Assignment 7:

Installation

1. Open a command prompt in the folder, and type ‘pip install –r requirements.txt’ to make sure that all the dependencies are installed
2. To run the file, on a command prompt type ‘python branch_and_cut.py’

Explanation

1. Describe the method.

A branch cut is a curve (with ends possibly open, closed, or half-open) in the complex plane across which an analytic multivalued function is discontinuous. For convenience, branch cuts are often taken as lines or line segments. Branch cuts (even those consisting of curves) are also known as cut lines, slits, or branch lines.

For example, consider the function z^2 which maps each complex number z to a well-defined number z^2. Its inverse function $\sqrt{z}$, on the other hand, maps, for example, the value $z = 1$ to $\sqrt{1} = \pm 1$. While a unique principal value can be chosen for such functions (in this case, the principal square root is the positive one), the choices cannot be made continuous over the whole complex plane. Instead, lines of discontinuity must occur. The most common approach for dealing with these discontinuities is the adoption of so-called branch cuts.

In other words, **Branch and cut** is a method of combinatorial optimization for solving integer linear programs (ILPs), that is, linear programming (LP) problems where some or all the unknowns are restricted to integer values. Branch and cut involves running a branch and bound algorithm and using cutting planes to tighten the linear programming relaxations. Note that if cuts are only used to tighten the initial LP relaxation, the algorithm is called **cut and branch.**

For example, assume an ILP with a maximization problem.

The method solves the linear program without the integer constraint using the regular simplex algorithm. When an optimal solution is obtained, and this solution has a non-integer value for a variable that is supposed to be integer, a cutting plane algorithm may be used to find further linear constraints which are satisfied by all feasible integer points but violated by the current fractional solution. These inequalities may be added to the linear program, such that resolving it will yield a different solution which is hopefully "less fractional".

At this point, the branch and bound part of the algorithm is started. The problem is split into multiple (usually two) versions. The new linear programs are then solved using the simplex method and the process repeats. During the branch and bound process, non-integral solutions to LP relaxations serve as upper bounds and integral solutions serve as lower bounds. A node can be pruned if an upper bound is lower than an existing lower bound.

Further, when solving the LP relaxations, additional cutting planes may be generated, which may be either *global cuts*, i.e., valid for all feasible integer solutions, or *local cuts*, meaning that they are satisfied by all solutions fulfilling the side constraints from the currently considered branch and bound subtree.

2. Write the Algorithm.

```
1. Add the initial ILP to L, the list of active problems
2. Set x* = null and v* = -inf
3. while the L is not empty
    1. Select and remove a problem from L
    2. Solve the LP relaxation of the problem.
    3. If the solution is infeasible, go back to 3 (while). Otherwise denote the
       solution by x with objective value v.
    4. If v < v*, go back to 3.
    5. If x is integer, set v* <- v, x* <- x and go back to 3.
    6. If desired, search for cutting planes that are violated by x. If any are found,
       add them to the LP relaxation and return to 3.2.
    7. Branch to partition the problem into new problems with restricted
       feasible regions. Add these problem to L and go back to 3
4. return x*
```

3. What are the branching strategies?

- **Most Infeasible Branching** This branching strategy chooses the variable with the fractional part closest to 0.5.
- **Pseudo Cost Branching** The basic idea of this strategy is to keep track for each variable xi the change in the objective function when this variable was previously chosen as the variable to branch on. The strategy then chooses the variable that is predicted to have the most change on the objective function based on past changes when it was chosen as the branching variable. Note that pseudo cost branching is initially uninformative in the search since few variables have been branched on.
- **Strong Branching** Strong branching involves testing which of the candidate variable gives the best improvement to the objective function before actually branching on them.
- **Full strong branching** tests all candidate variables and can be computationally expensive. The computational cost can be reduced by only considering a subset of the candidate variables and not solving each of the corresponding LP relaxations to completion.

There are also a large number of variations of these branching strategies, such as using strong branching early on when pseudo cost branching is relatively uninformative and then switching to pseudo cost branching later when there is enough branching history for pseudo cost to be informative.

The program attached contains branch cut solution for complex number space