APPLICATIONS FOR MATH 441
(DRAFT–MAY CONTAIN ERRORS)
(DON'T FORGET TO OCCASIONALLY HIT THE "REFRESH" BUTTON ON YOUR BROWSER)

JOEL FRIEDMAN

Contents

1. Introduction 2
2. Terminology for LP (Linear Programming) and Related Optimization Problems 2
2.1. LP Terminology 2
2.2. Linear Programming Without Linear Programming 3
2.3. Integer Programming (IP) 3
2.4. Other of Linear Programs 3
3. Basic LP (Linear Programming) Applications 4
3.1. Resource Allocation 4
3.2. Scheduling with Wait Times 4
3.3. Other Applications 5
3.4. Parametric LP's 5
4. Non-linear (Piecewise Linear, Concave Down) Programming via LP 6
5. Integer Programming with Linear Programming 7
5.1. Example of a Relaxation 7
5.2. IP's That Can Be Solved as Their LP Relaxations 7
5.3. Bipartite Weighted Matching 8
5.4. Total Unimodularity 8
6. Basic IP Applications 9
6.1. Bin Packing and Related Problems 9
6.2. Graph Colouring 10
6.3. Sudoku 10
6.4. Final Exam Scheduling 10
7. Objective Functions That are Non-Linear but Concave Up/Down 10
8. Exercises 10

Date: Last revised Friday 5th October, 2018 at 13:24(get rid of time in final version).
Research supported in part by an NSERC grant.

1. Introduction

This is a list of applications of linear (convex, quadratic, integer, etc.) programming, some of which will be discussed in Math 441. Most of these applications could be the basis of a Math 441 project.

In Section 2 we give some terminology for LP's (linear programs) and IP's (integer programs).

In Section 3 we discuss: resource allocation problems (scarce resources and expensive resouces), scheduling with wait times, and list some other applications. We give some examples of *parametric* linear programming.

In Sections 4 and 5 we discuss problems that are not LP's, but that can be solved as LP's by *relaxing* some of the constraints of the problem. Section 4 involves a piecewise-linear, concave-down objective function, and Section 5 involves an integer program (specifically, a weighted bipartite matching problem). In general, non-linear programming and integer programming cannot be solved using linear programming.

2. Terminology for LP (Linear Programming) and Related Optimization Problems

2.1. **LP Terminology.** An LP (linear program) in *standard form* asks to

$$\text{maximize} \quad \mathbf{c}^{\mathrm{T}}\mathbf{x} = \mathbf{c}\cdot\mathbf{x} = c_1x_1 + \cdots + c_nx_n \tag{1}$$

subject to

$$\mathbf{A}\mathbf{x} \le \mathbf{b} \quad \text{and} \quad \mathbf{x} \ge \mathbf{0}, \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^n$ (i.e., a $n \times 1$ real-valued column vector) of *(decision) variables*, and $\mathbf{A}, \mathbf{b}, \mathbf{c}$ are fixed of respective dimensions $m \times n$, $m \times 1$, and $n \times 1$.

More generally, an LP is to maximize or minimize a linear function of $\mathbf{x}$ subject to linear constraints—equalities and inequalities—in $\mathbf{x}$. Any LP can be written in an equivalent standard form (i.e., as (1) and (2)), and this can be good for teaching purposes but can be problematic in algorithms.

Any $\mathbf{x}$ satisfying (2) is called a *feasible solution* of the linear program, and any feasible $\mathbf{x}$ maximizing the *objective* $\mathbf{c}\cdot\mathbf{x}$ is called an *optimal solution*. The *feasible region* of the LP is the set of all feasible solutions; it is a convex subset of $\mathbb{R}^n$.

The simplex method introduces new *slack variables*

$$\mathbf{x}_{\text{slack}} = \mathbf{b} - \mathbf{A}\mathbf{x}_{\text{dec}} \tag{3}$$

where $\mathbf{x}_{\text{dec}}$ are the original $\mathbf{x}$ decision variables. If $\mathbf{b} \ge 0$, then $\mathbf{x}_{\text{dec}} = \mathbf{0}$ is a feasible solution and (3) is the initial (first) *dictionary* of the simplex method; if not, then the simplex method uses *two phases*, where the first phase determines whether or not there is a feasible solution (equivalently, a feasible dictionary). The textbooks by Chvatal and Vanderbei teach the dictionary notation for the simplex method; other textbooks use *tableau* notation.

An LP may be *infeasible*, meaning that there are no feasible solutions; a feasible LP may have an optimal solution; a feasible LP without an optimal solution is *unbounded*, meaning that there are feasible solutions whose objective attains arbitrarily large values.

The matrix $\mathbf{A}$ is often called the *coefficient matrix*; in certain applications $\mathbf{A}$ satisfies some nice properties, such as (1) its entries, a_{ij}, may be all non-negative,

or (2) **A** may be *totally unimodular* (see below). Some of these special properties have desirable consequences for the simplex method and/or the solution of the LP.

2.2. **Linear Programming Without Linear Programming.** There is a powerful fact about linear programs that is a consequence of the simplex method; this fact does not require you to run the simplex method, but to merely consider how your LP is set up: if you have an LP (linear program) in standard form with 100 variables and 5 constraints, then any optimal solution obtained with the simplex method will have at least 95 (decision) variables equal to zero. More generally, if an LP in standard form has n variables and m constraints with $n \geq m$, and if the LP is feasible and bounded (which is typical in applications), then the LP has an optimal solution with at least $n - m$ decision variables equal to zero; furthermore the simplex method always produces such a solution.

From this you can show that for a 100×5 matrix game, where Alice has 100 pure strategies and Betty has 5, then Alice has an optimal solution for the game "Alice announces a mixed strategy" where Alice plays only 5 of her 100 possible strategies. If someone tells you which 5 of these strategies Alice plays in optimality, then it would be easy to solve the linear program.

Unfortunately, this means that if you have 5 constraints on a diet problem involving 100 foods, then there is an optimal solution using only 5 foods (which is not particularly realistic).

2.3. **Integer Programming (IP).** We use $\mathbb{Z}$ to denote the integers $\{0, \pm 1, \pm 2, \ldots\}$, and $\mathbb{Z}^n$ to denote n-dimensional (column) vectors whose components are integers.

An example of an *integer program (IP)* is an optimization problem of the form (1) and (2) which also requires $\mathbf{x} \in \mathbb{Z}^n$, i.e., that the components of $\mathbf{x}$ are integers. More generally, an IP is any problem to maximize or minimize a linear function of $\mathbf{x} \in \mathbb{Z}^n$ subject to linear constraints.

The *feasible region* of an IP is the set of $\mathbf{x} \in \mathbb{Z}^n$ satisfying the linear constraints.

Algorithms to solve IP's typically take much more time than algorithms to solve LP's of the same size. In fact, solving an IP is *NP-complete*[1], whereas there are polynomial time algorithms[2] to solve linear programming.

2.4. **Other of Linear Programs.** Many optimization problems involve non-linear functions, and some require the decision variables $\mathbf{x}$ to have *integer* values. This class of problems are known as *mathematical programs* or *non-linear programs*. Sometimes one is allowed to add *Boolean expressions*.

It is often hopeless to solve such general problems without additional assumptions on the objective function and constraints.

Some special cases where one can hope to solve relatively small problems (relative to the size of LP problems) include *convex programming* and, in particular *quadratic programming*. The *Markowitz model*, which we will cover in class, is a quadratic program. In this article we will give some examples; in this class we will likely cover an additional example—the *Markowitz Model*—which is a quadratic program.

[1]This is covered in CPSC 421 and MATH 523.

[2]Some algorithms used in practice, such as the simplex method, are not currently known to run in polynomial time but appear to be fast.

3. Basic LP (Linear Programming) Applications

In Math 340 you have seen a number of standard applications of linear programming. We'll briefly review these, and then introduce some applications that may be new.

3.1. **Resource Allocation.** These problems are a basic type of LP application (see Chapter 1 of either Chvatal's or Vanderbei's textbook).

Scarce (i.e., limited) resources: Example: building furniture: you have a given amount of *resources* (e.g., wood, labour or time, other materials). You can use these to build *products* (e.g., tables, chairs, plant boxes). Your *utility* is the money you make from each product.

Expensive resources: Example: diet problems: you have dietary requirements (upper and lower bounds), and you want to minimize the cost.

3.2. **Scheduling with Wait Times.** You wish have 6 tasks to perform; these tasks take very little time (e.g., preheating an oven as part of baking cookies), but you have the following scheduling constraints:

(1) task 2 must be performed at least 30 minutes after task 1;
(2) task 3 must be done at least 26 minutes after task 1;
(3) and a number of similar constraints.

We organize all these constraints by saying that there is a function $\mathrm{Wait}(i,j)$, defined for certain pairs (i,j) with $1 \le i < j \le 6$, such that task j must be performed at least $\mathrm{Wait}(i,j)$ minutes after task i; the function $\mathrm{Wait}(i,j)$ is listed in the following table:

i	j	$\mathrm{Wait}(i,j)$
1	2	30
1	3	26
2	4	18
2	5	40
3	4	26
3	5	45
4	6	13
5	6	11

What is the shortest amount of time it takes to complete all these tasks?

Let x_i denotes the time that we permform task i; and let us introduce variables x_0, x_7 that represent the first time we perform a task and the last time we perform a task. Then the above constraints yield the LP:

$$\text{Minimize } \ x_7 - x_0 \ \text{ subject to}$$

$$x_0 \le x_1,\ x_0 \le x_2,\ \dots, x_0 \le x_6,$$

$$x_1 \le x_7,\ x_2 \le x_7,\ \dots, x_6 \le x_7,$$

and, based on the above table:

$$x_1 + 30 \le x_2,\ x_1 + 26 \le x_3,\ x_2 + 18 \le x_4,\ \dots\ , x_5 + 11 \le x_6.$$

The dual of this LP is very interesting: it identifies the "critical path of action." (This will be illustrated in the homework.)

3.3. **Other Applications.**

Matrix Games: These are two-player, zero sum games, such as rock-scissors-paper, two person poker, etc.

3.4. **Parametric LP's.** Any LP can be made more interesting when some coefficient(s), constant(s), and/or matrix entries are parameters rather than fixed values.

Example 3.1. Consider the LP:

$$\begin{aligned} \text{Maximize } & 3x_1 + ax_2 \text{ subject to} \\ & x_1 + x_2 \leq 5 \\ & x_1, x_2 \geq 0, \end{aligned}$$

where $a \in \mathbb{R}$ is a parameter. If $a \leq 3$, then $x_1^* = 5$, $x_2^* = 0$ is an optimal solution, and the objective value is 15. If $a \geq 3$, then $x_1^* = 0$, $x_2^* = 5$ is an optimal solution, and the objective value is $5a$. Hence the optimal objective value is the function of a given by

$$\begin{cases} 15 & \text{if } a \leq 3, \text{ and} \\ 5a & \text{if } a \geq 3. \end{cases}$$

Furthermore, something interesting happens (the optimal solution changes) near $a = 3$.

Example 3.2. Consider the LP:

$$\begin{aligned} \text{Maximize } & x_1 + x_2 \text{ subject to} \\ & x_1 + bx_2 \leq 7 \\ & x_1, x_2 \geq 0, \end{aligned}$$

where $b \in \mathbb{R}$ is a parameter; for simplicity let's assume that $b \geq 0$. It is not hard to see that:

(1) if $b \geq 1$, then $x_1^* = 7$, $x_2^* = 0$ is an optimal solution, and the objective value is 7;
(2) if $0 < b \leq 1$, then $x_1^* = 0$, $x_2^* = 7/b$ is an optimal solution, and the objective value is $7/b$;
(3) if $b = 0$, then the LP is unbounded.

Hence the optimal objective value is

$$\begin{cases} 7 & \text{if } b \geq 1, \\ 7/b & \text{if } 0 < b \leq 1, \\ \text{unbounded} & \text{if } b = 0 \end{cases}$$

Furthermore, something interesting happens (the optimal solution changes) near $b = 1$.

These LP's illustrate the fact that if a feasible and bounded LP has two decision variables and one constraint, then there is an optimal solution where one decision variable is zero.

4. Non-linear (Piecewise Linear, Concave Down) Programming via LP

Certain very special types of non-linear programming can be handled with linear programming. One example of this is an LP where the objective to be maximized is a *piecewise linear* function that is *concave down*; this corresponds to resources whose utility have a "diminishing return."

Example 4.1. Say that you can decide to go to some number of "rock concerts" and "trips to mountains," where

(1) each rock concert has a utility of 1; let r denote the number of rock concerts you attend;
(2) the first 10 mountain trips have a utility of 2, and each subsequent mountain trip has a utility of 0.5; let f denote the number of mountain trips up to 10, and let s denote any number of trips past the first ten.

You have a linear objective function

$$z = r + 2f + (0.5)s$$

with linear constraints

$$f \le 10, \qquad r, f, s \ge 0$$

plus you have a *Boolean* (i.e., *logical*) constraint

$$\text{If } s > 0, \text{ then } f = 10;$$

the Boolean constraint is not allowed in an LP. Assume that each rock concert requires a certain amount of money and time, and similarly for each travel to a mountain; then constraints may look like

$$r + 3(f + s) \le 300, \quad 2r + 5(f + s) \le 550.$$

It is crucial that the constrains involve functions of $f + s$, not individual functions of f and s. In this case f and s are interchangeable in the constraints, but f is more valuable to the objective. So the optimal solution of the linear program

$$\begin{gathered}\text{Maximize } \ r + 2f + (0.5)s \ \text{ subject to} \\ f \le 10 \\ r + 3(f + s) \le \text{something}, \\ 2r + 5(f + s) \le \text{something else} \\ r, f, s \ge 0\end{gathered}$$

will automatically satisfy the Boolean constraint

$$\text{If } s > 0, \text{ then } f = 10.$$

Example 4.2. A slight variant of the above situation behaves poorly and can't be handled with LP: consider the situation where the first 10 mountain trips have utility 0.5 and any subsequent ones have utility 2. The linear program

$$\begin{gathered}\text{Maximize } \ r + (0.5)f + 2s \ \text{ subject to} \\ f \le 10 \\ r + 3(f + s) \le \text{something}, \\ 2r + 5(f + s) \le \text{something else} \\ r, f, s \ge 0\end{gathered}$$

will always have $f = 0$ in the optimial solution, since s has a higher utility coefficient than f, and the constraints allow us to "move" any amount of f into the variable s (while maintaining feasibility). Hence the constraint

$$\text{if } s > 0, \text{ then } f = 10$$

is not satisfied by the optimal solution. We will later see that this is indicative of a fundamental difficulty with non-linear programming when the functions don't have good "concavity" (or "convexity") properties.

5. Integer Programming with Linear Programming

Certain IP's (integer programs) can be solved by solving the *LP (linear programming) relaxation*, which is the IP where we allow the decision variables to be real numbers.

5.1. **Example of a Relaxation.** It is easy to see that the "toy" integer program

$$\begin{aligned} \text{Maximize } \ z = 3x_1 + x_2 \ \text{ subject to} \\ x_1 + x_2 &\leq 12.3 \\ x_1 &\leq 5.1 \\ x_1, x_2 &\geq 0 \\ x_1, x_2 &\in \mathbb{Z} \end{aligned}$$

has the solution $x_1^* = 5, x_2^* = 7$ with objective value $z^* = 22$.[3] The LP relaxation is when we allow $x_1, x_2 \in \mathbb{R}$; it is easy to see that the optimal solution of the LP relaxation is $x_1^* = 5.1$, $x_2^* = 7.2$, with objective value $z^* = 22.5$; this objective value is higher than if we restrict x_1, x_2 to be integers.

5.2. **IP's That Can Be Solved as Their LP Relaxations.** If the LP relaxation of an IP has integer values in some optimal solution then this LP optimal solution is also an optimal solution to the IP.

For example, for the "toy" IP

$$\begin{aligned} \text{Maximize } \ 3x_1 + x_2 \ \text{ subject to} \\ x_1 + x_2 &\leq 12 \\ x_1 &\leq 5 \\ x_1, x_2 &\geq 0 \\ x_1, x_2 &\in \mathbb{Z}, \end{aligned}$$

it is easy to see that the LP relaxation has a (unique) optimal solution $x_1^* = 5, x_2^* = 7$. Since this solution $\mathbf{x}^*$ attains the highest value of the objective $3x_1 + x_2$ over all feasible $\mathbf{x} \in \mathbb{R}^2$, it also attains the highest value over all feasible $\mathbf{x} \in \mathbb{Z}^2$.

It turns out that there are important families of IP's such that if you run the Simplex Method on their LP relaxations, the optimal solution produced by the Simplex Method always has integer values.

[3] See Exercise **??**.

5.3. **Bipartite Weighted Matching.** Say that two people, A (Angelina) and B (Brad), each must give a research presentation in one of three time slots (call these slots 1,2,3). Let $x_{A1} = 1$ if A presents in slot 1, and 0 otherwise. Similarly define $x_{A2}, x_{A3}, x_{B1}, x_{B2}, x_{B3}$. Say that we want to maximize

$$U(\mathbf{x}) = 3x_{A1} + 2x_{A2} + x_{A3} + x_{B1} + 3x_{B2} + 2x_{B3}$$

subject to the constraints that each person speaks in at most one time slot:

$$x_{A1} + x_{A2} + x_{A3} \le 1$$
$$x_{B1} + x_{B2} + x_{B3} \le 1$$

and that each time slot is occupied by at most one person:

$$x_{A1} + x_{B1} \le 1$$
$$x_{A2} + x_{B2} \le 1$$
$$x_{A3} + x_{B3} \le 1$$

We want

$$x_{A1}, x_{A2}, x_{A3}, x_{B1}, x_{B2}, x_{B3} \ge 0$$

and (the IP constraint)

$$x_{A1}, x_{A2}, x_{A3}, x_{B1}, x_{B2}, x_{B3} \in \mathbb{Z}$$

(which implies that these variables are either 0 or 1). It turns out that if we relax this IP, allowing the decision variables to be reals, the simplex method will always return an integral value.

This property of the LP relaxation returning an integral solution (provided that we use the Simplex Method) is true for any similar such *weighted bipartite matching* problem.

Note that if we want to maximize the utility function

$$U(\mathbf{x}) = x_{A1} + x_{A2} + x_{A3} + x_{B1} + x_{B2} + x_{B3}$$

where all "weights" equal 1, then there are fractional optimal solutions such as

$$x_{A1} = x_{A2} = x_{A3} = x_{B1} = x_{B2} = x_{B3} = 1/3.$$

However, the simplex method will always produce an optimal solution where four of the decision variables are 0 and two are 1. Other methods, such as *interior point methods*, are not guaranteed to return an integral optimal solution.

A weighted bipartite matching problem can be viewed as a special case of a class of problems called *network flows (with integer data)*; these network flows also have the property that the simplex method always produces integer-valued optimal solutions.

5.4. **Total Unimodularity.** One reason (or proof) that bipartite matching (and network flow problems with integer data) produce integer-valued optimal solutions involves *unimodularity*. Here we explain the rough idea.

A square matrix with integer entires is *unimodular* if its determinant is either -1 or 1; it is known that the inverse of such a matrix has all its entries integers. Any shape matrix is *totally unimodular* if each of its square submatrices has determinant either $-1, 0, 1$. If $\mathbf{A}$ in (2) is totally unimodular and the coefficients of $\mathbf{b}$ are all integers, then any optimal solution found by the simplex method is necessarily integer-valued. Let us briefly explain why.

The formulas for the simplex method (useful in the *revised simplex method*) show that the general form of a dictionary has the basic variables, $\mathbf{x}_{\text{basic}}$ given as

$$\mathbf{x}_{\text{basic}} = \mathbf{B}^{-1}(\mathbf{b} - \mathbf{A}_{\text{nb}}\mathbf{x}_{\text{nb}}),$$

where $\mathbf{B}$, necessarily invertible, is the basic part of the combined coefficient matrix (i.e., $[I|A]$), $\mathbf{A}_{\text{nb}}$ the non-basic part, $\mathbf{x}_{\text{nb}}$ are the non-basic variables, and $\mathbf{b}$ is the vector of constants of the LP. By rearranging the order of the rows and columns of $\mathbf{B}$, we see that $\mathbf{B}$ is an identity matrix where the bottom right square replaced with a submatrix of A; it follows that $\mathbf{B}$ has determinant ± 1 (it can't be 0 since $\mathbf{B}$ is invertible), and hence $\mathbf{B}^{-1}$ has integer entries, and so if $\mathbf{b}$ is integer-valued then the corresponding *basic feasible solution* in the above dictionary will have integer values.

Once can prove that the matrix $\mathbf{A}$ involved in bipartite matching problems is totally unimodular. Since the vector of constants $\mathbf{b}$ consists entirely of 1's, it follows that the simplex method run on these problems always returns an integer-valued optimal solution. With more work one can prove that the simplex method always returns an optimal solution that is a matching.

6. Basic IP Applications

6.1. **Bin Packing and Related Problems.** A lot of problems are forms of "bin packing." Here is an example of "packing into two bins."

Example 6.1. You have a team of two people, and your project involves 6 tasks, each of which has to be done by one person alone; the tasks will take either person the following amount of time in hours: 5,4,2,6,8,9. You want the maximum number of hours of each team member to be as small as possible. Your problem can be stated as

$$\begin{aligned} &\text{Minimize } t \text{ subject to} \\ 5x_1 + 4x_2 + 2x_3 + 6x_4 + 8x_5 + 9x_6 &\le t \\ 5(1-x_1) + 4(1-x_2) + 2(1-x_3) + 6(1-x_4) + 8(1-x_5) + 9(1-x_6) &\le t \\ 0 \le x_1, \ldots, x_6 &\le 1 \\ x_1, \ldots, x_6 &\in \mathbb{Z}, \end{aligned}$$

where for each $i = 1, \ldots, 6$, $x_i = 1$ if the first person does task i, and $x_i = 0$ otherwise.

One good heuristic algorithm for the above problem is to assign the tasks from longest to shortest, giving the longest remaining task to the one with the smallest bin sum: e.g., in the above problem, the tasks are of length 9,8,6,5,4,2, so you assign 9 to person A, the 8 and 6 to person B, then 5 to person A, then 4 to either person A or B (since they are tied before you assign this task), and then 2 to the other person.

For most heuristic algorithms there are situations that foil them. In the above algorithm, if the tasks are of length 25,15,10,10,10,10, then the above heuristic assigns 25 to person A, 15 to person B, ultimately the tasks are divided 45,35; the optimal packing is to assign one person the tasks 25 and 15, and the rest of the tasks to the other person.

Bin packing is a big subject, and there are many good heuristics in practice.

6.2. **Graph Colouring.** [See class notes, in the notebook starting September 12.]

6.3. **Sudoku.** Sudoku can be viewed as a graph colouring problem, but there is a better way to set up the IP. Sudoku is a good example of a problem that can be set up as an IP in two different ways, with one way being shorter/better than another.

[See class notes, in the notebook starting September 12, and homework #2.]

6.4. **Final Exam Scheduling.** [See sample Research Proposal, discussed in class and posted on the course website.]

7. Objective Functions That are Non-Linear but Concave Up/Down

There is an area of optimization know as *convex programming*, with *quadratic (convex) programming* as a special case. This was briefly discussed in class on September 26, 2018 (see the notebook starting on that date).

The bottom line is that if $f = f(\mathbf{x})$ is a function of n-variables that you want to maximize over a convex region R, then if f is concave-down (i.e., $-f$ is convex) then any local maximum of f on R is a global maximum on R.

This will be discussed in more detail sometime in October, and is discussed in the article on non-linear programming posted on the course website.

8. Exercises

In all these problems you must **justify your answer** unless the problem states otherwise; you will not be given any credit for stating the correct answer without a **written justification** that your answer is correct.

(1) Consider the LP (in standard form): maximize x_1+x_2, subject to $x_1+x_2 \le 3$ and $x_1, x_2 \ge 0$.
   (a) Given an optimal solution where x_1, x_2 are both positive, and **justify your answer** (i.e., explain why your solution is feasible and optimal).
   (b) How many basic and non-basic variables does the initial dictionary (or tableau) have?
   (c) Will the simplex method ever return an optimal solution with both x_1, x_2 positive?

(2) [THIS EXERCISE IS IN PROGRESS] Say that you allow yourself at most 12 hours of TV per day, which consists of
   (a) x_1 hours of "The Expanse" reruns,
   (b) x_2 hours of "The X-Files" reruns,
   (c) x_3 hours of "The Walking Dead" reruns,
   (d) x_4 hours of other documentary programs,

   which gives you a utility of

$$U(\mathbf{x}) = 10x_1 + 9x_2 + 5x_3 + 2x_4 \ .$$

   This yields the following LP:

$$\max\ 10x_1 + 9x_2 + 5x_3 + 2x_4 \quad \text{s.t. } x_1 + x_2 + x_3 + x_4 \le 12, \quad x_1, \ldots, x_4 \ge 0.$$

   Answer the following questions:
   (a) What is the optimal solution of the above linear program? **Justify your answer** (you can appeal to the Simplex Method, but I don't recommend it).

(b) Say that you can watch at most 7 hours of The X-Files, and hence add the constraint $x_2 \leq 7$. What is the optimal solution of the above LP with this added constraint? **Justify your answer**.
(c) Say that you can watch at most 7 hours of The X-Files, and at most 3 of The Expanse. What is the optimal solution of the above LP with this added constraint? **Justify your answer**.

(3) Consider the following "toy" LP, involving a real parameter $A \in \mathbb{R}$:

$$\text{maximize } 10x_1 + 9x_2 + Ax_3 + 2x_4$$

subject to

$$\begin{aligned} x_1 + x_2 + x_3 + x_4 &\leq 12, \\ x_1 &\leq 7, \\ x_2 &\leq 3, \\ x_4 &\leq 1, \\ x_1, \ldots, x_4 &\geq 0 \end{aligned}$$

(there is no explicit upper bound on x_3). Describe the optimal solutions as A varies over $\mathbb{R}$ (this includes negative values of A). Justify your answer; no credit is given for simply stating the answer without justification; you may use the Simplex Method to justify your answer, but I don't recommend this.

(4) Consider the following variant of Exercise 2: $x_1, \ldots, x_4$ are the same, but your utility function involves a parameter $B \in \mathbb{R}$ and is given as

$$U(\mathbf{x}) = 10x_1 + 9x_2 + 5x_3 + 2x_4 + B(12 - x_1 - x_2 - x_3 - x_4),$$

which reflects an additional utility of B per each hour of TV that you don't watch. The constraints are

$$\begin{aligned} x_1 + x_2 + x_3 + x_4 &\leq 12, \\ x_1 &\leq 7, \\ x_2 &\leq 3, \\ x_1, \ldots, x_4 &\geq 0 \ . \end{aligned}$$

Describe the optimal solutions for all values of $B \in \mathbb{R}$.

(5) [THIS EXERCISE IS IN PROGRESS] Your life consists of (1) work, and (2) non-work; you are free to decide how many hours you work per day (on average); say that you work x hours per day (on average), and—for simplicity—that your non-work hours per day are $24 - x$.

The *utility* you get from the amount of dollars (Canadian dollars) per day is the following function, $f = f(d)$:
(a) you must earn at least \$80 per day (on a daily average);
(b) each dollar (or fraction thereof) between 80 and 120 is worth 10 units of utility per dollar;
(c) each dollar between between 120 and 160 is worth 7 units of utility;
(d) each dollar between between 160 and 320 is worth 5 units of utility;
(e) each dollar between between 320 and 640 is worth 3 units of utility;
(f) each dollar between between 640 and 3200 is worth 1 units of utility;
(g) each dollar after 3200 is worth 0.1 units of utility.

For example, if you earn $d = 200$ dollars per day, your utility is

$$f(200) = (120 - 80) \times 10 + (160 - 120) \times 7 + (200 - 160) \times 5 \ .$$

Your LP (linear program) has two parameters, A, B, and your overall utility is

$$U(x) = A(24 - x) + f(Bx),$$

where A is a measure of utility per hour of non-work, B is how many dollars you earn per hour (and therefore $f(Bx)$ is f applied to Bx, the number of dollars you earn per average day).

Department of Computer Science, University of British Columbia, Vancouver, BC V6T 1Z4, CANADA, and Department of Mathematics, University of British Columbia, Vancouver, BC V6T 1Z2, CANADA.

E-mail address: jf@cs.ubc.ca or jf@math.ubc.ca

URL: http://www.math.ubc.ca/~jf