Operations Research Primer

Berk Orbay

Table of contents

1 Introduction **3**

I Linear Programming **4**

Mathematical Model 5

Parts of the Model 5

Indices 6

2 Giapetto Example **7**

2.1 Problem Description 7

2.2 Model Building Steps 7

2.3 Mathematical Model 8

2.3.1 Decision Variables 8

2.3.2 Model 8

2.3.3 Constraints 9

3 Markowitz Portfolio Example **10**

3.1 Problem Description 10

3.2 Model Building Steps 10

3.3 Mathematical Model 10

3.3.1 Decision Variables 10

3.3.2 Parameters 11

3.3.3 Model 11

3.3.4 Constraints 11

4 Diet Example **12**

4.1 Problem Description 12

4.2 Model Building Steps 12

4.3 Mathematical Model 13

4.3.1 Decision Variables 13

4.3.2 Parameters 13

4.3.3 Model 13

4.3.4 Constraints 13

II Mixed Integer (Linear) Programming **14**
Mathematical Model 15

5 Knapsack Example **16**
5.1 Problem Description 16
5.2 Model Building Steps 16
5.3 Mathematical Model 17
5.3.1 Decision Variables 17
5.3.2 Parameters 17
5.3.3 Model 17
5.3.4 Constraints 17

6 Traveling Salesman Example **18**
6.1 Problem Description 18
6.2 Model Building Steps 18
6.3 Mathematical Model 19
6.3.1 Decision Variables 19
6.3.2 Parameters 19
6.3.3 Model 19
6.3.4 Constraints 19

1 Introduction

OR Primer is a collection of resources to teach fundamentals of computational optimization regarding Linear Programming and Mixed Integer (Linear) Programming.

This collection targets those who want to get a jumpstart without any technical and theoretical details, except the absolutely required fundamental terms. Therefore, there will be lots of examples.

Primary focus of OR Primer is to give the reader the ability to discern if a "business" problem can be converted to an OR (LP or MILP) problem. In essence "yes" or "no" depends on the answers of the following questions: "Is there a decision to be made? (*i.e. Decision Variables*)", "Are there limitations and requirements? (*i.e. Constraints*)", "What is the ultimate aim? (*i.e. Objective Function*)" and "Can we describe the problem in linear terms? (*i.e. Linearity*)".

At the second stage, we will focus on transferring the "business" problem from verbal to mathematical form. Reader is expected to get a sense of how to write a mathematical model in a standard way.

Finally, we will learn how to describe the model in code. We will learn about Algebraic Modelling Languages (AML) and solvers. Our focus will be on scripting languages such as R, Python and Julia.

Optionally, we will discuss theory. Topics such as simplex, duality, interior points etc. will be briefly explained but it is up to the reader to really dive in the theory or if she is just happy with the coding.

There will be lots of external resources. Feel free to add them on Discussions.

Part I

Linear Programming

Linear programming (LP) is the fundamental modeling method of Operations Research. But briefly an LP should adhere to the following rules.

- **Neither** constraints **nor** the objective function may contain non-linear terms.
- Decision variables are all **continuous**. They **may not** be *binary* or *integer*.
- Decision variables can be either non-negative ($x \geq 0$) or unrestricted (**urs**).

Mathematical Model

A mathematical representation of LP model is provided below.

$$\min c^{\mathsf{T}} x \tag{1.1}$$
$$Ax = b \tag{1.2}$$
$$x \geq 0 \tag{1.3}$$
$$\tag{1.4}$$

Parts of the Model

An LP model requires the following object types to be complete.

- **Decision Variables (DV)**: Decision variables are the objects which the algorithm (i.e. solver) decides its value. A combination of a decision variable value set is a **solution**. In LP it is not possible to define a DV in non-linear (e.g., x^2) terms or in interaction with other DVs (e.g., xy). In the model, elements of x are decision variables. x is an n-sized vector.
- **Coefficients and constants**: It is possible to add pre-defined constants as coefficients to decision variables or by themselves. In the model, elements of A, b and c are constants and coefficients. A is an m x n matrix, b is an m-sized vector and c is an n-sized vector.
- **Constraints**: Constraints are the rules which the decision variable values should satisfy in order to be a valid (i.e. **feasible**) solution. In the model, $Ax = b$ system of equations and non-negativity terms ($x \geq 0$) are the constraints.
- **Objective Function**: Objective function defines the direction (either minimization or maximization) and the evaluation formula of the solution quality. In the model, the term $\min c^{\mathsf{T}} x$ is the objective function and the solver will try to minimize $c^{\mathsf{T}} x$.

Indices

There are also **indices** which we will use to define elements in decision variables, coefficients, constants and constraints. For instance x_i is the i th element of the decision variable vector and $A_{i,j}$ is the (i,j)th element of the coefficient matrix. Let's rewrite the model.

$$\min \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\sum_{j=1}^{n} A_{i,j} x_j = b_i \ \forall_{i \in 1..m} \tag{2}$$

$$x_j \geq 0 \ \forall_{j \in 1..n} \tag{3}$$

$$\tag{1.5}$$

These parts may also be multi-dimensional. For instance $x_{i,j,k,t}$ is possible.

2 Giapetto Example

Giapetto is the introductory example of Linear Programming. This example is directly taken from Winston's Operations Research (4th Edition).

2.1 Problem Description

"Giapetto's Woodcarving, Inc., manufactures two types of wooden toys: **soldiers** and **trains**.

A soldier sells for **$27** and uses **$10** worth of raw materials. Each soldier that is manufactured increases Giapetto's variable labor and overhead costs by **$14**. A train sells for **$21** and uses **$9** worth of raw materials. Each train built increases Giapetto's variable labor and overhead costs by **$10**.

The manufacture of wooden soldiers and trains requires two types of skilled labor: **carpentry** and **finishing**. A soldier requires **2 hours** of finishing labor and **1 hour** of carpentry labor. A train requires **1 hour** of finishing labor and **1 hour** of carpentry labor.

Each week, Giapetto can acquire all of the needed raw material, but he is only allotted **100 finishing hours** and **80 carpentry hours**. There is an unlimited demand for trains. However, **at most, 40 soldiers are sold** each week.

Giapetto wants to **maximize his weekly profit** (Revenues - Costs). Formulate a mathematical model for Giapetto's situation that can be used to maximize Giapetto's weekly profit."

Let's convert problem statement into a number of model building steps in the next section.

2.2 Model Building Steps

1. Let's calculate the net profit of a soldier and a train, respectively. Sale price of a soldier is **$27**, raw material cost is **$10** and labor/overhead costs are **$14**. So producing a soldier toy yields **$3** of net profit. With the same process a train's net profit is **$2**.
2. Our aim is to maximize our total net profit. Let's denote x_1 as the number of soldiers produced and x_2 as the number of trains produced. Values x_1 and x_2 will be determined by the solver. Therefore they are **decision variables**.

3. So, our total net profit can be defined as $z = 3x_1 + 2x_2$. This is also our **objective function**.
4. For finishing tasks, a soldier requires **2 hours** and a train requires **1 hour** of labor. Finishing labor capacity is **100 hours**. So, its mathematical expression is $2x_1 + x_2 \leq 100$.
5. For carpentry tasks, a soldier requires **1 hour** and a train requires **1 hour** of labor. Carpentry labor capacity is **80 hours**. So, its mathematical expression is $x_1 + x_2 \leq 80$.
6. Demand for soldiers is limited with 40. So, its mathematical expression is $x_1 \leq 40$.
7. Also, it is not possible to sell negative amounts of soldiers or toys (no returns). Therefore both x_1 and x_2 should be greater than zero (non-negativity constraints).

Let's gather all the steps in a single model in the next section.

2.3 Mathematical Model

2.3.1 Decision Variables

- x_1: Number of soldiers to be manufactured.
- x_2: Number of trains to be manufactured.

2.3.2 Model

$$\max z = 3x_1 + 2x_2 \tag{2.1}$$

$$s.t.$$

$$2x_1 + x_2 \leq 100 \tag{2.2}$$

$$x_1 + x_2 \leq 80 \tag{2.3}$$

$$x_1 \leq 40 \tag{2.4}$$

$$x_1, x_2 \geq 0 \tag{2.5}$$

$$\tag{2.6}$$

2.3.3 Constraints

- (6.1) is the objective function to maximize total profits. Each soldier yields **$3** profit and each train **$2**.
- (2.2) is the finishing task capacity constraint. Each soldier requires **2 hours** of labor and each train requires **1 hour**. Total capacity for finishing task is **100 hours**.
- (2.3) is the carpentry task capacity constraint. Each soldier requires **1 hour** of labor and each train requires **1 hour**. Total capacity for finishing task is **80 hours**.
- (2.4) is the maximum demand constraint for soldiers. Maximum available demand for soldiers is **40**.
- (4.3) Non-negativity constraint. It is not possible to sell negative amount of each toys (i.e. no backorders, no returns etc. in this case).

3 Markowitz Portfolio Example

Harry Markowitz, one of the pioneers of computational finance and founder of Modern Portfolio Theory, introduced his financial portfolio model in 1952. He defines portfolio items (e.g. stocks) with their risk (usually standard deviation or variance of returns) and reward (mean return). (Trivia: Markowitz is a Nobel Laureate in Economics, 1990)

3.1 Problem Description

There are multiple versions of this problem, but we will model a simple one. Suppose there is an array of investment items with different risk and reward values. We would like form a portfolio to minimize the total risk, given a desired return level. Naturally, risk should be higher with the return.

3.2 Model Building Steps

1. Define the decision variable x_i as the fraction of our portfolio assigned to investment item (i.e. stock) i.
2. Define risk parameter as σ_i and reward parameter as μ_i for item i.
3. Define desired return level parameter as q.
4. Add constraint of the total portolio should add up to 1.
5. Add constraint of minimum return requirement from the portfolio.
6. Add objective function of minimum risk to the portfolio.

3.3 Mathematical Model

3.3.1 Decision Variables

- x_i: Fraction of the budget allocated to investment item i.

3.3.2 Parameters

- σ_j: Risk parameter of item i.
- μ_i: Reward (return) parameter of item i.
- q: Required minimum reward level from the portfolioe.

3.3.3 Model

$$\min z = \sum_i \sigma_i x_i \tag{3.1}$$

$$s.t.$$

$$\sum_i \mu_i x_i \geq q \tag{3.2}$$

$$\sum_i x_i = 1 \tag{3.3}$$

$$x_i \geq 0, \ \forall_i \tag{3.4}$$

$$\tag{3.5}$$

3.3.4 Constraints

- (6.1) is the objective function to minimize total spending.
- (3.2) is the minimum reward requirement constraint.
- (3.3) is the constraint to make sure that sum of all portfolio fractions is 1.
- (4.3) Non-negativity constraint. It is not possible to sell negative amount of each toys (i.e. no backorders, no returns etc. in this case).

4 Diet Example

Diet problem (or Stigler Diet) actually predates Linear Programming. It was coined by George Stigler in 1939. (Trivia: Stigler is a 1982 Economics Nobel Laureate)

4.1 Problem Description

Problem statement is as follows.

"For a moderately active man weighing 154 pounds, how much of each of 77 foods should be eaten on a daily basis so that the man's intake of nine nutrients will be at least equal to the recommended dietary allowances (RDAs) suggested by the National Research Council in 1943, with the cost of the diet being minimal?"

Stigler identifies 9 nutrients (calories, protein, calcium, iron etc.) and their recommended intakes. Also there is a list of nutrients with their nutritional data per dollar. Full data is given in the following (link).

Stigler wants the minimal cost menu which will satisfy nutritional requirements.

4.2 Model Building Steps

1. Define the decision variable x_i as the budget in dollars assigned to food i.
2. Define nutritional requirement parameter as b_j for the nutrient j.
3. Define $a_{i,j}$ as the nutritional level from food i of nutrient j.
4. Define objective function as them minimization of total cost.
5. Define constraints to satisfy nutritional demand by the food items.

4.3 Mathematical Model

4.3.1 Decision Variables

- x_i: Total budget allocated to food item i.

4.3.2 Parameters

- b_j: Nutritional requirement level of nutrient j.
- $a_{i,j}$: Nutritional content of food item i for nutrient j.

4.3.3 Model

$$\min z = \sum_i x_i \tag{4.1}$$

$$s.t.$$

$$\sum_i a_{i,j} x_i \geq b_j, \ \forall_j \tag{4.2}$$

$$x_i \geq 0, \ \forall_i \tag{4.3}$$

$$\tag{4.4}$$

4.3.4 Constraints

- (6.1) is the objective function to minimize total spending.
- (4.2) is the set of constraints to satisfy all nutritional requirements.
- (4.3) Non-negativity constraint. It is not possible to sell negative amount of each toys (i.e. no backorders, no returns etc. in this case).

Part II

Mixed Integer (Linear) Programming

Mixed Integer (Linear) Programming (MIP/MILP) is a type of linear programming which also includes integers as decision variables. So rules are changed slightly.

- **Neither** constraints **nor** the objective function may contain non-linear terms.
- Not all decision variables are necessarily **continuous**. They **may be** *binary* or *integer*.
- Decision variables can be either non-negative ($x \geq 0$) or unrestricted (**urs**).

There are more restrictive versions as well such as Integer Programming (IP) which rules out continuous variables or even Binary Programming (BP) which every decision variable is either 1 or 0.

Mathematical Model

Mathematical representation of MIP/MILP model is almost the same as LP with the exception of the inclusion of integer (and binary) decision variable types.

$$\min c^{\mathsf{T}} x \tag{4.5}$$
$$Ax + By + Cw = b \tag{4.6}$$
$$x \geq 0 \tag{4.7}$$
$$y \in \{0, 1, ...\} \tag{4.8}$$
$$w \in \{0, 1\} \tag{4.9}$$
$$\tag{4.10}$$

Here in our case y represents the set of integer decision variables and w represents binary decision variables. Even though binary decision variables are a subset of integer variables, due to their wide usage it is more convenient to make such a distinction.

5 Knapsack Example

Knapsack problem is one of the fundamental problems of MILP. Knapsack is a representation of a capacity that should be filled with discrete items with various weights and value. The trick is to find the "optimal" set of items to add to knapsack in order to get the maximum value while adhering to capacity constraint.

An entertaining example would be you are given a cart at a shopping mall and you are allowed to fill it for free. Would you fill it with cereal boxes or chocolate?

A more "real life" example which we experience almost all the time would be packing for the airport which you are constrained with the size of your luggage and total weight of the luggage and its contents.

In practice logistics companies deal with this problem every day in order to fill their trucks with packages of various sizes and weights.

5.1 Problem Description

Suppose there is a "space" (e.g. a knapsack) with a single dimensional capacity of W. Also we have a set of items with value and weight properties. We would like to determine which items to include in the knapsack to get the total maximum value from the items in the knapsack.

5.2 Model Building Steps

1. Define the decision variables (x_i) as whether item i is in the knapsack or not. These decision variables are defined as binary.
2. Define value (v_i) and weight (w_i) parameters for each item i.
3. Define a capacity parameter (W) for the knapsack.
4. Define capacity constraint as the total weight of items should not exceed capacity parameter.
5. Define objective function as the maximization of total value of items included in the knapsack.

5.3 Mathematical Model

5.3.1 Decision Variables

- x_i: Whether item i is included in the knapsack or not (binary variable).

5.3.2 Parameters

- W: Capacity of the knapsack.
- v_i: Value of item i.
- w_i: Weight of item i.

5.3.3 Model

$$\max z = \sum_i v_i x_i \tag{5.1}$$

$$s.t.$$

$$\sum_i w_i x_i \leq W \tag{5.2}$$

$$x_i \in \{0, 1\} \tag{5.3}$$

$$\tag{5.4}$$

5.3.4 Constraints

- (6.1) is the objective function to maximize total value for items included in the knapsack.
- (5.2) is the capacity constraint of the knapsack. Each item i has a weight parameter w_i so items included in the knapsack should not weigh more than W.
- (6.8) Binary constraint for decision variables.

6 Traveling Salesman Example

Traveling Salesman (Salesperson) Problem (TSP) is one of the most popular problems in combinatorial optimization. Briefly, suppose you need to visit N cities in a road trip. You will leave home, visit each city once and come back to your home. What is the shortest route?

TSP is an NP-complete problem making it exponentially hard as the problem size increases. There is a trove of literature around TSP and relevant problems such as Vehicle Routing Problem (VRP). Special algorithms and software are designed to solve TSP in addition to solvers. Arguably the most popular solver is Concorde.

6.1 Problem Description

Suppose there are N cities from 1..N. A traveling salesperson starts at city 1, visits each city once and returns "home" (city 1). You are given the (driving) distance between each city. You are expected to find the shortest route possible.

For instance, for a 5 city problem 1-3-2-5-4-1 is a valid route.

6.2 Model Building Steps

1. Define index of cities as i with N cities.
2. Define decision variables $x_{i,j}$ as a binary variable which represents whether city j is visited after city i. Both i and j are part of set of cities.
3. Define (symmetric) distance between each city as $d_{i,j}$.
4. Define the objective function as the minimization of total distance traveled.
5. You cannot visit the same city after itself ($i \neq j$).
6. Add constraints to eliminate subroutes. Each city should be in the source and destination once.

6.3 Mathematical Model

6.3.1 Decision Variables

- $x_{i,j}$: Whether the salesperson used the route from city i to j (binary variable).
- u_i: Subtour elimination variable. It represents the order of the city in the route. ($i \in 2..N$)

6.3.2 Parameters

- $d_{i,j}$: Distance from city i to j (binary variable).

6.3.3 Model

$$\max z = \sum_i \sum_j d_{i,j} x_{i,j} \tag{6.1}$$

$$s.t.$$

$$x_{j,j} = 0, \ \forall_i \tag{6.2}$$

$$\sum_i x_{i,j} = 1, \ \forall_j \tag{6.3}$$

$$\sum_j x_{i,j} = 1, \ \forall_i \tag{6.4}$$

$$u_i - u_j + (N-1)x_{i,j} \leq N-2, \ 2 \leq i \neq j \leq N \tag{6.5}$$

$$1 \leq u_i, \quad 2 \leq i \leq N \tag{6.6}$$

$$u_i \leq N-1, \quad 2 \leq i \leq N \tag{6.7}$$

$$x_{i,j} \in \{0,1\} \tag{6.8}$$

$$u_i \in Z \tag{6.9}$$

$$\tag{6.10}$$

6.3.4 Constraints

- (6.1) is the objective function to maximize total value for items included in the knapsack.
- (6.2) is the constraint to prevent visiting the same city.
- Constraints (6.3) and (6.4) ensure each city is used once as a source ("from") and once as a destination ("to").
- Constraints (6.5), (6.6) and (6.7) ensure we get a full tour as a feasible solution. Since this is an introductory document we will not go into detail but you can read the reference.

- (6.8) Binary constraint for decision variables.
- (6.9) Integer constraint for decision variables.