GLPK Case Study 3 - 60016 Operations Research

In this case study, we study a famous result in operations research. In 1972, Klee and Minty[1] showed that, in the worst-case, the simplex algorithm needs to visit all the vertices of the feasible set before finding the optimal one. A feasible set of this kind is called a *Klee-Minty cube.*

Consider the following Klee-Minty problem:

$$\min z = -10^{n-1}x_1 - 10^{n-2}x_2 - \ldots - 10x_{n-1} - x_n$$

subject to

$$\begin{array}{rcrcccccccl}
x_1 & & & & & & & & & \leq & 1 \\
20x_1 & + & x_2 & & & & & & & \leq & 100 \\
200x_1 & + & 20x_2 & + & x_3 & & & & & \leq & 100^2 \\
\vdots & & & & & & & & & & \\
2\cdot 10^{n-1}x_1 & + & 2\cdot 10^{n-2}x_2 & + & \ldots & + & 2\cdot 10x_{n-1} & + & x_n & \leq & 100^{n-1}
\end{array}$$

and

$$x_1 \geq 0,\ x_2 \geq 0,\ \ldots,\ x_n \geq 0$$

The problem has n constraints and n variables.

1. Write a GMPL model to solve the problem for arbitrary n. Call the resulting file klee.mod.
2. To show the behavior of a "textbook" simplex algorithm implementation on this problem, run `glpsol -m klee.mod -o klee.out --norelax --nosteep --noscale --nopresol` for $n = 2, 4, 8$. Determine the general structure of the optimal solution. Then comment on the number of iterations performed by the solver. Is the simplex algorithm visiting all the vertices?
3. Run glpsol with the same parameters for $n = 20, 50, 500$. Discuss qualitatively the behavior of the solver.
4. Repeat the experiment for $n = 20$ after appending the `--exact` parameter. What happened?
5. The *steepest edge* technique is a criteria to select the NBV that enters the basis in the pivoting operation of the simplex algorithm. It is an alternative to the techniques we see in the course and it is enabled by default in glpsol. It work as follows. Given the current vertex V (current BFS), it seeks for the neighbouring vertex V' such that the edge VV' forms the smallest angle with the direction of improvement of the objective function, i.e., the gradient of the objective. Remove the `--nosteep` parameter, thus re-enabling the steepest edge technique in glpsol, and evaluate $n = 2, 4, 8$.
6. The *interior point method* is an algorithm to solve LPs searching through the interior of the feasible set. Its main advantage is that, even in the worst case, for a problem with n variables and n constraints, it requires a computational effort that is proportional to a polynomial in

[1] Klee, Victor; Minty, George J. (1972). "How good is the simplex algorithm?". In Shisha, Oved. Inequalities III (Proceedings of the Third Symposium on Inequalities held at the University of California, Los Angeles, Calif., September 1–9, 1969, dedicated to the memory of Theodore S. Motzkin). New York-London: Academic Press. pp. 159–175.

n (e.g., $n^3 - 3n^2$). For the simplex algorithm, instead, this effort can grow exponentially with n and thus become much larger than in the interior point method. To run the interior point method, use glpsol -m klee.mod -o klee.out --interior --noscale --nopresol. Compare the number of iterations of the simplex algorithm and of the interior point method for $n = 2, 4, 8$.

Solution 1.

Listing 1: klee.mod

```
# klee.mod

param n := 2;

# DECISION VARIABLES
var x {i in 1..n}, >= 0;

# LINEAR PROGRAM

minimize z: sum {j in 1..n} -10^(n-j)*x[j];

s.t.
c1 {i in 1..n}: 2*sum {j in 1..(i-1)} 10^(i-j)*x[j] + x[i] <= 100^(i-1);

# SOLVE LINEAR PROGRAM
solve;
end;
```

2. The following table summarizes the results:

n	z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	iterations
2	-10^2	0	10^2							4
4	-10^6	0	0	0	10^6					16
8	-10^{14}	0	0	0	0	0	0	0	10^{14}	256

From these results we conclude that $z^* = -10^{2n-2}$ and $x^* = (x_1, \ldots, x_{n-1}, x_n) = (0, \ldots, 0, 10^n)$. The number of iterations is 2^n. This can be verified with GLPK for other values of n. An illustration of the Klee-Minty cube for $n = 3$ and the sequence of vertices visited by the simplex algorithm is shown in Figure 1.

If we assume that no degenerate basis is found during the iterations, which can be proved formally, we can conclude that 2^n is surely the total number of vertices for the feasible set. This can be seen from the fact that the feasible set is very similar to a cube in n dimension, which has 2^n vertices. To see this, note that we can rewrite the conditions of the feasible set as:

$$0 \leq x_i \leq 100^{i-1} - 2 \cdot 10^{i-1} x_1 - 2 \cdot 10^{i-2} x_2 - \ldots - 2 \cdot 10 x_{i-1}$$

but since 100^{i-1} is at least an order of magnitude larger than all other coefficients, this feasible set is just a small perturbation of a cube defined by the conditions

$$0 \leq x_i \leq 100^{i-1}$$

3. In all three cases, GLPK returns an error. We discuss these errors to illustrate the variety of problems that can arise in real-world LP solving.

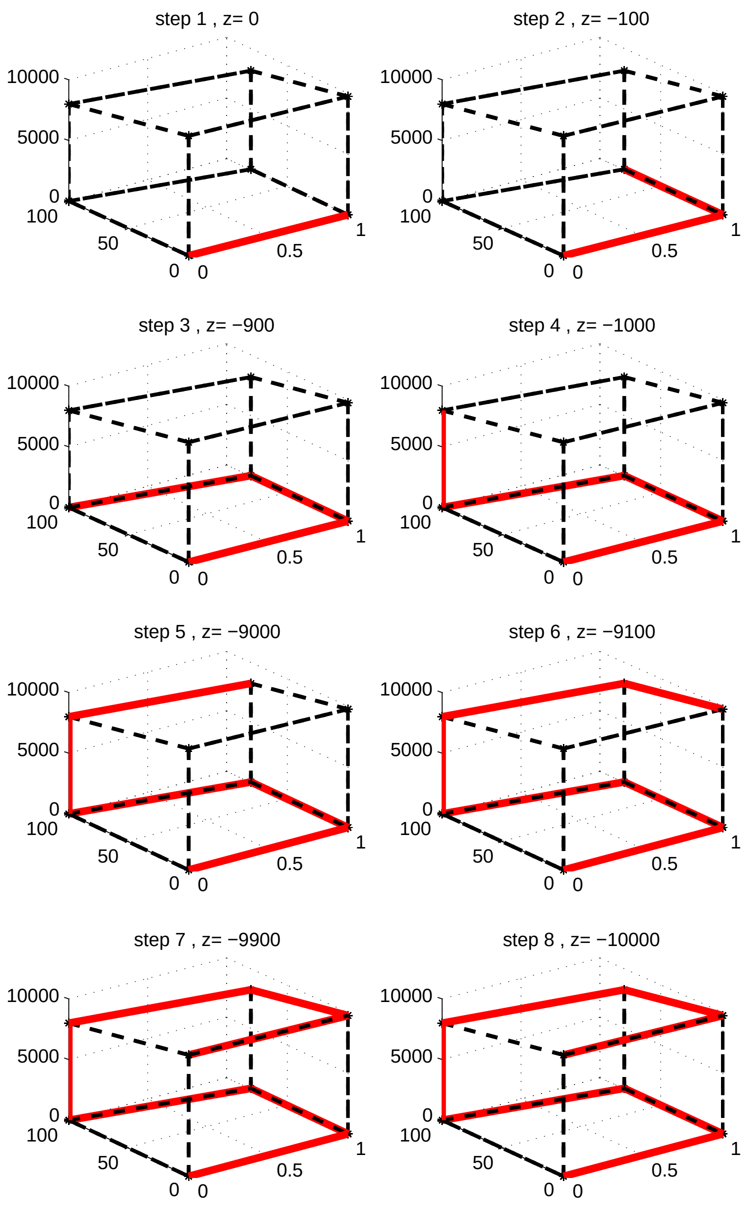

Figure 1: Standard simplex algorithm for $n = 3$.

n	Error
20	Warning: numerical instability (primal simplex, phase II); Error: unable to choose basic variable on phase I.
50	Error: unable to factorize the basis matrix (1); Sorry, basis recovery procedure not implemented yet.
500	klee.mod:10: 10 ** 499; floating-point overflow; MathProg model processing error.

All three problems arise due to numerical errors. Modern PCs have 15–17 significant decimal digits precision, however we see from the definition of the Klee-Minty problem that each constraint will have a coefficient 1 on x_i and a known term that can be as large as 100^{n-1}. Therefore, there will be $2(n-1)$ orders of magnitude of difference between the coefficients. This implies that, upon performing the elementary row operations, terms that are very small may be rounded to zero, introducing small inaccuracies, called *round-off errors*. Round-off errors, when accumulate, can generate fatal errors in the simplex algorithm.

The case $n = 20$ represents one such case. The glpsol solver has detected that the current basic solution has became infeasible due to excessive round-off errors, i.e., some variables are now negative. In this case, the solver automatically switches to the phase I of the simplex algorithm to find a new feasible solution, adding artificial variables to the current basic representation to find a feasible basis, and then continue the search. The final error hints at the failure of this recovery procedure due to numerical difficulties. At this point the simplex algorithm stops.

The case $n = 50$ shows another fatal error. The glpsol solver is constructing the basic representation for the linear program. It does so using elementary row operations after transforming the basis B into the product of a lower triangular matrix L and a upper triangular matrix U. This is transformation is called a LU factorization and it a convenient form to perform Gaussian elimination. The LU transformation has failed in the case $n = 50$ due to the numerical round-offs and GLPK must stop.

The case $n = 500$ represents another fatal error. The GMPL parser has detected that $b_n = 10^{499}$, but the largest number that can be represented with the precision of ordinary PCs is around 10^{323}. This GMPL cannot instantiate a floating-point number of the requested magnitude.

4. Typing glpsol --help we can see that the --exact parameter stands for "use simplex method based on exact arithmetic". Informally, by exact arithmetic we mean that glpsol uses additional floating-point variables to store all the significant digits that are required to ensure that the calculations are correct[2]. The drawback is that, if a number is represented by K floating-point variables, the number of operations that the PC has to do to multiply or sum two numbers is several times more than before. Therefore, we see in this experiment that the final solution for $n = 20$ is now correct, but the time required by the solver to obtain it is much greater than without the --exact parameter.

5. We now compare the steepest edge method against the standard simplex. Running glpsol with and without the --nosteep parameter we find that:

n	iterations standard simplex	iterations steepest edge
2	4	4
4	16	8
8	256	16

We see that the number of iterations of the steepest edge method is markedly better than for the standard simplex method. However, why is this happening? The answer is *chance*! Let us run again the same experiments, but adding the --exact parameter to both standard simplex and steepest edge simplex. Surprisingly, we now discover that

[2]Further details can be found at the GNU GMP project page: https://gmplib.org/. GNU GMP is the default library used by glpsol to implement exact arithmetic

n	iterations	iterations
	standard simplex	steepest edge
2	4	4
4	16	16
8	256	256

Hence, without the help of casual round-off errors, the steepest edge method would be no better than the standard simplex algorithm.

The lesson to be taken from this experiment is that linear programs can be numerically delicate, especially when coefficients differ by orders of magnitude. In these situations, it is worth comparing alternative solution algorithms and numerical tolerances to identify the more robust solution for a given class of problems.

6. The experiment returns the following result:

n	iterations	iterations
	standard simplex	interior point
2	4	11
4	16	11
8	256	12

We see that the number of iterations of the interior point method is roughly constant, whereas the performance of the standard simplex deteriorates exponentially fast with the problem size. Furthermore, even if we set $n = 50$ the interior point method finds a solution in just 14 iterations.