Step 4: Cut your way to ILPs: Instructions

Help

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: TAR.GZ or ZIP .`

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 but dual feasible. You can proceed by solving it using dualization (no need to change objective) and solving the dual.
- 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: but this will be more expensive.

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). But as a warning ILP solving using cutting planes will be extremely sensitive to floating point errors. We apologize for the inconvenience that this will create.
- Define integer values carefully: allow numbers like n + 1E-6or n - 1E-6 as integers. In other words, your test for integrality must accepts numbers with fractional parts: 0.000001 or 0.999999. We ourselves 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` **Unbounded Problem** The file must simply say `unbounded` **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: TAR.GZ or ZIP .` As usual, the zip file has two directories: unitTests and assignmentParts. The unitTests folder has many unit tests with sample outputs and step by step executions that you can use to compare your answer. All the best.