts, 109  
   cutsPointer, 109  
   getApplicationData, 110  
   getCutoff, 107  
   gutsOfDestructor, 110  
   messageHandler, 110  
   messages, 110  
   messagesPointer, 110  
   modelAtPass, 107

- modifiedModel, 108
- newLanguage, 110
- numberCutGenerators, 109
- numberIterationsPost, 109
- numberIterationsPre, 109
- numberSOS, 108
- operator=, 110
- originalColumns, 108
- originalModel, 107
- originalRows, 108
- passInMessageHandler, 110
- passInProhibited, 108
- passInRowTypes, 109
- postProcess, 107
- preProcess, 106
- preProcessNonDefault, 106
- presolve, 108
- prohibited, 108
- reducedCostFix, 107
- rowTypes, 109
- setApplicationData, 110
- setCutoff, 107
- setLanguage, 110
- setOptions, 109
- someFixed, 107
- startModel, 107
- startSOS, 108
- tightenPrimalBounds, 107
- typeSOS, 108
- update, 109
- weightSOS, 108
- whichSOS, 108
- CglProbing, 111
  - ~CglProbing, 114
  - CglProbing, 114
  - CglProbing::disaggregation\_struct\_tag, 118
  - CglProbingUnitTest, 118
  - CglProbing, 114
  - clone, 117
  - createCliques, 115
  - deleteCliques, 115
  - deleteSnapshot, 115
  - generateCpp, 118
  - generateCuts, 114
  - generateCutsAndModify, 115
  - getLogLevel, 116
  - getMaxElements, 116
  - getMaxElementsRoot, 117
  - getMaxLook, 116
  - getMaxLookRoot, 117
  - getMaxPass, 116
  - getMaxPassRoot, 116
  - getMaxProbe, 116
  - getMaxProbeRoot, 116
  - getMode, 115
  - getUsingObjective, 117
  - lookedAt, 117
  - mayGenerateRowCutsInTree, 117
  - numberThisTime, 117
  - operator=, 118
  - refreshSolver, 118
  - relaxedRowLower, 115
  - relaxedRowUpper, 115
  - rowCuts, 117
  - setLogLevel, 116
  - setMaxElements, 116
  - setMaxElementsRoot, 117
  - setMaxLook, 116
  - setMaxLookRoot, 116
  - setMaxPass, 115
  - setMaxPassRoot, 116
  - setMaxProbe, 116
  - setMaxProbeRoot, 116
  - setMode, 115
  - setRowCuts, 117
  - setUsingObjective, 117
  - snapshot, 115
  - tightLower, 115
  - tightUpper, 115
  - tightenBounds, 115
  - tightenThese, 117
- CglProbing.hpp
  - affectedInDisaggregation, 220
  - affectedToUBInDisaggregation, 221
  - CglProbingUnitTest, 221
  - setAffectedInDisaggregation, 220
  - setAffectedToUBInDisaggregation, 221
  - setWhenAtUBInDisaggregation, 221
  - setZeroOneInDisaggregation, 221
  - whenAtUBInDisaggregation, 221
  - zeroOneInDisaggregation, 220
- CglProbing::disaggregation\_struct\_tag
  - CglProbing, 118
- CglProbingUnitTest
  - CglProbing, 118
  - CglProbing.hpp, 221
- CglRedSplit, 118
  - ~CglRedSplit, 120
  - CglRedSplit, 120
  - CglRedSplitUnitTest, 123
  - CglRedSplit, 120
  - clone, 123
  - compute\_is\_integer, 121
  - compute\_is\_lub, 121
  - generateCpp, 123
  - generateCuts, 121
  - getAway, 122
  - getEPS, 122

- getEPS\_COEFF, 122
- getEPS\_COEFF\_LUB, 122
- getEPS\_RELAX, 122
- getLUB, 122
- getLimit, 122
- getMaxTab, 123
- getMinReduc, 123
- getNormIsZero, 123
- getParam, 121
- needsOptimalBasis, 121
- operator=, 123
- print, 121
- printOptTab, 121
- set\_given\_optsol, 121
- setAway, 122
- setEPS, 122
- setEPS\_COEFF, 122
- setEPS\_COEFF\_LUB, 122
- setEPS\_RELAX, 122
- setLUB, 122
- setLimit, 121
- setMaxTab, 123
- setMinReduc, 123
- setNormIsZero, 122
- setParam, 121
- CglRedSplit.hpp
  - CglRedSplitUnitTest, 221
- CglRedSplit2, 124
  - ~CglRedSplit2, 125
  - CglRedSplit2, 125
  - CglRedSplit2UnitTest, 126
  - CglRedSplit2, 125
  - clone, 126
  - generateCuts, 125
  - generateMultipliers, 126
  - getParam, 126
  - needsOptimalBasis, 126
  - operator=, 126
  - print, 126
  - printOptTab, 126
  - setParam, 126
  - tiltLandPcut, 126
- CglRedSplit2.hpp
  - CglRedSplit2UnitTest, 222
- CglRedSplit2Param, 127
  - ~CglRedSplit2Param, 134
  - addColumnSelectionStrategy, 137
  - addColumnSelectionStrategyLAP, 137
  - addNumRowsReduction, 136
  - addNumRowsReductionLAP, 137
  - addRowSelectionStrategy, 137
  - addRowSelectionStrategyLAP, 138
  - away\_, 141
  - CglRedSplit2Param, 134
  - CglRedSplit2Param, 134
  - clone, 139
  - columnScalingBoundLAP\_, 142
  - ColumnScalingStrategy, 133
  - columnScalingStrategyLAP\_, 141
  - ColumnSelectionStrategy, 132
  - columnSelectionStrategy\_, 141
  - columnSelectionStrategyLAP\_, 141
  - EPS\_ELIM, 140
  - EPS\_RELAX\_ABS, 140
  - EPS\_RELAX\_REL, 140
  - getAway, 134
  - getColumnScalingBoundLAP, 138
  - getColumnScalingStrategyLAP, 138
  - getColumnSelectionStrategy, 137
  - getColumnSelectionStrategyLAP, 138
  - getEPS\_ELIM, 134
  - getEPS\_RELAX\_ABS, 135
  - getEPS\_RELAX\_REL, 135
  - getMAX\_SUPP\_ABS, 135
  - getMAX\_SUPP\_REL, 135
  - getMAXDYN, 135
  - getMINVIOL, 135
  - getMaxNonzeroesTab, 139
  - getMaxNumComputedCuts, 139
  - getMaxNumCuts, 139
  - getMaxSumMultipliers, 136
  - getMinNormReduction, 136
  - getNormIsZero, 136
  - getNormalization, 136
  - getNumRowsReduction, 136
  - getNumRowsReductionLAP, 137
  - getRowSelectionStrategy, 137
  - getRowSelectionStrategyLAP, 138
  - getSkipGomory, 139
  - getTimeLimit, 138
  - getUSE\_INTSLACKS, 136
  - MAX\_SUPP\_REL, 140
  - MAXDYN, 140
  - MINVIOL, 140
  - maxNonzeroesTab\_, 142
  - maxNumComputedCuts\_, 142
  - maxNumCuts\_, 142
  - maxSumMultipliers\_, 140
  - minNormReduction\_, 140
  - normIsZero\_, 140
  - normalization\_, 141
  - numRowsReduction\_, 141
  - numRowsReductionLAP\_, 141
  - operator=, 139
  - resetColumnSelectionStrategy, 137
  - resetColumnSelectionStrategyLAP, 138
  - resetNumRowsReduction, 137
  - resetNumRowsReductionLAP, 137

- resetRowSelectionStrategy, 137
- resetRowSelectionStrategyLAP, 138
- RowSelectionStrategy, 132
- rowSelectionStrategy\_, 141
- rowSelectionStrategyLAP\_, 141
- setAway, 134
- setColumnScalingBoundLAP, 138
- setColumnScalingStrategyLAP, 138
- setEPS\_ELIM, 134
- setEPS\_RELAX\_ABS, 134
- setEPS\_RELAX\_REL, 135
- setMAX\_SUPP\_ABS, 135
- setMAX\_SUPP\_REL, 135
- setMaxDYN, 135
- setMINVIOL, 135
- setMaxNonzeroesTab, 139
- setMaxNumComputedCuts, 139
- setMaxNumCuts, 139
- setMaxSumMultipliers, 136
- setMinNormReduction, 136
- setNormlsZero, 136
- setNormalization, 136
- setSkipGomory, 139
- setTimeLimit, 138
- setUSE\_INTSLACKS, 135
- skipGomory\_, 142
- timeLimit\_, 142
- USE\_INTSLACKS, 140
- CglRedSplit2UnitTest
  - CglRedSplit2, 126
  - CglRedSplit2.hpp, 222
- CglRedSplitParam, 142
  - ~CglRedSplitParam, 146
  - away\_, 150
  - CglRedSplitParam, 145
  - CglRedSplitParam, 145
  - clone, 148
  - EPS\_COEFF\_LUB, 149
  - EPS\_ELIM, 148
  - EPS\_RELAX\_ABS, 149
  - EPS\_RELAX\_REL, 149
  - getAway, 146
  - getEPS\_COEFF\_LUB, 147
  - getEPS\_ELIM, 146
  - getEPS\_RELAX\_ABS, 146
  - getEPS\_RELAX\_REL, 146
  - getLUB, 146
  - getMAXDYN, 147
  - getMAXDYN\_LUB, 147
  - getMINVIOL, 147
  - getMaxTab, 148
  - getMinReduc, 148
  - getNormlsZero, 148
  - getUSE\_CG2, 147
  - getUSE\_INTSLACKS, 147
  - LUB, 148
  - MAXDYN, 149
  - MAXDYN\_LUB, 149
  - MINVIOL, 149
  - maxTab\_, 150
  - minReduc, 150
  - normlsZero, 149
  - operator=, 148
  - setAway, 146
  - setEPS\_COEFF\_LUB, 147
  - setEPS\_ELIM, 146
  - setEPS\_RELAX\_ABS, 146
  - setEPS\_RELAX\_REL, 146
  - setLUB, 146
  - setMaxDYN, 147
  - setMaxDYN\_LUB, 147
  - setMINVIOL, 147
  - setMaxTab, 148
  - setMinReduc, 148
  - setNormlsZero, 148
  - setUSE\_CG2, 147
  - setUSE\_INTSLACKS, 147
  - USE\_CG2, 149
  - USE\_INTSLACKS, 149
- CglRedSplitUnitTest
  - CglRedSplit, 123
  - CglRedSplit.hpp, 221
- CglResidualCapacity, 150
  - ~CglResidualCapacity, 152
  - CglResidualCapacity, 151, 152
  - CglResidualCapacityUnitTest, 153
  - CglResidualCapacity, 151, 152
  - clone, 152
  - generateCuts, 152
  - getDoPreproc, 152
  - getEpsilon, 152
  - getTolerance, 152
  - operator=, 152
  - refreshPrep, 152
  - setDoPreproc, 152
  - setEpsilon, 152
  - setTolerance, 152
- CglResidualCapacity.hpp
  - CGL\_DEBUG, 223
  - CglResidualCapacityUnitTest, 223
- CglResidualCapacityUnitTest
  - CglResidualCapacity, 153
  - CglResidualCapacity.hpp, 223
- cglShortestPath
  - CglZeroHalf.hpp, 237
- CglSimpleRounding, 153
  - ~CglSimpleRounding, 154
  - CglSimpleRounding, 154

- CglSimpleRoundingUnitTest, 155
- CglSimpleRounding, 154
- clone, 154
- generateCpp, 154
- generateCuts, 154
- operator=, 154
- CglSimpleRounding.hpp
  - CglSimpleRoundingUnitTest, 224
- CglSimpleRoundingUnitTest
  - CglSimpleRounding, 155
  - CglSimpleRounding.hpp, 224
- CglStored, 155
  - ~CglStored, 157
  - addCut, 157, 158
  - bestObjective, 158
  - bestSolution, 158
  - bestSolution\_, 159
  - bounds\_, 159
  - CglStored, 157
  - CglStored, 157
  - clone, 158
  - cuts\_, 159
  - generateCuts, 157
  - getRequiredViolation, 157
  - numberColumns\_, 159
  - operator=, 158
  - probingInfo\_, 159
  - requiredViolation\_, 159
  - rowCutPointer, 158
  - saveStuff, 158
  - setProbingInfo, 157
  - setRequiredViolation, 157
  - sizeRowCuts, 158
  - tightLower, 158
  - tightUpper, 158
- CglTreeInfo, 159
  - ~CglTreeInfo, 161
  - CglTreeInfo, 160
  - CglTreeInfo, 160
  - clone, 161
  - fixes, 161
  - formulation\_rows, 161
  - inTree, 162
  - initializeFixing, 161
  - level, 161
  - operator=, 161
  - options, 161
  - pass, 161
  - randomNumberGenerator, 162
  - strengthenRow, 162
- CglTreeInfo.hpp
  - oneFixesInCliqueEntry, 225
  - sequenceInCliqueEntry, 225
  - setOneFixesInCliqueEntry, 225
  - setSequenceInCliqueEntry, 225
- CglTreeProbingInfo, 162
  - ~CglTreeProbingInfo, 164
  - analyze, 164
  - backward, 165
  - backward\_, 166
  - CglTreeProbingInfo, 164
  - CglTreeProbingInfo, 164
  - clone, 164
  - fixColumns, 165
  - fixEntries, 165
  - fixEntry\_, 166
  - fixes, 164
  - fixingEntry\_, 166
  - generateCuts, 165
  - initializeFixing, 165
  - integerVariable, 165
  - integerVariable\_, 166
  - maximumEntries\_, 166
  - numberEntries\_, 166
  - numberIntegers, 165
  - numberIntegers\_, 166
  - numberVariables, 165
  - numberVariables\_, 166
  - operator=, 164
  - packDown, 165
  - toOne, 165
  - toOne\_, 166
  - toZero, 165
  - toZero\_, 166
- CglTwomir, 167
  - ~CglTwomir, 169
  - CglTwomir, 169
  - CglTwomirUnitTest, 172
  - CglTwomir, 169
  - clone, 171
  - generateCpp, 171
  - generateCuts, 169
  - getAmax, 170
  - getAway, 170
  - getAwayAtRoot, 171
  - getIfFormulation, 170
  - getIfMir, 170
  - getIfTableau, 170
  - getIfTwomir, 170
  - getMaxElements, 170
  - getMaxElementsRoot, 170
  - getQmax, 170
  - getQmin, 170
  - getTmax, 170
  - getTmin, 170
  - maxLengthOfCutInTree, 171
  - needsOptimalBasis, 169
  - operator=, 171



- originalSolver, [171](#)
- passInOriginalSolver, [171](#)
- probname\_, [172](#)
- refreshSolver, [171](#)
- setAMax, [169](#)
- setAway, [170](#)
- setAwayAtRoot, [171](#)
- setCutTypes, [169](#)
- setFormulationRows, [170](#)
- setMaxElements, [169](#)
- setMaxElementsRoot, [169](#)
- setMirScale, [169](#)
- setTwomirScale, [169](#)
- setTwomirType, [171](#)
- twomirType, [171](#)
- CglTwomir.hpp
  - ABOV, [234](#)
  - CBC\_CHECK\_CUT, [232](#)
  - CglTwomirUnitTest, [235](#)
  - DGG\_2STEP\_CUT, [231](#)
  - DGG\_ALPHA\_ALL, [231](#)
  - DGG\_ALPHA\_MIN\_SUM, [231](#)
  - DGG\_ALPHA\_RANDOM\_01, [231](#)
  - DGG\_BOUND\_THRESH, [232](#)
  - DGG\_CHECKRVAL, [233](#)
  - DGG\_CHECKRVAL1, [233](#)
  - DGG\_CPX, [231](#)
  - DGG\_DEBUG\_DGG, [229](#)
  - DGG\_DEBUG\_SOLVER, [231](#)
  - DGG\_DEFAULT\_METHOD, [229](#)
  - DGG\_DEFAULT\_NICEFY, [230](#)
  - DGG\_DEFAULT\_TAUMAX, [230](#)
  - DGG\_DEFAULT\_TAUMIN, [230](#)
  - DGG\_DEFAULT\_TMAX, [229](#)
  - DGG\_DEFAULT\_TMIN, [229](#)
  - DGG\_DISPLAY, [229](#)
  - DGG\_GOMORY\_THRESH, [232](#)
  - DGG\_IF\_EXIT, [233](#)
  - DGG\_MAX, [233](#)
  - DGG\_MAX\_L2NORM, [232](#)
  - DGG\_MIN, [233](#)
  - DGG\_MIN\_ALPHA, [232](#)
  - DGG\_MIN\_RHO, [232](#)
  - DGG\_MIN\_STEEPNESS, [232](#)
  - DGG\_NORM\_CRITERIA, [232](#)
  - DGG\_NULL\_SLACK, [233](#)
  - DGG\_OSI, [231](#)
  - DGG\_QSO, [231](#)
  - DGG\_RHS\_THRESH, [232](#)
  - DGG\_SHIFT\_THRESH, [232](#)
  - DGG\_SOLVER, [231](#)
  - DGG\_TEST, [233](#)
  - DGG\_TEST2, [233](#)
  - DGG\_TEST3, [233](#)
  - DGG\_THROW, [233](#)
  - DGG\_TMIR\_CUT, [231](#)
  - DGG\_TRACE\_ERRORS, [229](#)
  - DGG\_add2stepToList, [235](#)
  - DGG\_addMirToList, [235](#)
  - DGG\_build2step, [235](#)
  - DGG\_buildMir, [235](#)
  - DGG\_copyConstraint, [234](#)
  - DGG\_cutLHS, [235](#)
  - DGG\_cutsOffPoint, [235](#)
  - DGG\_freeConstraint, [234](#)
  - DGG\_freeData, [234](#)
  - DGG\_generateCutsFromBase, [235](#)
  - DGG\_generateFormulationCuts, [235](#)
  - DGG\_generateFormulationCutsFromBase, [235](#)
  - DGG\_generateTabRowCuts, [235](#)
  - DGG\_getData, [234](#)
  - DGG\_getFormulaConstraint, [235](#)
  - DGG\_getSlackExpression, [235](#)
  - DGG\_getTableauConstraint, [235](#)
  - DGG\_is2stepValid, [235](#)
  - DGG\_is\_a\_multiple\_of\_b, [234](#)
  - DGG\_is\_even, [234](#)
  - DGG\_isBaseTrivial, [235](#)
  - DGG\_isBasic, [228](#)
  - DGG\_isConstraintBoundedAbove, [228](#)
  - DGG\_isConstraintBoundedBelow, [228](#)
  - DGG\_isConstraintViolated, [235](#)
  - DGG\_isCutDesirable, [235](#)
  - DGG\_isEqualityConstraint, [228](#)
  - DGG\_isInteger, [228](#)
  - DGG\_isNonBasicAtLB, [228](#)
  - DGG\_isNonBasicAtUB, [228](#)
  - DGG\_isStructural, [228](#)
  - DGG\_list\_addcut, [234](#)
  - DGG\_list\_delcut, [234](#)
  - DGG\_list\_free, [234](#)
  - DGG\_list\_init, [234](#)
  - DGG\_newConstraint, [234](#)
  - DGG\_nicefyConstraint, [235](#)
  - DGG\_scaleConstraint, [234](#)
  - DGG\_setEqualityConstraint, [229](#)
  - DGG\_setIsBasic, [228](#)
  - DGG\_setIsConstraintBoundedAbove, [229](#)
  - DGG\_setIsConstraintBoundedBelow, [229](#)
  - DGG\_setIsInteger, [229](#)
  - DGG\_setIsNonBasicAtLB, [229](#)
  - DGG\_setIsNonBasicAtUB, [229](#)
  - DGG\_setIsStructural, [229](#)
  - DGG\_substituteSlacks, [234](#)
  - DGG\_transformConstraint, [234](#)
  - DGG\_unTransformConstraint, [234](#)
  - frac\_part, [234](#)
  - KREM, [234](#)



- LMIN, [234](#)
  - QINT, [234](#)
  - UB\_MAX, [232](#)
  - V2I, [234](#)
- CglTwomirUnitTest
  - CglTwomir, [172](#)
  - CglTwomir.hpp, [235](#)
- CglUniqueRowCuts, [172](#)
  - ~CglUniqueRowCuts, [173](#)
  - addCuts, [173](#)
  - CglUniqueRowCuts, [173](#)
  - CglUniqueRowCuts, [173](#)
  - cut, [173](#)
  - eraseRowCut, [173](#)
  - insert, [173](#)
  - insertIfNotDuplicate, [173](#)
  - numberCuts, [173](#)
  - operator=, [173](#)
  - rowCutPtr, [173](#)
  - sizeRowCuts, [173](#)
- CglZeroHalf, [173](#)
  - ~CglZeroHalf, [175](#)
  - CglZeroHalf, [174](#)
  - CglZeroHalfUnitTest, [175](#)
  - CglZeroHalf, [174](#)
  - clone, [175](#)
  - generateCpp, [175](#)
  - generateCuts, [175](#)
  - getFlags, [175](#)
  - operator=, [175](#)
  - refreshSolver, [175](#)
  - setFlags, [175](#)
- CglZeroHalf.hpp
  - cglShortestPath, [237](#)
  - CglZeroHalfUnitTest, [237](#)
- CglZeroHalfUnitTest
  - CglZeroHalf, [175](#)
  - CglZeroHalf.hpp, [237](#)
- changeBasis
  - LAP::CglLandPSimplex, [81](#)
- cind
  - cut, [177](#)
- cl\_del\_indices
  - CglClique, [30](#)
- cl\_del\_length
  - CglClique, [30](#)
- cl\_indices
  - CglClique, [29](#)
- cl\_length
  - CglClique, [29](#)
- cl\_perm\_indices
  - CglClique, [29](#)
- cl\_perm\_length
  - CglClique, [29](#)
- cleanCut
  - LAP::Validator, [205](#)
- cleanCut2
  - LAP::Validator, [205](#)
- CleaningProcedure
  - CglGMIParam, [55](#)
- cliqueEntry, [176](#)
  - fixes, [176](#)
- cliquelt
  - CglPreProcess, [107](#)
- clone
  - CglAllDifferent, [20](#)
  - CglClique, [26](#)
  - CglCutGenerator, [33](#)
  - CglDuplicateRow, [39](#)
  - CglFakeClique, [42](#)
  - CglFlowCover, [45](#)
  - CglGMI, [50](#)
  - CglGMIParam, [61](#)
  - CglGomory, [67](#)
  - CglImplication, [70](#)
  - CglKnapsackCover, [72](#)
  - CglLandP, [76](#)
  - CglLiftAndProject, [87](#)
  - CglMixedIntegerRounding, [90](#)
  - CglMixedIntegerRounding2, [93](#)
  - CglOddHole, [100](#)
  - CglParam, [103](#)
  - CglProbing, [117](#)
  - CglRedSplit, [123](#)
  - CglRedSplit2, [126](#)
  - CglRedSplit2Param, [139](#)
  - CglRedSplitParam, [148](#)
  - CglResidualCapacity, [152](#)
  - CglSimpleRounding, [154](#)
  - CglStored, [158](#)
  - CglTreeInfo, [161](#)
  - CglTreeProbingInfo, [164](#)
  - CglTwomir, [171](#)
  - CglZeroHalf, [175](#)
- cnum
  - cut\_list, [178](#)
  - cycle\_list, [181](#)
  - pool\_cut\_list, [198](#)
- cnzcnt
  - cut, [177](#)
- code
  - pool\_cut, [197](#)
- coeff
  - DGG\_constraint\_t, [182](#)
- col\_to\_delete
  - parity\_ilp, [196](#)
- columnScalingBoundLAP\_
  - CglRedSplit2Param, [142](#)

- ColumnScalingStrategy
  - CglRedSplit2Param, 133
- columnScalingStrategyLAP\_
  - CglRedSplit2Param, 141
- ColumnSelectionStrategy
  - CglRedSplit2Param, 132
- columnSelectionStrategy\_
  - CglRedSplit2Param, 141
- columnSelectionStrategyLAP\_
  - CglRedSplit2Param, 141
- compute\_is\_integer
  - CglRedSplit, 121
- compute\_is\_lub
  - CglRedSplit, 121
- computeCglpObjective
  - LAP::CglLandPSimplex, 82, 85
- computeCglpRedCost
  - LAP::CglLandPSimplex, 84
- computeIsInteger
  - CglGMI, 50
- computeRedCostConstantsInRow
  - LAP::CglLandPSimplex, 85
- computeWeights
  - LAP::CglLandPSimplex, 84
- config\_cgl\_default.h
  - CGL\_VERSION, 237
- config\_default.h
  - COIN\_HAS\_OSI, 238
  - COIN\_HAS\_OSICLP, 238
  - COIN\_HAS\_VOL, 238
- considerRows
  - CglClique, 26
- constr
  - edge, 186
- constr\_list
  - cut, 177
  - pool\_cut, 197
- countMistakenRc
  - CglLandP::Parameters, 194
- cparams
  - DGG\_data\_t, 184
- createCliqueList
  - CglOddHole, 99
- createCliques
  - CglProbing, 115
- createIntersectionCut
  - LAP::CglLandPSimplex, 83
- createMIG
  - LAP::CglLandPSimplex, 83
- createRowList
  - CglOddHole, 99
- crhs
  - cut, 177
  - select\_cut, 199
- csense
  - cut, 177
- ctype
  - DGG\_list\_t, 184
- cut, 176
  - CglUniqueRowCuts, 173
  - cind, 177
  - cnzcnt, 177
  - constr\_list, 177
  - crhs, 177
  - csense, 177
  - cval, 177
  - in\_constr\_list, 177
  - n\_of\_constr, 177
  - violation, 177
- CutStat
  - LAP, 14
- cut\_list, 177
  - cnum, 178
  - list, 178
- cutGenerator
  - CglPreProcess, 110
- cutGenerators
  - CglPreProcess, 109
- cutParams, 178
  - a\_max, 178
  - max\_elements, 179
  - q\_max, 178
  - q\_min, 178
  - t\_max, 178
  - t\_min, 178
- Cuts
  - LAP::Cuts, 179
- cuts
  - CglPreProcess, 109
- cuts\_
  - CglStored, 159
- cutsPointer
  - CglPreProcess, 109
- cval
  - cut, 177
- cycle, 180
  - edge\_list, 181
  - length, 181
  - weight, 180
- cycle\_list, 181
  - cnum, 181
  - list, 181
- DUMMY\_END
  - LAP, 14
- DURING\_SEP
  - LAP, 13
- DGG\_2STEP\_CUT

- CglTwomir.hpp, [231](#)
- DGG\_ALPHA\_ALL
  - CglTwomir.hpp, [231](#)
- DGG\_ALPHA\_MIN\_SUM
  - CglTwomir.hpp, [231](#)
- DGG\_ALPHA\_RANDOM\_01
  - CglTwomir.hpp, [231](#)
- DGG\_BOUND\_THRESH
  - CglTwomir.hpp, [232](#)
- DGG\_CHECKRVAL
  - CglTwomir.hpp, [233](#)
- DGG\_CHECKRVAL1
  - CglTwomir.hpp, [233](#)
- DGG\_CPX
  - CglTwomir.hpp, [231](#)
- DGG\_DEBUG\_DGG
  - CglTwomir.hpp, [229](#)
- DGG\_DEBUG\_SOLVER
  - CglTwomir.hpp, [231](#)
- DGG\_DEFAULT\_METHOD
  - CglTwomir.hpp, [229](#)
- DGG\_DEFAULT\_NICEFY
  - CglTwomir.hpp, [230](#)
- DGG\_DEFAULT\_TAUMAX
  - CglTwomir.hpp, [230](#)
- DGG\_DEFAULT\_TAUMIN
  - CglTwomir.hpp, [230](#)
- DGG\_DEFAULT\_TMAX
  - CglTwomir.hpp, [229](#)
- DGG\_DEFAULT\_TMIN
  - CglTwomir.hpp, [229](#)
- DGG\_DISPLAY
  - CglTwomir.hpp, [229](#)
- DGG\_EQUALITY\_THRESH
  - CglTwomir.hpp, [232](#)
- DGG\_GOMORY\_THRESH
  - CglTwomir.hpp, [232](#)
- DGG\_IF\_EXIT
  - CglTwomir.hpp, [233](#)
- DGG\_MAX
  - CglTwomir.hpp, [233](#)
- DGG\_MAX\_L2NORM
  - CglTwomir.hpp, [232](#)
- DGG\_MIN
  - CglTwomir.hpp, [233](#)
- DGG\_MIN\_ALPHA
  - CglTwomir.hpp, [232](#)
- DGG\_MIN\_RHO
  - CglTwomir.hpp, [232](#)
- DGG\_MIN\_STEEPNESS
  - CglTwomir.hpp, [232](#)
- DGG\_NICEFY\_MIN\_FIX
  - CglTwomir.hpp, [233](#)
- DGG\_NORM\_CRITERIA
  - CglTwomir.hpp, [232](#)
- DGG\_NULL\_SLACK
  - CglTwomir.hpp, [233](#)
- DGG\_OSI
  - CglTwomir.hpp, [231](#)
- DGG\_QSO
  - CglTwomir.hpp, [231](#)
- DGG\_RHS\_THRESH
  - CglTwomir.hpp, [232](#)
- DGG\_SHIFT\_THRESH
  - CglTwomir.hpp, [232](#)
- DGG\_SOLVER
  - CglTwomir.hpp, [231](#)
- DGG\_TEST
  - CglTwomir.hpp, [233](#)
- DGG\_TEST2
  - CglTwomir.hpp, [233](#)
- DGG\_TEST3
  - CglTwomir.hpp, [233](#)
- DGG\_THROW
  - CglTwomir.hpp, [233](#)
- DGG\_TMIR\_CUT
  - CglTwomir.hpp, [231](#)
- DGG\_TRACE\_ERRORS
  - CglTwomir.hpp, [229](#)
- DGG\_add2stepToList
  - CglTwomir.hpp, [235](#)
- DGG\_addMirToList
  - CglTwomir.hpp, [235](#)
- DGG\_build2step
  - CglTwomir.hpp, [235](#)
- DGG\_buildMir
  - CglTwomir.hpp, [235](#)
- DGG\_constraint\_t, [181](#)
  - coeff, [182](#)
  - index, [182](#)
  - max\_nz, [182](#)
  - nz, [182](#)
  - rhs, [182](#)
  - sense, [182](#)
- DGG\_copyConstraint
  - CglTwomir.hpp, [234](#)
- DGG\_cutLHS
  - CglTwomir.hpp, [235](#)
- DGG\_cutsOffPoint
  - CglTwomir.hpp, [235](#)
- DGG\_data\_t, [182](#)
  - cparams, [184](#)
  - gomory\_threshold, [183](#)
  - info, [183](#)
  - lb, [183](#)
  - nbasic\_col, [183](#)
  - nbasic\_row, [183](#)
  - ncol, [183](#)

- ninteger, [183](#)
- nrow, [183](#)
- opt\_x, [184](#)
- rc, [183](#)
- ub, [183](#)
- x, [183](#)
- DGG\_freeConstraint
  - CglTwomir.hpp, [234](#)
- DGG\_freeData
  - CglTwomir.hpp, [234](#)
- DGG\_generateCutsFromBase
  - CglTwomir.hpp, [235](#)
- DGG\_generateFormulationCuts
  - CglTwomir.hpp, [235](#)
- DGG\_generateFormulationCutsFromBase
  - CglTwomir.hpp, [235](#)
- DGG\_generateTabRowCuts
  - CglTwomir.hpp, [235](#)
- DGG\_getData
  - CglTwomir.hpp, [234](#)
- DGG\_getFormulaConstraint
  - CglTwomir.hpp, [235](#)
- DGG\_getSlackExpression
  - CglTwomir.hpp, [235](#)
- DGG\_getTableauConstraint
  - CglTwomir.hpp, [235](#)
- DGG\_is2stepValid
  - CglTwomir.hpp, [235](#)
- DGG\_is\_a\_multiple\_of\_b
  - CglTwomir.hpp, [234](#)
- DGG\_is\_even
  - CglTwomir.hpp, [234](#)
- DGG\_isBaseTrivial
  - CglTwomir.hpp, [235](#)
- DGG\_isBasic
  - CglTwomir.hpp, [228](#)
- DGG\_isConstraintBoundedAbove
  - CglTwomir.hpp, [228](#)
- DGG\_isConstraintBoundedBelow
  - CglTwomir.hpp, [228](#)
- DGG\_isConstraintViolated
  - CglTwomir.hpp, [235](#)
- DGG\_isCutDesirable
  - CglTwomir.hpp, [235](#)
- DGG\_isEqualityConstraint
  - CglTwomir.hpp, [228](#)
- DGG\_isInteger
  - CglTwomir.hpp, [228](#)
- DGG\_isNonBasicAtLB
  - CglTwomir.hpp, [228](#)
- DGG\_isNonBasicAtUB
  - CglTwomir.hpp, [228](#)
- DGG\_isStructural
  - CglTwomir.hpp, [228](#)
- DGG\_list\_addcut
  - CglTwomir.hpp, [234](#)
- DGG\_list\_delcut
  - CglTwomir.hpp, [234](#)
- DGG\_list\_free
  - CglTwomir.hpp, [234](#)
- DGG\_list\_init
  - CglTwomir.hpp, [234](#)
- DGG\_list\_t, [184](#)
  - alpha, [184](#)
  - c, [184](#)
  - ctype, [184](#)
  - n, [184](#)
- DGG\_newConstraint
  - CglTwomir.hpp, [234](#)
- DGG\_nicifyConstraint
  - CglTwomir.hpp, [235](#)
- DGG\_scaleConstraint
  - CglTwomir.hpp, [234](#)
- DGG\_setEqualityConstraint
  - CglTwomir.hpp, [229](#)
- DGG\_setIsBasic
  - CglTwomir.hpp, [228](#)
- DGG\_setIsConstraintBoundedAbove
  - CglTwomir.hpp, [229](#)
- DGG\_setIsConstraintBoundedBelow
  - CglTwomir.hpp, [229](#)
- DGG\_setIsInteger
  - CglTwomir.hpp, [229](#)
- DGG\_setIsNonBasicAtLB
  - CglTwomir.hpp, [229](#)
- DGG\_setIsNonBasicAtUB
  - CglTwomir.hpp, [229](#)
- DGG\_setIsStructural
  - CglTwomir.hpp, [229](#)
- DGG\_substituteSlacks
  - CglTwomir.hpp, [234](#)
- DGG\_transformConstraint
  - CglTwomir.hpp, [234](#)
- DGG\_unTransformConstraint
  - CglTwomir.hpp, [234](#)
- degeneratePivotLimit
  - CglLandP::Parameters, [193](#)
- deleteCliques
  - CglProbing, [115](#)
- deleteSnapshot
  - CglProbing, [115](#)
- DenseCut
  - LAP::Validator, [204](#)
- disaggregationAction, [185](#)
  - affected, [185](#)
- dist
  - short\_path\_node, [200](#)
- distanceBack

- cgl\_node, 18
- do\_row\_clique
  - CgIClique, 28
- do\_star\_clique
  - CgIClique, 28
- DummyEnd
  - LAP::Validator, 204
- duplicate
  - CgIDuplicateRow, 37
- duplicate\_
  - CgIDuplicateRow, 39
- Dynamic
  - CgILandP, 76
- END\_ROUND
  - LAP, 13
- ENFORCE\_SCALING
  - CgIGMIParam, 62
- EPS
  - CgIParam, 103
- EPS\_COEFF
  - CgIParam, 103
- EPS\_COEFF\_LUB
  - CgIRedSplitParam, 149
- EPS\_ELIM
  - CgIGMIParam, 61
  - CgIRedSplit2Param, 140
  - CgIRedSplitParam, 148
- EPS\_RELAX\_ABS
  - CgIGMIParam, 61
  - CgIRedSplit2Param, 140
  - CgIRedSplitParam, 149
- EPS\_RELAX\_REL
  - CgIGMIParam, 61
  - CgIRedSplit2Param, 140
  - CgIRedSplitParam, 149
- edge, 185
  - constr, 186
  - endpoint1, 185
  - endpoint2, 185
  - parity, 186
  - weak, 186
  - weight, 186
- edge\_list
  - cycle, 181
- eliminate\_slacks
  - LAP::CgILandPSimplex, 84
- EmptyCut
  - LAP::Validator, 204
- endpoint1
  - edge, 185
- endpoint2
  - edge, 185
- eraseRowCut
  - CglUniqueRowCuts, 173
- even\_adj\_list
  - separation\_graph, 200
- extraCuts
  - LAP::CgILandPSimplex, 81
- extraCutsLimit
  - CgILandP::Parameters, 193
- ExtraCutsMode
  - CgILandP, 74
- FailedSigmaIncreased
  - LAP, 13
- failedPivotLimit
  - CgILandP::Parameters, 193
- failureDynamism
  - CgIGMI, 48
- failureFractionality
  - CgIGMI, 48
- failureScale
  - CgIGMI, 49
- failureSupport
  - CgIGMI, 49
- failureViolation
  - CgIGMI, 49
- failureString
  - LAP::Validator, 205
- fakeSolver\_
  - CglFakeClique, 42
- fastFindBestPivotColumn
  - LAP::CgILandPSimplex, 82
- fastFindCutImprovingPivotRow
  - LAP::CgILandPSimplex, 82
- fgraph
  - CgIClique, 28
- findBestPivot
  - LAP::CgILandPSimplex, 82
- findBestPivotColumn
  - LAP::CgILandPSimplex, 85
- findCutImprovingPivotRow
  - LAP::CgILandPSimplex, 85
- FinishedOptimal
  - LAP, 13
- firstArc
  - cgl\_node, 18
- fixColumns
  - CglTreeProbingInfo, 165
- fixEntries
  - CglTreeProbingInfo, 165
- fixEntry\_
  - CglTreeProbingInfo, 166
- Fixed
  - CgILandP, 76
- fixes
  - CglTreeInfo, 161

- CglTreeProbingInfo, [164](#)
  - cliqueEntry, [176](#)
- fixingEntry\_
  - CglTreeProbingInfo, [166](#)
- flowPreprocess
  - CglFlowCover, [44](#)
- formulation\_rows
  - CglTreeInfo, [161](#)
- FoundBestImprovingCol
  - LAP, [13](#)
- FoundImprovingRow
  - LAP, [13](#)
- frac\_graph
  - CglClique, [27](#)
- frac\_part
  - CglTwomir.hpp, [234](#)
- Fractional
  - CglLandP, [75](#)
- Fractional\_rc
  - CglLandP, [75](#)
- free\_ilp
  - Cgl012Cut, [16](#)
- free\_log\_var
  - Cgl012Cut, [16](#)
- free\_parity\_ilp
  - Cgl012Cut, [16](#)
- freeSi
  - LAP::CglLandPSimplex, [81](#)
- Full
  - CglLandP, [75](#)
- gcd
  - parity\_ilp, [196](#)
- genThisBasisMigs
  - LAP::CglLandPSimplex, [81](#)
- generateCuts
  - CglAllDifferent, [20](#)
  - CglClique, [26](#)
  - CglCutGenerator, [32](#)
  - CglDuplicateRow, [37](#)
  - CglFakeClique, [42](#)
  - CglFlowCover, [44](#)
  - CglGMI, [49](#)
  - CglGomory, [65](#)
  - CglImplication, [70](#)
  - CglKnapsackCover, [72](#)
  - CglLandP, [76](#)
  - CglLiftAndProject, [86](#)
  - CglMixedIntegerRounding, [90](#)
  - CglMixedIntegerRounding2, [93](#)
  - CglOddHole, [99](#)
  - CglProbing, [114](#)
  - CglRedSplit, [121](#)
  - CglRedSplit2, [125](#)
  - CglResidualCapacity, [152](#)
  - CglSimpleRounding, [154](#)
  - CglStored, [157](#)
  - CglTreeProbingInfo, [165](#)
  - CglTwomir, [169](#)
  - CglZeroHalf, [175](#)
- generateCutsAndModify
  - CglProbing, [115](#)
- generateExtraCut
  - LAP::CglLandPSimplex, [80](#)
- generateExtraCuts
  - CglLandP::Parameters, [194](#)
  - LAP::CglLandPSimplex, [80](#)
- generateMig
  - LAP::CglLandPSimplex, [80](#)
- generateMultipliers
  - CglRedSplit2, [126](#)
- get\_M1\_M2\_M3
  - LAP::CglLandPSimplex, [84](#)
- getAWAY
  - CglGMIParam, [57](#)
- getAggressiveness
  - CglCutGenerator, [33](#)
- getAmax
  - CglTwomir, [170](#)
- getApplicationData
  - CglPreProcess, [110](#)
- getAway
  - CglGMIParam, [57](#)
  - CglGomory, [66](#)
  - CglRedSplit, [122](#)
  - CglRedSplit2Param, [134](#)
  - CglRedSplitParam, [146](#)
  - CglTwomir, [170](#)
- getAwayAtRoot
  - CglGomory, [66](#)

- CglTwomir, [171](#)
- getBasics
  - LAP::CglLandPSimplex, [81](#)
- getBasis
  - LAP::CglLandPSimplex, [81](#)
- getBeta
  - CglLiftAndProject, [87](#)
- getCHECK\_DUPLICATES
  - CglGMIParam, [59](#)
- getCLEAN\_PROC
  - CglGMIParam, [60](#)
- getCRITERION\_
  - CglMixedIntegerRounding, [91](#)
  - CglMixedIntegerRounding2, [94](#)
- getCheckDuplicates
  - CglGMIParam, [59](#)
- getCleaningProcedure
  - CglGMIParam, [60](#)
- getColsolToCut
  - LAP::CglLandPSimplex, [83](#)
- getColumnScalingBoundLAP
  - CglRedSplit2Param, [138](#)
- getColumnScalingStrategyLAP
  - CglRedSplit2Param, [138](#)
- getColumnSelectionStrategy
  - CglRedSplit2Param, [137](#)
- getColumnSelectionStrategyLAP
  - CglRedSplit2Param, [138](#)
- getConditionNumberMultiplier
  - CglGomory, [67](#)
- getCutoff
  - CglPreProcess, [107](#)
- getDoPreproc
  - CglMixedIntegerRounding, [91](#)
  - CglMixedIntegerRounding2, [94](#)
  - CglResidualCapacity, [152](#)
- getENFORCE\_SCALING
  - CglGMIParam, [60](#)
- getEPS
  - CglParam, [102](#)
  - CglRedSplit, [122](#)
- getEPS\_COEFF
  - CglParam, [102](#)
  - CglRedSplit, [122](#)
- getEPS\_COEFF\_LUB
  - CglRedSplit, [122](#)
  - CglRedSplitParam, [147](#)
- getEPS\_ELIM
  - CglGMIParam, [57](#)
  - CglRedSplit2Param, [134](#)
  - CglRedSplitParam, [146](#)
- getEPS\_RELAX
  - CglRedSplit, [122](#)
- getEPS\_RELAX\_ABS
  - CglGMIParam, [57](#)
  - CglRedSplit2Param, [135](#)
  - CglRedSplitParam, [146](#)
- getEPS\_RELAX\_REL
  - CglGMIParam, [58](#)
  - CglRedSplit2Param, [135](#)
  - CglRedSplitParam, [146](#)
- getEnforcescaling
  - CglGMIParam, [61](#)
- getEps
  - CglGMIParam, [56](#)
- getEpsCoeff
  - CglGMIParam, [56](#)
- getEpsElim
  - CglGMIParam, [57](#)
- getEpsRelaxAbs
  - CglGMIParam, [57](#)
- getEpsRelaxRel
  - CglGMIParam, [58](#)
- getEpsilon
  - CglResidualCapacity, [152](#)
- getFlags
  - CglZeroHalf, [175](#)
- getINFINIT
  - CglParam, [102](#)
- getIfFormulation
  - CglTwomir, [170](#)
- getIfMir
  - CglTwomir, [170](#)
- getIfTableau
  - CglTwomir, [170](#)
- getIfTwomir
  - CglTwomir, [170](#)
- getInfinity
  - CglGMIParam, [56](#)
- getIntegralScaleCont
  - CglGMIParam, [60](#)
- getLUB
  - CglRedSplit, [122](#)
  - CglRedSplitParam, [146](#)
- getLargestFactorMultiplier
  - CglGomory, [67](#)
- getLimit
  - CglGomory, [66](#)
  - CglRedSplit, [122](#)
- getLimitAtRoot
  - CglGomory, [66](#)
- getLoBound
  - LAP::CglLandPSimplex, [83](#)
- getLogLevel
  - CglAllDifferent, [21](#)
  - CglProbing, [116](#)
- getMAX\_SUPP\_ABS
  - CglRedSplit2Param, [135](#)

getMAX\_SUPP\_REL  
    CglRedSplit2Param, 135  
getMAX\_SUPPORT  
    CglParam, 103  
getMAX\_SUPPORT\_ABS  
    CglGMIParam, 56  
getMAX\_SUPPORT\_REL  
    CglGMIParam, 59  
getMAXAGGR\_  
    CglMixedIntegerRounding, 90  
    CglMixedIntegerRounding2, 93  
getMAXDYN  
    CglGMIParam, 58  
    CglRedSplit2Param, 135  
    CglRedSplitParam, 147  
getMAXDYN\_LUB  
    CglRedSplitParam, 147  
getMINVIOL  
    CglGMIParam, 58  
    CglRedSplit2Param, 135  
    CglRedSplitParam, 147  
getMULTIPLY\_  
    CglMixedIntegerRounding, 91  
    CglMixedIntegerRounding2, 94  
getMaxDyn  
    CglGMIParam, 58  
getMaxElements  
    CglProbing, 116  
    CglTwomir, 170  
getMaxElementsRoot  
    CglProbing, 117  
    CglTwomir, 170  
getMaxFillIn  
    LAP::Validator, 205  
getMaxInKnapsack  
    CglKnapsackCover, 72  
getMaxLook  
    CglAllDifferent, 21  
    CglProbing, 116  
getMaxLookRoot  
    CglProbing, 117  
getMaxNonzeroesTab  
    CglRedSplit2Param, 139  
getMaxNumComputedCuts  
    CglRedSplit2Param, 139  
getMaxNumCuts  
    CglFlowCover, 44  
    CglRedSplit2Param, 139  
getMaxPass  
    CglProbing, 116  
getMaxPassRoot  
    CglProbing, 116  
getMaxProbe  
    CglProbing, 116  
getMaxProbeRoot  
    CglProbing, 116  
getMaxRatio  
    LAP::Validator, 205  
getMaxSumMultipliers  
    CglRedSplit2Param, 136  
getMaxSupport  
    CglGMIParam, 56  
getMaxSupportAbs  
    CglGMIParam, 56  
getMaxSupportRel  
    CglGMIParam, 59  
getMaxTab  
    CglRedSplit, 123  
    CglRedSplitParam, 148  
getMaximumEntries  
    CglOddHole, 100  
getMinNormReduction  
    CglRedSplit2Param, 136  
getMinReduc  
    CglRedSplit, 123  
    CglRedSplitParam, 148  
getMinViol  
    CglGMIParam, 58  
getMinViolation  
    CglClique, 27  
    LAP::Validator, 205  
getMinimumViolation  
    CglOddHole, 100  
getMinimumViolationPer  
    CglOddHole, 100  
getMode  
    CglProbing, 115  
getNonBasics  
    LAP::CglLandPSimplex, 81  
getNormIsZero  
    CglRedSplit, 123  
    CglRedSplit2Param, 136  
    CglRedSplitParam, 148  
getNormalization  
    CglRedSplit2Param, 136  
getNumCols  
    LAP::CglLandPSimplex, 81  
getNumFlowCuts  
    CglFlowCover, 44  
getNumRows  
    LAP::CglLandPSimplex, 81  
getNumRowsReduction  
    CglRedSplit2Param, 136  
getNumRowsReductionLAP  
    CglRedSplit2Param, 137  
getNumberGeneratedCuts  
    CglGMI, 50  
getNumberRejectedCuts



- CglGMI, 50
- getParam
  - CglGMI, 50
  - CglRedSplit, 121
  - CglRedSplit2, 126
- getQmax
  - CglTwomir, 170
- getQmin
  - CglTwomir, 170
- getRequiredViolation
  - CglStored, 157
- getRowSelectionStrategy
  - CglRedSplit2Param, 137
- getRowSelectionStrategyLAP
  - CglRedSplit2Param, 138
- getSkipGomory
  - CglRedSplit2Param, 139
- getStatus
  - LAP::CglLandPSimplex, 84
- getTimeLimit
  - CglRedSplit2Param, 138
- getTmax
  - CglTwomir, 170
- getTmin
  - CglTwomir, 170
- getTolerance
  - CglResidualCapacity, 152
- getTrackRejection
  - CglGMI, 50
- getUSE\_CG2
  - CglRedSplitParam, 147
- getUSE\_INTSLACKS
  - CglGMIParam, 59
  - CglRedSplit2Param, 136
  - CglRedSplitParam, 147
- getUpBound
  - LAP::CglLandPSimplex, 83
- getUseIntSlacks
  - CglGMIParam, 59
- getUsingObjective
  - CglProbing, 117
- getVal
  - CglFlowVUB, 46
  - CglMixIntRoundVUB, 95
  - CglMixIntRoundVUB2, 97
- getVar
  - CglFlowVUB, 46
  - CglMixIntRoundVUB, 95
  - CglMixIntRoundVUB2, 97
- gomory\_threshold
  - DGG\_data\_t, 183
- gomoryType
  - CglGomory, 66
- gutsOfDestructor
  - CglPreProcess, 110
- HitLimit
  - LAP, 13
- INFINIT
  - CglParam, 103
- ilp, 186
  - mc, 186
  - mnz, 187
  - mr, 186
  - mrhs, 187
  - msense, 187
  - mtbeg, 187
  - mtcnt, 187
  - mtind, 187
  - mtval, 187
  - vlb, 187
  - vub, 187
  - xstar, 187
- ilp\_load
  - Cgl012Cut, 16
- in\_constr\_list
  - cut, 177
- inTree
  - CglTreeInfo, 162
- incNumFlowCuts
  - CglFlowCover, 45
- ind
  - separation\_graph, 200
- index
  - cgl\_node, 18
  - CglHashLink, 68
  - DGG\_constraint\_t, 182
- Infinity
  - CglLandP, 75
- info
  - DGG\_data\_t, 183
- info\_weak, 187
  - nweak, 188
  - type, 188
  - var, 188
- initialReducedCosts
  - CglLandP, 74
- initialize\_log\_var
  - Cgl012Cut, 16
- initializeFixing
  - CglTreeInfo, 161
  - CglTreeProbingInfo, 165
- insert
  - CglUniqueRowCuts, 173
  - LAP::Cuts, 180
- insertAll
  - LAP::Cuts, 180
- insertAllExtr

- LAP::CglLandPSimplex, 81
- insertIfNotDuplicate
  - CglUniqueRowCuts, 173
- int\_val
  - LAP, 14
- integerVariable
  - CglTreeProbingInfo, 165
- integerVariable\_
  - CglTreeProbingInfo, 166
- intersectionCutCoef
  - LAP, 14
- isGtConst
  - LAP::CglLandPSimplex, 83
- isInteger
  - LAP::CglLandPSimplex, 84
- isIntegerValue
  - CglGMI, 50
- isZero
  - CglGMI, 49
- it\_found
  - pool\_cut, 197
- justOriginalRows\_
  - CglClique, 27
- KREM
  - CglTwomir.hpp, 234
- L1
  - CglLandP, 75
- L2
  - CglLandP, 75
- LAP
  - BEGIN\_ROUND, 13
  - CUT\_FAILED, 13
  - CUT\_GAP, 13
  - CUT\_REJECTED, 13
  - CutStat, 14
  - DUMMY\_END, 14
  - DURING\_SEP, 13
  - END\_ROUND, 13
  - FailedSigmaIncreased, 13
  - FinishedOptimal, 13
  - FoundBestImprovingCol, 13
  - FoundImprovingRow, 13
  - HitLimit, 13
  - LAP\_CUT\_FAILED\_DO\_MIG, 13
  - LAP\_MESSAGES\_DUMMY\_END, 13
  - LogHead, 13
  - NumberNegRc, 13
  - NumberPosRc, 13
  - NumberZeroRc, 13
  - PivotFailedSigmaIncreased, 13
  - PivotFailedSigmaUnchanged, 13
  - PivotLog, 13
  - RoundStats, 14
  - Separating, 13
  - WarnBadRhsComputation, 13
  - WarnBadRowComputation, 13
  - WarnBadSigmaComputation, 13
  - WarnFailedBestImprovingCol, 13
  - WarnFailedPivotIf, 13
  - WarnFailedPivotTol, 13
  - WarnGiveUpRow, 13
  - WeightsStats, 13
- LAP::Validator
  - BigDynamic, 204
  - DenseCut, 204
  - DummyEnd, 204
  - EmptyCut, 204
  - NoneAccepted, 204
  - SmallCoefficient, 204
  - SmallViolation, 204
- LAP\_CUT\_FAILED\_DO\_MIG
  - LAP, 13
- LAP\_MESSAGES\_DUMMY\_END
  - LAP, 13
- LAP, 12
  - int\_val, 14
  - intersectionCutCoef, 14
  - LAP\_messages, 13
  - LapMessagesTypes, 13
  - modularizeRow, 14
  - modularizedCoef, 14
  - normCoef, 14
  - scale, 14
- LAP::CglLandPSimplex, 77
  - ~CglLandPSimplex, 80
  - adjustTableauRow, 83
  - cacheUpdate, 80
  - CglLandPSimplex, 80
  - CglLandP, 77
  - changeBasis, 81
  - computeCglpObjective, 82, 85
  - computeCglpRedCost, 84
  - computeRedCostConstantsInRow, 85
  - computeWeights, 84
  - createIntersectionCut, 83
  - createMIG, 83
  - eliminate\_slacks, 84
  - extraCuts, 81
  - fastFindBestPivotColumn, 82
  - fastFindCutImprovingPivotRow, 82
  - findBestPivot, 82
  - findBestPivotColumn, 85
  - findCutImprovingPivotRow, 85
  - freeSi, 81
  - genThisBasisMigs, 81
  - generateExtraCut, 80

- generateExtraCuts, 80
- generateMig, 80
- get\_M1\_M2\_M3, 84
- getBasics, 81
- getBasis, 81
- getColsolToCut, 83
- getLoBound, 83
- getNonBasics, 81
- getNumCols, 81
- getNumRows, 81
- getStatus, 84
- getUpBound, 83
- insertAllExtr, 81
- isGtConst, 83
- isInteger, 84
- loadBasis, 81
- newRowCoefficient, 82
- normalizationFactor, 83
- normedCoef, 84
- optimize, 80
- outPivInfo, 81
- plotCGLPobj, 85
- printCglpBasis, 84
- printCutLateX, 84
- printEverything, 84
- printRowLateX, 84
- printTableau, 84
- printTableauLateX, 84
- pullTableauRow, 83
- rescanReducedCosts, 82
- resetOriginalTableauRow, 83
- resetSolver, 80
- scaleCut, 83
- setColsolToCut, 83
- setLogLevel, 81
- setSi, 81
- strengthenedIntersectionCutCoef, 82
- LAP::Cuts, 179
  - ~Cuts, 179
  - Cuts, 179
  - insert, 180
  - insertAll, 180
  - numberCuts, 180
  - resize, 180
  - rowCut, 180
- LAP::LandPMessages, 188
  - LandPMessages, 189
- LAP::LapMessages, 189
  - ~LapMessages, 189
  - LapMessages, 189
- LAP::TabRow, 201
  - ~TabRow, 202
  - modularize, 202
  - modularized\_, 203
  - num, 203
  - operator=, 202
  - operator==, 202
  - print, 202
  - rhs, 203
  - si\_, 203
  - TabRow, 202
- LAP::Validator, 203
  - cleanCut, 205
  - cleanCut2, 205
  - failureString, 205
  - getMaxFillIn, 205
  - getMaxRatio, 205
  - getMinViolation, 205
  - numRejected, 205
  - operator(), 205
  - RejectionsReasons, 204
  - setMaxFillIn, 205
  - setMaxRatio, 205
  - setMinViolation, 205
  - setRhsScale, 205
  - Validator, 204
- LAP\_messages
  - LAP, 13
- LHSnorm
  - CglLandP, 75
- LMIN
  - CglTwomir.hpp, 234
- LUB
  - CglRedSplitParam, 148
- LandPMessages
  - LAP::LandPMessages, 189
- LapMessages
  - LAP::LapMessages, 189
- LapMessagesTypes
  - LAP, 13
- lb
  - DGG\_data\_t, 183
- length
  - cgl\_arc, 17
  - cycle, 181
- level
  - CglTreeInfo, 161
- lhs\_norm
  - CglLandP::Parameters, 194
- list
  - cut\_list, 178
  - cycle\_list, 181
  - pool\_cut\_list, 198
- loadBasis
  - LAP::CglLandPSimplex, 81
- LogHead
  - LAP, 13
- log\_var, 190

- n\_it\_zero, [190](#)
- logLevel
  - CglDuplicateRow, [38](#)
- logLevel\_
  - CglDuplicateRow, [40](#)
- lookedAt
  - CglProbing, [117](#)
- loss\_even\_weak
  - parity\_ilp, [196](#)
- loss\_odd\_weak
  - parity\_ilp, [197](#)
- lower\_
  - CglDuplicateRow, [39](#)
- MAX\_SUPP\_REL
  - CglRedSplit2Param, [140](#)
- MAX\_SUPPORT
  - CglParam, [103](#)
- MAX\_SUPPORT\_REL
  - CglGMIParam, [62](#)
- MAXDYN
  - CglGMIParam, [61](#)
  - CglRedSplit2Param, [140](#)
  - CglRedSplitParam, [149](#)
- MAXDYN\_LUB
  - CglRedSplitParam, [149](#)
- MINVIOL
  - CglGMIParam, [61](#)
  - CglRedSplit2Param, [140](#)
  - CglRedSplitParam, [149](#)
- matrix\_
  - CglDuplicateRow, [39](#)
- matrixByRow\_
  - CglDuplicateRow, [39](#)
- max\_elements
  - cutParams, [179](#)
- max\_nz
  - DGG\_constraint\_t, [182](#)
- maxCutPerRound
  - CglLandP::Parameters, [193](#)
- maxNonzeroesTab\_
  - CglRedSplit2Param, [142](#)
- maxNumComputedCuts\_
  - CglRedSplit2Param, [142](#)
- maxNumCuts\_
  - CglRedSplit2Param, [142](#)
- maxSumMultipliers\_
  - CglRedSplit2Param, [140](#)
- maxTab\_
  - CglRedSplitParam, [150](#)
- maximumDominated
  - CglDuplicateRow, [38](#)
- maximumDominated\_
  - CglDuplicateRow, [39](#)
- maximumEntries\_
  - CglTreeProbingInfo, [166](#)
- maxLengthOfCutInTree
  - CglCutGenerator, [34](#)
  - CglGomory, [66](#)
  - CglTwomir, [171](#)
- maximumRhs
  - CglDuplicateRow, [38](#)
- maximumRhs\_
  - CglDuplicateRow, [40](#)
- mayGenerateRowCutsInTree
  - CglAllDifferent, [21](#)
  - CglCutGenerator, [34](#)
  - CglProbing, [117](#)
- mc
  - ilp, [186](#)
  - parity\_ilp, [195](#)
- messageHandler
  - CglPreProcess, [110](#)
- messages
  - CglPreProcess, [110](#)
- messagesPointer
  - CglPreProcess, [110](#)
- min\_loss\_by\_weak
  - parity\_ilp, [197](#)
- minNormReduction\_
  - CglRedSplit2Param, [140](#)
- minReduc
  - CglRedSplitParam, [150](#)
- mnz
  - ilp, [187](#)
  - parity\_ilp, [196](#)
- mode
  - CglDuplicateRow, [38](#)
- mode\_
  - CglDuplicateRow, [40](#)
- modelAtPass
  - CglPreProcess, [107](#)
- modifiedModel
  - CglPreProcess, [108](#)
- modularize
  - CglLandP::Parameters, [194](#)
  - LAP::TabRow, [202](#)
- modularizeRow
  - LAP, [14](#)
- modularized\_
  - LAP::TabRow, [203](#)
- modularizedCoef
  - LAP, [14](#)
- mostNegativeRc
  - CglLandP, [74](#)
- mr
  - ilp, [186](#)
  - parity\_ilp, [195](#)

- mrhs
  - ilp, [187](#)
  - parity\_ilp, [196](#)
- msense
  - ilp, [187](#)
- mtbeg
  - ilp, [187](#)
  - parity\_ilp, [196](#)
- mtcnt
  - ilp, [187](#)
  - parity\_ilp, [196](#)
- mtind
  - ilp, [187](#)
  - parity\_ilp, [196](#)
- mtval
  - ilp, [187](#)
- n
  - DGG\_list\_t, [184](#)
- n\_it\_violated
  - pool\_cut, [197](#)
- n\_it\_zero
  - log\_var, [190](#)
- n\_of\_constr
  - cut, [177](#)
  - pool\_cut, [197](#)
- narcs
  - auxiliary\_graph, [15](#)
  - cgl\_graph, [17](#)
- nbasic\_col
  - DGG\_data\_t, [183](#)
- nbasic\_row
  - DGG\_data\_t, [183](#)
- ncod
  - pool\_cut\_list, [198](#)
- ncol
  - DGG\_data\_t, [183](#)
- nedges
  - separation\_graph, [199](#)
- needsOptimalBasis
  - CglCutGenerator, [34](#)
  - CglGMI, [49](#)
  - CglGomory, [65](#)
  - CglLandP, [76](#)
  - CglRedSplit, [121](#)
  - CglRedSplit2, [126](#)
  - CglTwomir, [169](#)
- newLanguage
  - CglPreProcess, [110](#)
- newRowCoefficient
  - LAP::CglLandPSimplex, [82](#)
- newSolver
  - CglBK, [22](#)
- next
  - CglHashLink, [68](#)
- ninteger
  - DGG\_data\_t, [183](#)
- nnodes
  - auxiliary\_graph, [15](#)
  - cgl\_graph, [17](#)
  - separation\_graph, [199](#)
- NoBasisError
  - CglLandP::NoBasisError, [190](#)
- node\_node
  - CglClique, [28](#)
- nodes
  - auxiliary\_graph, [15](#)
  - cgl\_graph, [18](#)
  - separation\_graph, [199](#)
- none
  - CglLandP, [75](#)
- NoneAccepted
  - LAP::Validator, [204](#)
- normCoef
  - LAP, [14](#)
- normIsZero
  - CglRedSplitParam, [149](#)
- normIsZero\_
  - CglRedSplit2Param, [140](#)
- Normalization
  - CglLandP, [75](#)
- normalization
  - CglLandP::Parameters, [194](#)
- normalization\_
  - CglRedSplit2Param, [141](#)
- normalizationFactor
  - LAP::CglLandPSimplex, [83](#)
- normedCoef
  - LAP::CglLandPSimplex, [84](#)
- nrow
  - DGG\_data\_t, [183](#)
- num
  - LAP::TabRow, [203](#)
- numRejected
  - LAP::Validator, [205](#)
- numRowsReduction\_
  - CglRedSplit2Param, [141](#)
- numRowsReductionLAP\_
  - CglRedSplit2Param, [141](#)
- NumberNegRc
  - LAP, [13](#)
- NumberPosRc
  - LAP, [13](#)
- NumberZeroRc
  - LAP, [13](#)
- numberColumns\_
  - CglStored, [159](#)
- numberCutGenerators

- CglPreProcess, 109
- numberCuts
  - CglUniqueRowCuts, 173
  - LAP::Cuts, 180
- numberEntries\_
  - CglTreeProbingInfo, 166
- numberIntegers
  - CglTreeProbingInfo, 165
- numberIntegers\_
  - CglTreeProbingInfo, 166
- numberIterationsPost
  - CglPreProcess, 109
- numberIterationsPre
  - CglPreProcess, 109
- numberOriginalRows
  - CglDuplicateRow, 38
- numberPossible
  - CglOddHole, 99
- numberSOS
  - CglPreProcess, 108
- numberThisTime
  - CglProbing, 117
- numberVariables
  - CglTreeProbingInfo, 165
- numberVariables\_
  - CglTreeProbingInfo, 166
- nweak
  - info\_weak, 188
- nz
  - DGG\_constraint\_t, 182
- OLD\_COMPUTATION
  - CglLandPSimplex.hpp, 214
- odd\_adj\_list
  - separation\_graph, 200
- oneFixesInCliqueEntry
  - CglTreeInfo.hpp, 225
- operator<<
  - CglFlowCover.hpp, 209
- operator()
  - LAP::Validator, 205
- operator=
  - Cgl012Cut, 16
  - CglAllDifferent, 20
  - CglBK, 22
  - CglClique, 26
  - CglCutGenerator, 33
  - CglDuplicateRow, 39
  - CglFakeClique, 42
  - CglFlowCover, 45
  - CglFlowVUB, 46
  - CglGMI, 50
  - CglGMIParam, 61
  - CglGomory, 67
  - CglImplication, 70
  - CglKnapsackCover, 72
  - CglLandP, 76
  - CglLandP::Parameters, 192
  - CglLiftAndProject, 87
  - CglMixedIntegerRounding, 90
  - CglMixedIntegerRounding2, 93
  - CglMixIntRoundVUB, 95
  - CglMixIntRoundVUB2, 97
  - CglOddHole, 100
  - CglParam, 103
  - CglPreProcess, 110
  - CglProbing, 118
  - CglRedSplit, 123
  - CglRedSplit2, 126
  - CglRedSplit2Param, 139
  - CglRedSplitParam, 148
  - CglResidualCapacity, 152
  - CglSimpleRounding, 154
  - CglStored, 158
  - CglTreeInfo, 161
  - CglTreeProbingInfo, 164
  - CglTwomir, 171
  - CglUniqueRowCuts, 173
  - CglZeroHalf, 175
  - LAP::TabRow, 202
- operator==
  - LAP::TabRow, 202
- opt\_x
  - DGG\_data\_t, 184
- optimize
  - LAP::CglLandPSimplex, 80
- options
  - CglTreeInfo, 161
- originalColumns
  - CglPreProcess, 108
- originalModel
  - CglPreProcess, 107
- originalRows
  - CglPreProcess, 108
- originalSolver
  - CglGomory, 66
  - CglTwomir, 171
- outDuplicates
  - CglDuplicateRow, 37
- outPivInfo
  - LAP::CglLandPSimplex, 81
- packDown
  - CglTreeProbingInfo, 165
- parameter
  - CglLandP, 77
- Parameters
  - CglLandP::Parameters, 192

- parentNode
  - cgl\_node, 18
- parity
  - edge, 186
- parity\_ilp, 195
  - col\_to\_delete, 196
  - gcd, 196
  - loss\_even\_weak, 196
  - loss\_odd\_weak, 197
  - mc, 195
  - min\_loss\_by\_weak, 197
  - mnz, 196
  - mr, 195
  - mrhs, 196
  - mtbeg, 196
  - mtcnt, 196
  - mtind, 196
  - possible\_weak, 196
  - row\_to\_delete, 196
  - slack, 196
  - type\_even\_weak, 196
  - type\_odd\_weak, 196
  - xstar, 196
- pass
  - CglTreeInfo, 161
- passInMessageHandler
  - CglPreProcess, 110
- passInOriginalSolver
  - CglGomory, 65
  - CglTwomir, 171
- passInProhibited
  - CglPreProcess, 108
- passInRowTypes
  - CglPreProcess, 109
- perturb
  - CglLandP::Parameters, 194
- petol
  - CglClique, 28
- PivotFailedSigmaIncreased
  - LAP, 13
- PivotFailedSigmaUnchanged
  - LAP, 13
- PivotLog
  - LAP, 13
- pivotLimit
  - CglLandP::Parameters, 193
- pivotLimitInTree
  - CglLandP::Parameters, 193
- pivotSelection
  - CglLandP::Parameters, 195
- pivotTol
  - CglLandP::Parameters, 193
- plotCGLPobj
  - LAP::CglLandPSimplex, 85
- pool\_cut, 197
  - code, 197
  - constr\_list, 197
  - it\_found, 197
  - n\_it\_violated, 197
  - n\_of\_constr, 197
  - score, 197
- pool\_cut\_list, 198
  - cnum, 198
  - list, 198
  - ncod, 198
- pool\_index
  - select\_cut, 199
- possible\_weak
  - parity\_ilp, 196
- postProcess
  - CglPreProcess, 107
- preProcess
  - CglPreProcess, 106
- preProcessNonDefault
  - CglPreProcess, 106
- pred
  - short\_path\_node, 200
- presolve
  - CglPreProcess, 108
- print
  - CglRedSplit, 121
  - CglRedSplit2, 126
  - LAP::TabRow, 202
- printCglpBasis
  - LAP::CglLandPSimplex, 84
- printCutLateX
  - LAP::CglLandPSimplex, 84
- printEverything
  - LAP::CglLandPSimplex, 84
- printOptTab
  - CglGMI, 50
  - CglRedSplit, 121
  - CglRedSplit2, 126
- printRowLateX
  - LAP::CglLandPSimplex, 84
- printTableau
  - LAP::CglLandPSimplex, 84
- printTableauLateX
  - LAP::CglLandPSimplex, 84
- probing\_
  - CglFakeClique, 42
- probingInfo\_
  - CglStored, 159
- probnam\_
  - CglTwomir, 172
- prohibited
  - CglPreProcess, 108
- pullTableauRow

- LAP::CglLandPSimplex, [83](#)
- q\_max
  - cutParams, [178](#)
- q\_min
  - cutParams, [178](#)
- QINT
  - CglTwomir.hpp, [234](#)
- RS1
  - CglRedSplit2Param, [132](#)
- RS2
  - CglRedSplit2Param, [132](#)
- RS3
  - CglRedSplit2Param, [132](#)
- RS4
  - CglRedSplit2Param, [132](#)
- RS5
  - CglRedSplit2Param, [132](#)
- RS6
  - CglRedSplit2Param, [132](#)
- RS7
  - CglRedSplit2Param, [132](#)
- RS8
  - CglRedSplit2Param, [132](#)
- RS\_ALL
  - CglRedSplit2Param, [132](#)
- RS\_BEST
  - CglRedSplit2Param, [132](#)
- REDUCTION
  - Cgl012cut.hpp, [236](#)
- randomNumberGenerator
  - CglTreeInfo, [162](#)
- rc
  - DGG\_data\_t, [183](#)
- rcl\_candidate\_length\_threshold
  - CglClique, [29](#)
- rcl\_report\_result
  - CglClique, [29](#)
- reducedCostFix
  - CglPreProcess, [107](#)
- refreshPrep
  - CglResidualCapacity, [152](#)
- refreshSolver
  - CglAllDifferent, [20](#)
  - CglCutGenerator, [33](#)
  - CglDuplicateRow, [39](#)
  - CglGomory, [67](#)
  - CglKnapsackCover, [72](#)
  - CglMixedIntegerRounding, [90](#)
  - CglMixedIntegerRounding2, [93](#)
  - CglOddHole, [100](#)
  - CglProbing, [118](#)
  - CglTwomir, [171](#)
  - CglZeroHalf, [175](#)
- RejectionType
  - CglGMI, [48](#)
- RejectionsReasons
  - LAP::Validator, [204](#)
- relaxedRowLower
  - CglProbing, [115](#)
- relaxedRowUpper
  - CglProbing, [115](#)
- requiredViolation\_
  - CglStored, [159](#)
- rescanReducedCosts
  - LAP::CglLandPSimplex, [82](#)
- resetColumnSelectionStrategy
  - CglRedSplit2Param, [137](#)
- resetColumnSelectionStrategyLAP
  - CglRedSplit2Param, [138](#)
- resetNumRowsReduction
  - CglRedSplit2Param, [137](#)
- resetNumRowsReductionLAP
  - CglRedSplit2Param, [137](#)
- resetOriginalTableauRow
  - LAP::CglLandPSimplex, [83](#)
- resetRejectionCounters
  - CglGMI, [50](#)
- resetRowSelectionStrategy
  - CglRedSplit2Param, [137](#)
- resetRowSelectionStrategyLAP
  - CglRedSplit2Param, [138](#)
- resetSolver
  - LAP::CglLandPSimplex, [80](#)
- resize
  - LAP::Cuts, [180](#)
- rhs
  - DGG\_constraint\_t, [182](#)
  - LAP::TabRow, [203](#)
- rhs\_
  - CglDuplicateRow, [39](#)
- rhsWeight
  - CglLandP::Parameters, [194](#)
- RhsWeightType
  - CglLandP, [75](#)
- rhsWeightType
  - CglLandP::Parameters, [194](#)
- RoundStats
  - LAP, [14](#)
- row\_to\_delete
  - parity\_ilp, [196](#)
- rowCut
  - LAP::Cuts, [180](#)
- rowCutPointer
  - CglStored, [158](#)
- rowCutPtr
  - CglUniqueRowCuts, [173](#)
- rowCuts



- CglProbing, 117
- RowSelectionStrategy
  - CglRedSplit2Param, 132
- rowSelectionStrategy\_
  - CglRedSplit2Param, 141
- rowSelectionStrategyLAP\_
  - CglRedSplit2Param, 141
- rowTypes
  - CglPreProcess, 109
- SC\_LINEAR
  - CglRedSplit2Param, 133
- SC\_LINEAR\_BOUNDED
  - CglRedSplit2Param, 133
- SC\_LOG\_BOUNDED
  - CglRedSplit2Param, 133
- SC\_NONE
  - CglRedSplit2Param, 133
- SC\_UNIFORM
  - CglRedSplit2Param, 133
- SC\_UNIFORM\_NZ
  - CglRedSplit2Param, 133
- SCL\_MAX\_DEGREE
  - CglClique, 25
- SCL\_MAX\_XJ\_MAX\_DEG
  - CglClique, 25
- SCL\_MIN\_DEGREE
  - CglClique, 25
- saveStuff
  - CglStored, 158
- scale
  - LAP, 14
- scaleCut
  - LAP::CglLandPSimplex, 83
- scl\_candidate\_length\_threshold
  - CglClique, 29
- scl\_next\_node\_method
  - CglClique, 25
- scl\_next\_node\_rule
  - CglClique, 29
- scl\_report\_result
  - CglClique, 29
- score
  - pool\_cut, 197
  - select\_cut, 199
- select\_cut, 198
  - ccoef, 199
  - crhs, 199
  - pool\_index, 199
  - score, 199
- SelectionRules
  - CglLandP, 74
- sense
  - DGG\_constraint\_t, 182
- sep\_012\_cut
  - Cgl012Cut, 16
- sepSpace
  - CglLandP::Parameters, 194
- Separating
  - LAP, 13
- separation\_graph, 199
  - even\_adj\_list, 200
  - ind, 200
  - nedges, 199
  - nnodes, 199
  - nodes, 199
  - odd\_adj\_list, 200
- SeparationSpaces
  - CglLandP, 75
- sequenceInCliqueEntry
  - CglTreeInfo.hpp, 225
- set\_given\_optsol
  - CglRedSplit, 121
- setAMax
  - CglTwomir, 169
- setAWAY
  - CglGMIParam, 57
- setAffectedInDisaggregation
  - CglProbing.hpp, 220
- setAffectedToUBInDisaggregation
  - CglProbing.hpp, 221
- setAggressiveness
  - CglCutGenerator, 33
- setApplicationData
  - CglPreProcess, 110
- setAway
  - CglGMIParam, 56
  - CglGomory, 66
  - CglRedSplit, 122
  - CglRedSplit2Param, 134
  - CglRedSplitParam, 146
  - CglTwomir, 170
- setAwayAtRoot
  - CglGomory, 66
  - CglTwomir, 171
- setBeta
  - CglLiftAndProject, 87
- setCHECK\_DUPLICATES
  - CglGMIParam, 59
- setCLEAN\_PROC
  - CglGMIParam, 60
- setCRITERION\_
  - CglMixedIntegerRounding, 91
  - CglMixedIntegerRounding2, 94
- setCheckDuplicates
  - CglGMIParam, 59
- setCleanProc
  - CglGMIParam, 60

setColsolToCut  
     LAP::CglLandPSimplex, 83  
 setColumnScalingBoundLAP  
     CglRedSplit2Param, 138  
 setColumnScalingStrategyLAP  
     CglRedSplit2Param, 138  
 setConditionNumberMultiplier  
     CglGomory, 66  
 setCutTypes  
     CglTwomir, 169  
 setCutoff  
     CglPreProcess, 107  
 setDoPreproc  
     CglMixedIntegerRounding, 91  
     CglMixedIntegerRounding2, 94  
     CglResidualCapacity, 152  
 setDoRowClique  
     CglClique, 27  
 setDoStarClique  
     CglClique, 27  
 setENFORCE\_SCALING  
     CglGMIParam, 60  
 setEPS  
     CglParam, 102  
     CglRedSplit, 122  
 setEPS\_COEFF  
     CglParam, 102  
     CglRedSplit, 122  
 setEPS\_COEFF\_LUB  
     CglRedSplit, 122  
     CglRedSplitParam, 147  
 setEPS\_ELIM  
     CglGMIParam, 57  
     CglRedSplit2Param, 134  
     CglRedSplitParam, 146  
 setEPS\_RELAX  
     CglRedSplit, 122  
 setEPS\_RELAX\_ABS  
     CglGMIParam, 57  
     CglRedSplit2Param, 134  
     CglRedSplitParam, 146  
 setEPS\_RELAX\_REL  
     CglGMIParam, 57  
     CglRedSplit2Param, 135  
     CglRedSplitParam, 146  
 setEnforceScaling  
     CglGMIParam, 60  
 setEps  
     CglGMIParam, 56  
 setEpsCoeff  
     CglGMIParam, 56  
 setEpsElim  
     CglGMIParam, 57  
 setEpsRelaxAbs  
     CglGMIParam, 57  
 setEpsRelaxRel  
     CglGMIParam, 58  
 setEpsilon  
     CglResidualCapacity, 152  
 setFlags  
     CglZeroHalf, 175  
 setFormulationRows  
     CglTwomir, 170  
 setGlobalCuts  
     CglCutGenerator, 33  
 setGomoryType  
     CglGomory, 66  
 setINFINIT  
     CglParam, 102  
 setInfinity  
     CglGMIParam, 55  
 setIntegralScaleCont  
     CglGMIParam, 60  
 setLUB  
     CglRedSplit, 122  
     CglRedSplitParam, 146  
 setLanguage  
     CglPreProcess, 110  
 setLargestFactorMultiplier  
     CglGomory, 67  
 setLimit  
     CglGomory, 66  
     CglRedSplit, 121  
 setLimitAtRoot  
     CglGomory, 66  
 setLogLevel  
     CglAllDifferent, 21  
     CglDuplicateRow, 38  
     CglLandP, 76  
     CglProbing, 116  
     LAP::CglLandPSimplex, 81  
 setMAX\_SUPP\_ABS  
     CglRedSplit2Param, 135  
 setMAX\_SUPP\_REL  
     CglRedSplit2Param, 135  
 setMAX\_SUPPORT  
     CglParam, 102  
 setMAX\_SUPPORT\_REL  
     CglGMIParam, 58  
 setMAXAGGR\_  
     CglMixedIntegerRounding, 90  
     CglMixedIntegerRounding2, 93  
 setMAXDYN  
     CglGMIParam, 58  
     CglRedSplit2Param, 135  
     CglRedSplitParam, 147  
 setMAXDYN\_LUB  
     CglRedSplitParam, 147

- setMINVIOL
  - CglGMIParam, [58](#)
  - CglRedSplit2Param, [135](#)
  - CglRedSplitParam, [147](#)
- setMULTIPLY\_
  - CglMixedIntegerRounding, [90](#)
  - CglMixedIntegerRounding2, [94](#)
- setMaxDyn
  - CglGMIParam, [58](#)
- setMaxElements
  - CglProbing, [116](#)
  - CglTwomir, [169](#)
- setMaxElementsRoot
  - CglProbing, [117](#)
  - CglTwomir, [169](#)
- setMaxFillIn
  - LAP::Validator, [205](#)
- setMaxInKnapsack
  - CglKnapsackCover, [72](#)
- setMaxLook
  - CglAllDifferent, [21](#)
  - CglProbing, [116](#)
- setMaxLookRoot
  - CglProbing, [116](#)
- setMaxNonzeroesTab
  - CglRedSplit2Param, [139](#)
- setMaxNumComputedCuts
  - CglRedSplit2Param, [139](#)
- setMaxNumCuts
  - CglFlowCover, [44](#)
  - CglRedSplit2Param, [139](#)
- setMaxPass
  - CglProbing, [115](#)
- setMaxPassRoot
  - CglProbing, [116](#)
- setMaxProbe
  - CglProbing, [116](#)
- setMaxProbeRoot
  - CglProbing, [116](#)
- setMaxRatio
  - LAP::Validator, [205](#)
- setMaxSumMultipliers
  - CglRedSplit2Param, [136](#)
- setMaxSupport
  - CglGMIParam, [56](#)
- setMaxSupportAbs
  - CglGMIParam, [56](#)
- setMaxSupportRel
  - CglGMIParam, [59](#)
- setMaxTab
  - CglRedSplit, [123](#)
  - CglRedSplitParam, [148](#)
- setMaximumDominated
  - CglDuplicateRow, [38](#)
- setMaximumEntries
  - CglOddHole, [100](#)
- setMaximumRhs
  - CglDuplicateRow, [38](#)
- setMinNormReduction
  - CglRedSplit2Param, [136](#)
- setMinReduc
  - CglRedSplit, [123](#)
  - CglRedSplitParam, [148](#)
- setMinViol
  - CglGMIParam, [58](#)
- setMinViolation
  - CglClique, [27](#)
  - LAP::Validator, [205](#)
- setMinimumViolation
  - CglOddHole, [100](#)
- setMinimumViolationPer
  - CglOddHole, [100](#)
- setMirScale
  - CglTwomir, [169](#)
- setMode
  - CglDuplicateRow, [38](#)
  - CglProbing, [115](#)
- setNormIsZero
  - CglRedSplit, [122](#)
  - CglRedSplit2Param, [136](#)
  - CglRedSplitParam, [148](#)
- setNormalization
  - CglRedSplit2Param, [136](#)
- setNumFlowCuts
  - CglFlowCover, [44](#)
- setOneFixesInCliqueEntry
  - CglTreeInfo.hpp, [225](#)
- setOptions
  - CglPreProcess, [109](#)
- setPacking\_
  - CglClique, [27](#)
- setParam
  - CglGMI, [50](#)
  - CglRedSplit, [121](#)
  - CglRedSplit2, [126](#)
- setProbingInfo
  - CglImplication, [70](#)
  - CglStored, [157](#)
- setRequiredViolation
  - CglStored, [157](#)
- setRhsScale
  - LAP::Validator, [205](#)
- setRowCliqueCandidateLengthThreshold
  - CglClique, [26](#)
- setRowCliqueReport
  - CglClique, [27](#)
- setRowCuts
  - CglProbing, [117](#)

- setSequenceInCliqueEntry
  - CglTreeInfo.hpp, 225
- setSi
  - LAP::CglLandPSimplex, 81
- setSkipGomory
  - CglRedSplit2Param, 139
- setStarCliqueCandidateLengthThreshold
  - CglClique, 26
- setStarCliqueNextNodeMethod
  - CglClique, 26
- setStarCliqueReport
  - CglClique, 26
- setTestedRowIndices
  - CglKnapsackCover, 72
- setTimeLimit
  - CglRedSplit2Param, 138
- setTolerance
  - CglResidualCapacity, 152
- setTrackRejection
  - CglGMI, 50
- setTwomirScale
  - CglTwomir, 169
- setTwomirType
  - CglTwomir, 171
- setUSE\_CG2
  - CglRedSplitParam, 147
- setUSE\_INTSLACKS
  - CglGMIParam, 59
  - CglRedSplit2Param, 135
  - CglRedSplitParam, 147
- setUseIntSlacks
  - CglGMIParam, 59
- setUsingObjective
  - CglProbing, 117
- setVal
  - CglFlowVUB, 46
  - CglMixIntRoundVUB, 95
  - CglMixIntRoundVUB2, 97
- setVar
  - CglFlowVUB, 46
  - CglMixIntRoundVUB, 95
  - CglMixIntRoundVUB2, 97
- setWhenAtUBInDisaggregation
  - CglProbing.hpp, 221
- setZeroOneInDisaggregation
  - CglProbing.hpp, 221
- short\_path\_node, 200
  - dist, 200
  - pred, 200
- si\_
  - LAP::TabRow, 203
- SimplexInterfaceError
  - CglLandP::SimplexInterfaceError, 201
- singleCutTimeLimit
  - CglLandP::Parameters, 193
- sizeDynamic
  - CglDuplicateRow, 38
- sizeDynamic\_
  - CglDuplicateRow, 40
- sizeRowCuts
  - CglStored, 158
  - CglUniqueRowCuts, 173
- skipGomory\_
  - CglRedSplit2Param, 142
- slack
  - parity\_ilp, 196
- SmallCoefficient
  - LAP::Validator, 204
- SmallViolation
  - LAP::Validator, 204
- snapshot
  - CglProbing, 115
- someFixed
  - CglPreProcess, 107
- sp\_col\_ind
  - CglClique, 28
- sp\_col\_start
  - CglClique, 28
- sp\_colsol
  - CglClique, 28
- sp\_numcols
  - CglClique, 28
- sp\_numrows
  - CglClique, 27
- sp\_orig\_col\_ind
  - CglClique, 28
- sp\_orig\_row\_ind
  - CglClique, 27
- sp\_row\_ind
  - CglClique, 28
- sp\_row\_start
  - CglClique, 28
- src/CglAllDifferent/CglAllDifferent.hpp, 206
- src/CglClique/CglClique.hpp, 206
- src/CglConfig.h, 207
- src/CglCutGenerator.hpp, 207
- src/CglDuplicateRow/CglDuplicateRow.hpp, 207
- src/CglFlowCover/CglFlowCover.hpp, 207
- src/CglGMI/CglGMI.hpp, 210
- src/CglGMI/CglGMIParam.hpp, 210
- src/CglGomory/CglGomory.hpp, 211
- src/CglKnapsackCover/CglKnapsackCover.hpp, 211
- src/CglLandP/CglLandP.hpp, 212
- src/CglLandP/CglLandPMessages.hpp, 212
- src/CglLandP/CglLandPSimplex.hpp, 213
- src/CglLandP/CglLandPTabRow.hpp, 214
- src/CglLandP/CglLandPUtils.hpp, 214
- src/CglLandP/CglLandPValidator.hpp, 215

- src/CglLiftAndProject/CglLiftAndProject.hpp, 215
- src/CglMessage.hpp, 216
- src/CglMixedIntegerRounding/CglMixedIntegerRounding.-hpp, 217
- src/CglMixedIntegerRounding2/CglMixedIntegerRounding2.-hpp, 218
- src/CglOddHole/CglOddHole.hpp, 219
- src/CglParam.hpp, 219
- src/CglPreProcess/CglPreProcess.hpp, 219
- src/CglProbing/CglProbing.hpp, 220
- src/CglRedSplit/CglRedSplit.hpp, 221
- src/CglRedSplit/CglRedSplitParam.hpp, 222
- src/CglRedSplit2/CglRedSplit2.hpp, 222
- src/CglRedSplit2/CglRedSplit2Param.hpp, 222
- src/CglResidualCapacity/CglResidualCapacity.hpp, 223
- src/CglSimpleRounding/CglSimpleRounding.hpp, 223
- src/CglStored.hpp, 224
- src/CglTreeInfo.hpp, 224
- src/CglTwomir/CglTwomir.hpp, 225
- src/CglZeroHalf/Cgl012cut.hpp, 235
- src/CglZeroHalf/CglZeroHalf.hpp, 236
- src/config\_cgl\_default.h, 237
- src/config\_default.h, 238
- startModel
  - CglPreProcess, 107
- startSOS
  - CglPreProcess, 108
- storedCuts\_
  - CglDuplicateRow, 39
- strengthen
  - CglLandP::Parameters, 194
- strengthenRow
  - CglTreeInfo, 162
- strengthenedIntersectionCutCoef
  - LAP::CglLandPSimplex, 82
- SupportSize
  - CglLandP, 75
- switchOffExpensive
  - CglKnapsackCover, 72
- switchOnExpensive
  - CglKnapsackCover, 72
- t\_max
  - cutParams, 178
- t\_min
  - cutParams, 178
- TabRow
  - LAP::TabRow, 202
- tightLower
  - CglProbing, 115
  - CglStored, 158
- tightUpper
  - CglProbing, 115
  - CglStored, 158
- tightenBounds
  - CglProbing, 115
- tightenPrimalBounds
  - CglPreProcess, 107
- tightenThese
  - CglProbing, 117
- tiltLandPcut
  - CglRedSplit2, 126
- timeLimit
  - CglLandP::Parameters, 193
- timeLimit\_
  - CglRedSplit2Param, 142
- to
  - cgl\_arc, 17
- toOne
  - CglTreeProbingInfo, 165
- toOne\_
  - CglTreeProbingInfo, 166
- toZero
  - CglTreeProbingInfo, 165
- toZero\_
  - CglTreeProbingInfo, 166
- twomirType
  - CglTwomir, 171
- type
  - info\_weak, 188
- type\_even\_weak
  - parity\_ilp, 196
- type\_odd\_weak
  - parity\_ilp, 196
- typeSOS
  - CglPreProcess, 108
- UB\_MAX
  - CglTwomir.hpp, 232
- USE\_CG2
  - CglRedSplitParam, 149
- USE\_INTSLACKS
  - CglGMIPParam, 62
  - CglRedSplit2Param, 140
  - CglRedSplitParam, 149
- ub
  - DGG\_data\_t, 183
- Uniform
  - CglLandP, 75
- Unweighted
  - CglLandP, 75
- update
  - CglPreProcess, 109
- upper\_
  - CglFlowVUB, 47
- useAlternativeFactorization
  - CglGomory, 67
- useTableauRow

- CglLandP::Parameters, 194
- V2l
  - CglTwomir.hpp, 234
- val\_
  - CglMixIntRoundVUB, 96
  - CglMixIntRoundVUB2, 97
- Validator
  - LAP::Validator, 204
- validator
  - CglLandP, 76
- var
  - info\_weak, 188
- var\_
  - CglMixIntRoundVUB, 95
  - CglMixIntRoundVUB2, 97
- varInd\_
  - CglFlowVUB, 46
- violation
  - cut, 177
- vlb
  - ilp, 187
- vub
  - ilp, 187
- WarnBadRhsComputation
  - LAP, 13
- WarnBadRowComputation
  - LAP, 13
- WarnBadSigmaComputation
  - LAP, 13
- WarnFailedBestImprovingCol
  - LAP, 13
- WarnFailedPivotIf
  - LAP, 13
- WarnFailedPivotTol
  - LAP, 13
- WarnGiveUpRow
  - LAP, 13
- weak
  - edge, 186
- weight
  - cycle, 180
  - edge, 186
- WeightBoth
  - CglLandP, 75
- WeightLHS
  - CglLandP, 75
- WeightRHS
  - CglLandP, 75
- weightSOS
  - CglPreProcess, 108
- WeightsStats
  - LAP, 13
- WhenEnteringBasis
  - CglLandP, 75
  - whenAtUBInDisaggregation
    - CglProbing.hpp, 221
  - whichSOS
    - CglPreProcess, 108
- x
  - DGG\_data\_t, 183
- xstar
  - ilp, 187
  - parity\_ilp, 196
- zeroOneInDisaggregation
  - CglProbing.hpp, 220