Part 4: Cut your way to ILPs: Instructions

Help Center

Part 4 (Cut your way to ILP)

This assignment asks you to implement cutting plane method on top of your solver to give it full integer linear programming capabilities. We will focus purely on using Gomory cuts for Integer Linear Programs (ILPs). `Zip files containing unit tests and assignment tests are available here: ZIP or TAR.GZ . The dictionary printOuts for unit tests are here: printOut.pdf .`

Instructions

The goal is to input a dictionary for the LP relaxation of the original ILP and use cutting plane methods.

Inputs

We are given a dictionary in the usual format that we have used for all the previous steps.

- Assume that the problem and slack variables in the given dictionary are integer valued.
- Even if the dictionary has floating point entries, start with the given input dictionary, and do not attempt to scale the problem.

Steps

The goal is to iterate the following steps:

- Solve the current dictionary to obtain a final dictionary. (`Reuse your existing code`)
- Add **all** cutting planes corresponding to all non integer rows to the cutting plane.
- The resulting dictionary will be primal infeasible. You can proceed by solving it using dual simplex. We have demonstrated this in our unit tests (see printOut.pdf file accompanying these tests).
- Alternatively, instead of using Dual dictionaries, you can go back to initialization phase (auxiliary problem) and then use optimization phase to get a final dictionary.

We note that constructing dual dictionaries may require some very simple steps to rearrange the data in the primal dictionary. You can reuse the pivoting code you have already implemented to pivot over the dual dictionaries. But do not forget to translate them back to primal form at the end. Some pitfalls to be noted:

- Floating point issues are going to be there. Use some of the tricks we already did for initialization phase (step 3).

- Define integer values carefully: allow numbers like 1 + 1E-10 or 1 - 1E-10 as integers. In other words, your test for integrality must accepts numbers with fractional parts: 0.00000001 or 0.999999999. We recommend using 10^{-10} as a tolerance for testing integrality.

Outputs

The output format is simply a single file with a single line. **Infeasible problem** The file must simply say `infeasible` (all small letters) **Unbounded Problem** The file must simply say `unbounded` (all small letters) **Optimal solution** Simply write the solution correct to two decimal points in a file. The file must just have one number which is the optimal objective value. Note that it is helpful to write outputs as "12.0" instead of "12". Somehow the latter seems to confuse the autograder.

Zip Files

`Zip files containing unit tests and assignment tests are available here: ZIP or TAR.GZ . The dictionary printOuts for unit tests are here: printOut.pdf .` As usual, the zip file has two directories: unitTests and assignmentParts. The answers to the unitTests are given in printOut.pdf file which can be used as a reference to compare the steps of your code to ours and thus debug. All the best.