Scientific Compuing

Lecture 06: Linear Programming
Part 02

Vladimir Kulyukin
Department of Computer Science
Utah State University

Announcements

I put my Python source for a multiprocessing optimization of computing the Leibnitz determinants in the Lecture 04 PDF (cf. slides 18 – 21) in Canvas.

One way to handle destructive and non-destructive matrix unit tests is to implement a destructive function and then write a non-destructive wrapper function that makes a copy of the input matrix, calls the destructive version on the copy, and returns the results of the latter call. This way the input matrix remains intact.

Example 1

Recall this problem that we discussed at the end of Lecture 05 on 01/25/2024.

A hiker believes that on a long hike she will need snacks with at least 600 calories. She plans to take chocolate and raisins. The chocolate has 150 calories per ounce and raisins have 80 calories per ounce. Find and graph the system of inequalities that satisfy the hiker's constraints.

Example 1

1. Let us have the following decision variables: x – number of ounces of chocolate; y – number of ounces of raisins.

2. We have 3 constraints for this problem (one "real" constraint, i.e., constraint 1, and two "common sense" constraints, i.e., constraints 2 and 3):

   1. $150x + 80y \geq 600$;
   2. $x \geq 0$;
   3. $y \geq 0$.

Example 1

Once we have the decision variables and constraints, we can determine the feasible set of the problem. The feasible set F for an LP problem tells us where a solution will be found if it exists. In this case, a solution is a 2-tuple (x, y), where x is the number of ounces of number of ounces of chocolate and y is the number of ounces of raisins.

3. The feasible set for this problem is the intersection of the 3 sets: $\{(x, y)|150x + 80y \geq 600\}$ (this is the set of 2-tuples (x, y) that satisfy constraint 1); $\{(x, y)|x \geq 0\}$ (this is the set of 2-tuples (x, y) that satisfy constraint 2; $\{(x, y)|y \geq 0\}$ (this is the set of 2-tuples (x, y) that satisfy constraint 3), i.e.,

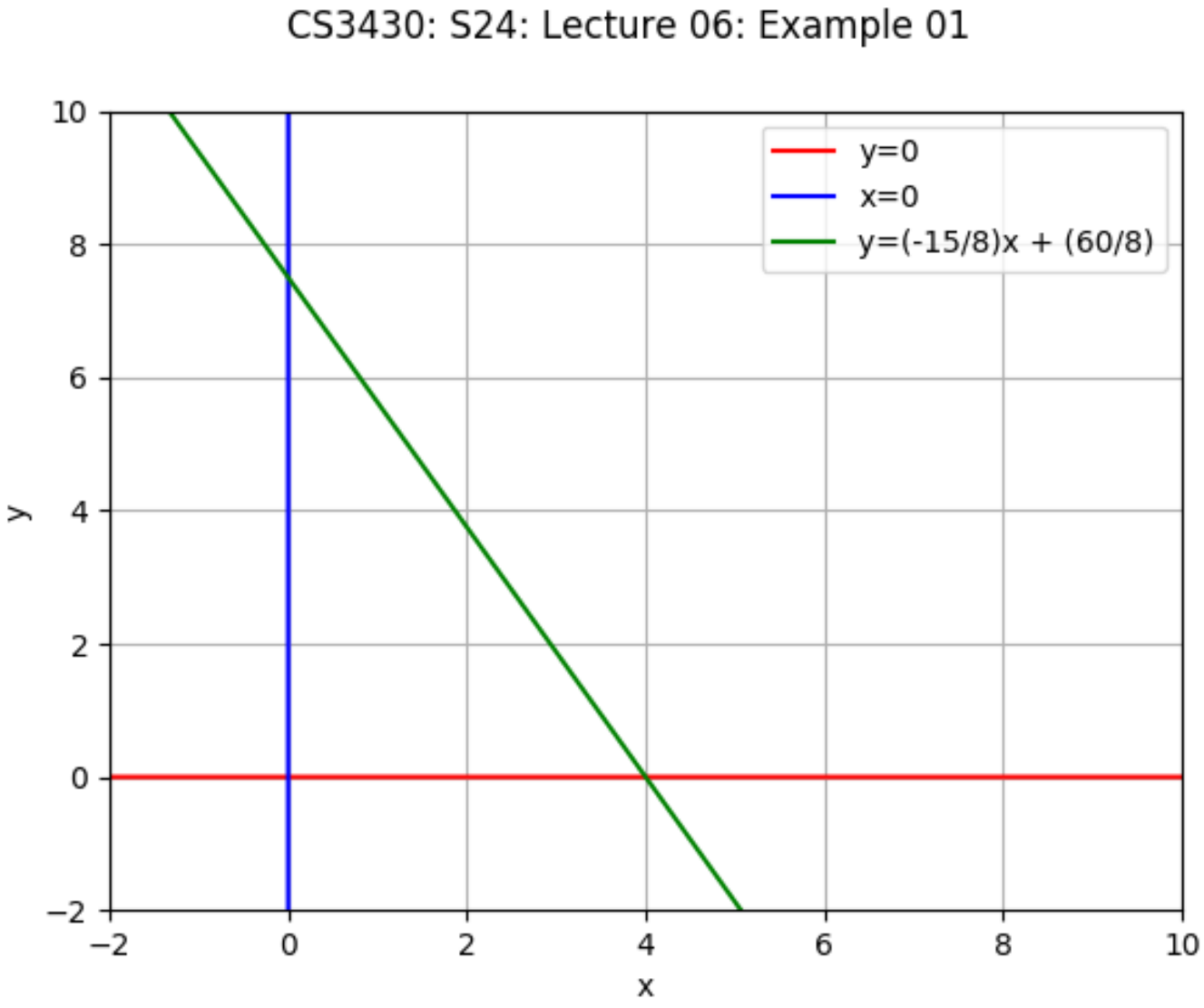
$$\{(x, y)|150x + 80y \geq 600\} \cap \{(x, y)|x \geq 0\} \cap \{(x, y)|y \geq 0\}.$$

Example 1

We can write the following function (source code in `cs3430_s24_lecture_06.py`) to graph the feasible set for this problem. The local function $y(x)$ defines the only real constraint for this problem.

```
def lec_06_example_01():
    def y(x): return (-15/8)*x + (60/8)
    xvals = np.linspace(-2, 10, 10000)
    yvals1 = np.array([y(x) for x in xvals])
    yvals = np.linspace(-2, 10, 10000)
    zero_x = np.linspace(0, 0, 10000)
    zero_y = np.linspace(0, 0, 10000)
    fig1 = plt.figure(1)
    fig1.suptitle('CS3430: S24: Lecture 06: Example 01')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.ylim([-2, 10])
    plt.xlim([-2, 10])
    plt.grid()
    plt.plot(xvals, zero_y, label='y=0', c='r')
    plt.plot(zero_x, yvals, label='x=0', c='b')
    plt.plot(xvals, yvals1, label='y=(-15/8)x + (60/8)', c='g')
    plt.legend(loc='best')
    plt.show()
```

Example 1


CS3430: S24: Lecture 06: Example 01
y=0
x=0
y=(-15/8)x + (60/8)
10
8
6
4
2
0
−2
y
−2
0
2
4
6
8
10
x

Example 2

Graph the set of points satisfying the inequalities

1. $x \geq 1$;
2. $x \leq 2y$;
3. $3x + 4y \leq 12$.

The corresponding boundary lines are

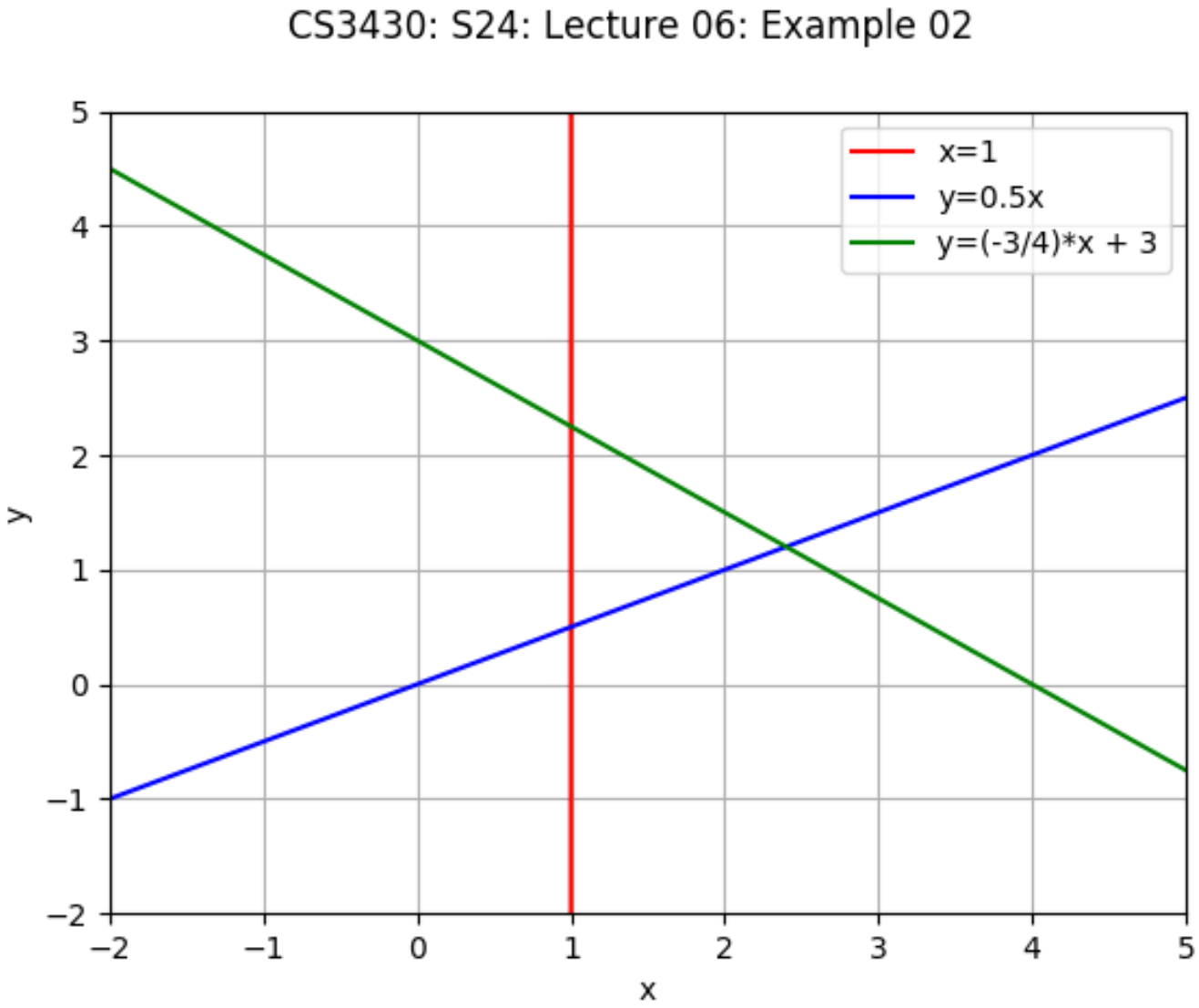
1. $x = 1$;
2. $x - 2y = 0$;
3. $3x + 4y = 12$.

Example 2

We can write the following function (source code in `graphs2D.py`) to graph the feasible set for this problem. The local function $y1(x)$ defines the line $x - 2y = 0$. The local function $y2(x)$ defines the line $3x + 4y = 12$.

```
def lec_06_example_02():
    def y1(x): return 0.5*x
    def y2(x): return (-3/4.0)*x + 3
    xvals = np.linspace(-2, 10, 10000)
    xvals1 = np.linspace(1, 1, 10000)
    yvals = np.linspace(-2, 10, 10000)
    yvals2 = np.array([y1(x) for x in xvals])
    yvals3 = np.array([y2(x) for x in xvals])
    fig1 = plt.figure(1)
    fig1.suptitle('CS3430: S24: Lecture 06: Example 02')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.ylim([-2, 5])
    plt.xlim([-2, 5])
    plt.grid()
    plt.plot(xvals1, yvals, label='x=1', c='r')
    plt.plot(xvals, yvals2, label='y=0.5x', c='b')
    plt.plot(xvals, yvals3, label='y=(-3/4)*x + 3', c='g')
    plt.legend(loc='best')
    plt.show()
```

Example 2


CS3430: S24: Lecture 06: Example 02
x=1
y=0.5x
y=(-3/4)*x + 3
y
x
5
4
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4
5

Bounded and Unbounded Sets

A set of points in the plane is **bounded** if it is contained in some circle centered at (0, 0). Otherwise, it is **unbounded**.

Corner/Extreme Points

A point T is a **corner/extreme point** of a feasible set if every line segment contained in the set that contains T has T as one of its end points.

A Fundamental LP Theorem

Let F be a feasible set for an LP problem such that $F \neq \emptyset$ (i.e., F is not the empty set). Let f be an objective function. If F is bounded, then f attains its maximum/minimum value at a corner point of F. If F is unbounded, then f attains its maximum/minimum value at a corner point or takes arbitrarily large positive/negative values on F.

LP Algorithm in 2D

1. Graph the feasible set (see `graph2D.py` from the Lecture 05 zip and `cs3430_s24_lecture_06.py` in this lecture's zip on how to do it in Python with `matplotlib`).

2. Determine the coordinates of each corner point of the feasible set. This is can be done with `np.linalg.solve()` to do it. An example on the next slide.

3. Evaluate the objective function f at each corner point. This step works because of the LP Theorem on the previous slide.

4. If F is bounded, choose the corner point at which the objective function f achieves a maximum/minimum (break ties arbitrarily).

5. If F is unbounded, then

5a. if f takes arbitrarily large positive/negative values on F, then there is no solution;

5b. If f doesn't take arbitrarily large positive/negative values on F, then return the corner point on which f attains a maximum/minimum.

Computing Coordinates of Corner Points with Gauss-Jordan Elimination

Let us assume that we need to find the intersection of $3x + 4y = 12$ and $x = 2y$. This is one of the corner points in Example 2. After we turn the second line equation into $x - 2y = 0$, then we can use `np.linalg.solve()` to compute this intersection.

```
import numpy as np
import numpy.linalg
A = np.array([[3, 4], [1, -2]], dtype=float)
x = np.linalg.solve(A, b)
assert np.allclose(np.dot(A, x), b)
```

Testing Corner Points

Suppose that we know all the corner points of a feasible set for a given LP problem.

We can brute force a solution by checking the values of the objective function at each corner point and returning the corner point at which the objective function achieves a maximum/minimum.

It is a reasonable idea if the number of corner points is reasonably small.

It is not a reasonable idea in higher dimensional spaces where the number of corner points can be exponential.

SMP: A Standard Maximum Problem

A problem is an SMP if

1. There is a linear function to maximize;
2. All variables are constrained to be non-negative;
3. All constraints are of the form: linear function of the variables $\leq$ positive number.

Example: An SMP

maximize: $p = 4x + 10y + 6z$ (p typically stands for *profit*)

subject to:

1. $x \geq 0$;
2. $y \geq 0$;
3. $z \geq 0$;
4. $15x + 20y + 30z \leq 4800$;
5. $30x + 60y + 45z \leq 10800$;
6. $6y + 3z \leq 1800$.

Example: NOT An SMP

maximize: $p = x - y + 3z$

subject to:

1. $x \geq 0$;
2. $y \geq 0$;
3. $z \geq 0$;
4. $3x - y + 4z \leq 0$;
5. $-x + 5y - 3z \geq 8$;

Constraint 4 does not have a positive number on the right hand side.

Constraint 5, after we multiply both sides by -1 to turn $\geq$ into $\leq$, will have -8 on the right hand side.

Slack Variables

Let us consider this SMP.

maximize $p = 3x + 8y + 6z$

subject to

1. $x \geq 0$, $y \geq 0$, $z \geq 0$;
2. $20x + 4y + 4z \leq 6000$;
3. $8x + 8y + 4z \leq 10000$;
4. $8x + 4y + 2z \leq 4000$.

Slack Variables

To satisfy constraint 2 on the previous slide, we have to find x, y, z such that $20x + 4y + 4z \leq 6000$.

What if for some values of x, y, z we have $20x + 4y + 4z < 6000$? Then there must be a number $u > 0$ such that $20x + 4y + 4z + u = 6000$. u is a **slack** variable for constraint 2. We can add u even if we have $20x + 4y + 4z \leq 6000$. In that case, $u \geq 0$.

We can introduce slack variables for constraints 3 and 4 and they will look as follows:

- $8x + 8y + 4z + v = 10000$;
- $8x + 4y + 2z + w = 4000$.

For this problem, we have 3 decision variables x, y, and z and 3 slack variables u, v, and w.

Basic Solutions

Once we add the slack variables, there is always at least one solution which has non-negative values for all variables, i.e., it satisifes the common sense constraints. This solution is obtained by setting all decision variables to 0 and each slack variable to the right-hand side of the equation. For example, consider the following constraints with the slack variables u, v, and w:

1. $20x + 4y + 4z + u = 600$;
2. $8x + 8y + 4z + v = 1000$;
3. $8x + 4y + 2z + w = 4000$.

Then $x = 0, y = 0, z = 0, u = 600, v = 1000, w = 4000$ is a solution. The associated feasible vector is $[0, 0, 0]$. In other words, $x = 0$, $y = 0$, $z = 0$ is a feasible solution in that it satisfies the constraints.

A **basic solution** to an SMP is a feasible vector that satisfies all constraints.

Example 3

Convert the following SMP problem to the corresponding augmented matrix.

Maximize $p = 2x + y$ subject to

1. $x \geq 0$;
2. $y \geq 0$;
3. $x \leq 40$;
4. $3x + 4y \leq 240$;
5. $2x + y \leq 100$.

Example 3

Let us add the slacks to constraints 3, 4, 5.

1. $u + 0v + 0w + x + 0y = 40$;
2. $0u + v + 0w + 3x + 4y = 240$;
3. $0u + 0v + w + 2x + y = 100$.

The above three constraints correspond to the following augmented matrix.

$$\left[\begin{array}{ccccc|c} 1 & 0 & 0 & 1 & 0 & 40 \\ 0 & 1 & 0 & 3 & 4 & 240 \\ 0 & 0 & 1 & 2 & 1 & 100 \end{array}\right]$$

Simplex and Basic Solutions

In the next lecture, lecture 07, we will discuss the simplex algorithm that solves LP problems with arbitrary numbers of decision variables.

In solving the SMP, the simplex method moves from basic solution to basic solution increasing/decreasing the objective function at each step until there is no larger/smaller value.

Consider an SMP with n decision variables and m constraints. Then after we introduce m slack variables, we have a system of m equations with $m + n$ variables.

References

1. D.P. Maki, M. Thompson. *Finite Mathematics*.