Artificial Intelligence I

Week 3 Tutorial and Additional Exercises

Optimisation Problems

School of Computer Science

March 10, 2023

In this tutorial...

In this tutorial we will be covering

- Terminology of optimisation problems.
- How to formally represent an optimisation problem.
- Examples of optimisation problems.

Formal optimisation problem

- Recall the canonical form of an optimisation problem:

$$\begin{array}{rll} \text{maximise/minimise} & f(\mathbf{x}) & \\ \text{subject to} & g_i(\mathbf{x}) \leq 0, & i = 1, \ldots, m \\ & h_j(\mathbf{x}) = 0, & j = 1, \ldots, n \end{array}$$

- $\mathbf{x}$ is the vector of *design variables*.
- f is the *objective function*, e.g. the cost or quality of a solution.
- $g_1, \ldots, g_m$ are the *inequality constraints* and $h_1, \ldots, h_n$ are the *equality constraints*.
- In a *multi-objective* optimisation problem, there are more than one objective functions, e.g. $f_1, f_2, \ldots, f_k$.

Formal optimisation problem (continued)

Some more definitions:

- Each value of $\mathbf{x}$ is a *solution* to the optimisation problem.
- The *search space* consists of all possible solutions.
- A solution that satisfies the constraints is called *feasible*. A solution that does not satisfy the constraints is called *infeasible*.

Exercise 1

Consider the following problem:

- A company makes square boxes and triangular boxes. Square boxes take 2 minutes to make and sell for a profit of 4. Triangular boxes take 3 minutes to make and sell for a profit of 5. No two boxes can be created simultaneously. A client wants at least 25 boxes including at least 5 of each type in one hour. What is the best combination of square and triangular boxes to make so that the company makes the most profit from this client?
- Formalize this problem as a canonical optimisation problem, but consider it is ok for the objective to be a function to be maximised instead of minimised. Identify the design variables, the objective function and the constraints.

Exercise 1: Solution

- Let $x_1 \in \mathbb{N}$ be the number of square boxes and $x_2 \in \mathbb{N}$ be the number of triangular boxes. These are the design variables. Write $\mathbf{x} = (x_1, x_2)$. The formal optimisation problem is the following:

$$\begin{aligned} \text{maximise} \quad & 4x_1 + 5x_2 \\ \text{subject to} \quad & 2x_1 + 3x_2 \leq 60 \\ & x_1 \geq 5 \\ & x_2 \geq 5 \\ & x_1 + x_2 \geq 25 \end{aligned}$$

- Let us find the objective function and the constraints so that they follow the canonical formulation.

Exercise 1: Solution (continued)

- Objective function:

$$f(\mathbf{x}) = 4x_1 + 5x_2$$

- Constraints:

$$g_1(\mathbf{x}) = 2x_1 + 3x_2 - 60$$
$$g_2(\mathbf{x}) = 5 - x_1$$
$$g_3(\mathbf{x}) = 5 - x_2$$
$$g_4(\mathbf{x}) = 25 - x_1 - x_2$$

  There are no equality constraints, so no $h_1, h_2, \ldots$ functions.
- With these definitions, the canonical problem can be written

$$\begin{array}{ll} \text{maximise} & f(\mathbf{x}) \\ \text{subject to} & g_i(\mathbf{x}) \leq 0, \qquad i = 1, 2, 3, 4 \end{array}$$

Exercise 2

Consider the following problem:

- A woman makes pins and earrings. Each pin takes 1 hour to make and sells for a profit of 8. Each earring takes 2 hours to make and sells for a profit of 20. She wants to make exactly as many pins as earrings. She has 40 hours and wants to have made at least 20 items, including at least 4 of each item. How many each of pins and earrings should the woman make to maximise her profit?
- Formalize this problem as a canonical optimisation problem, but consider it is ok for the objective to be a function to be maximised instead of minimised. Identify the design variables, the objective function and the constraints.

Exercise 2: Solution

- Let $x_1 \in \mathbb{N}$ be the number of pins and $x_2 \in \mathbb{N}$ be the number of earrings. These are the design variables. Write $\mathbf{x} = (x_1, x_2)$. The formal optimisation problem is the following:

$$\begin{aligned} \text{maximise} \quad & 8x_1 + 20x_2 \\ \text{subject to} \quad & x_1 + 2x_2 \leq 40 \\ & x_1 + x_2 \geq 20 \\ & x_1 \geq 4 \\ & x_2 \geq 4 \\ & x_1 = x_2 \end{aligned}$$

- Let us find the objective function and the constraints so that they follow the canonical formulation.

Exercise 2: Solution (continued)

- Objective function:

$$f(\mathbf{x}) = 8x_1 + 20x_2$$

- Constraints:

$$g_1(\mathbf{x}) = x_1 + 2x_2 - 40$$
$$g_2(\mathbf{x}) = 20 - x_1 - x_2$$
$$g_3(\mathbf{x}) = 4 - x_1$$
$$g_4(\mathbf{x}) = 4 - x_2$$
$$h_1(\mathbf{x}) = x_1 - x_2$$

- With these definitions, the canonical problem can be written

$$\begin{array}{lll} \text{maximise} & f(\mathbf{x}) & \\ \text{subject to} & g_i(\mathbf{x}) \leq 0, & i = 1, 2, 3, 4 \\ & h_j(\mathbf{x}) = 0, & j = 1 \end{array}$$

Exercise 3

Consider the following problem:

- A student works at the library and as a tutor. They must work for at least 5 hours per week at the library and 2 hours per week as a tutor, but they are not allowed to work more than 20 hours per week total. They earn 15 per hour at the library and 20 per hour at tutoring. They prefer working at the library, so they want to have exactly twice as many library hours as tutoring hours. If they need to make exactly 360 in one week, how many hours should they work at each job to minimise their total working hours and meet their preferences?
- Formalize this problem as a canonical optimisation problem, but consider it is ok for the objective to be a function to be maximised instead of minimised. Identify the design variables, the objective function and the constraints.

Exercise 3: Solution

- Let $x_1 \in \mathbb{N}$ be the number of library hours per week and $x_2 \in \mathbb{N}$ be the number tutor hours per week. These are the design variables. Write $\mathbf{x} = (x_1, x_2)$. The formal optimisation problem is the following:

$$\begin{aligned} \text{minimise} \quad & x_1 + x_2 \\ \text{subject to} \quad & x_1 \geq 5 \\ & x_2 \geq 2 \\ & x_1 + x_2 \leq 20 \\ & x_1 = 2x_2 \\ & 15x_1 + 20x_2 = 360 \end{aligned}$$

- Let us find the objective function and the constraints so that they follow the canonical formulation.

Exercise 3: Solution (continued)

- Objective function:

$$f(\mathbf{x}) = x_1 + x_2$$

- Constraints:

$$\begin{aligned} g_1(\mathbf{x}) &= 5 - x_1 \\ g_2(\mathbf{x}) &= 2 - x_2 \\ g_3(\mathbf{x}) &= x_1 + x_2 - 20 \\ h_1(\mathbf{x}) &= x_1 - 2x_2 \\ h_2(\mathbf{x}) &= 15x_1 + 20x_2 - 360 \end{aligned}$$

- With these definitions, the canonical problem can be written

$$\begin{array}{lll} \text{minimise} & f(\mathbf{x}) & \\ \text{subject to} & g_i(\mathbf{x}) \leq 0, & i = 1, 2, 3 \\ & h_j(\mathbf{x}) = 0, & j = 1, 2 \end{array}$$

Exercise 4

Consider the following problem:

- A telescope manufacturer produces three types of telescopes; reflector, refractor and catadioptric. A reflector takes 3 hours to make and is sold for 500. A refractor takes 2 hours to make and is sold for 300. A catadioptric takes 4 hours to make and is sold for 600. The manufacturer wants to earn exactly 2000 profit in one day, while making an equal number of reflectors and refractors and at least 1 of each type. What is the best combination of telescope types to minimise the working hours?
- Formalize this problem as a canonical optimisation problem, but consider it is ok for the objective to be a function to be maximised instead of minimised. Identify the design variables, the objective function and the constraints.

Exercise 4: Solution

- Let $x_1 \in \mathbb{N}$ be the number of reflectors, $x_2 \in \mathbb{N}$ the number of refractors and $x_3 \in \mathbb{N}$ the number of catadioptrics per day. These are the design variables. Write $\mathbf{x} = (x_1, x_2, x_3)$. The formal optimisation problem is the following:

$$\begin{aligned} \text{minimise} \quad & 3x_1 + 2x_2 + 4x_3 \\ \text{subject to} \quad & x_1 \geq 1 \\ & x_2 \geq 1 \\ & x_3 \geq 1 \\ & 3x_1 + 2x_2 + 4x_3 \leq 24 \\ & 500x_1 + 300x_2 + 600x_3 = 2000 \\ & x_1 = x_2 \end{aligned}$$

- Let us find the objective function and the constraints so that they follow the canonical formulation.

Exercise 4: Solution (continued)

- Objective function:

$$f(\mathbf{x}) = 3x_1 + 2x_2 + 4x_3$$

- Constraints:

$$\begin{aligned} g_1(\mathbf{x}) &= 1 - x_1 \\ g_2(\mathbf{x}) &= 1 - x_2 \\ g_3(\mathbf{x}) &= 1 - x_3 \\ g_4(\mathbf{x}) &= 3x_1 + 2x_2 + 4x_3 - 24 \\ h_1(\mathbf{x}) &= 500x_1 + 300x_2 + 600x_3 - 2000 \\ h_2(\mathbf{x}) &= x_1 - x_2 \end{aligned}$$

- With these definitions, the canonical problem can be written

$$\begin{array}{lll} \text{minimise} & f(\mathbf{x}) & \\ \text{subject to} & g_i(\mathbf{x}) \leq 0, & i = 1, 2, 3, 4 \\ & h_j(\mathbf{x}) = 0, & j = 1, 2 \end{array}$$

Exercise 5

The following is the well-known *knapsack problem*:

- You need to load a knapsack with items. The maximum total weight of items that the knapsack can stand is W. You have N items that can be loaded, and each item i has a weight w_i, and a profit p_i. You would like to decide which items to load so as to maximise the total profit of loaded items.
- Formalize this problem as a canonical optimisation problem, but consider it is ok for the objective to be a function to be maximised instead of minimised. Identify the design variables, the objective function and the constraints.
- Hint: Use N design variables, each of which is either 0 or 1.

Exercise 5: Solution

There exist more than one equivalent formulations. Here is one:

- Let $v_1, \ldots, v_N$ be variables where v_i equals 1 if item i is loaded and 0 if not. These are the design variables. Write $\mathbf{v} = (v_1, \ldots, v_N)$. The formal optimisation problem is the following:

$$\begin{array}{ll} \text{maximise} & \sum_{i=1}^{N} v_i p_i \\ \text{subject to} & \sum_{i=1}^{N} v_i w_i \leq W \end{array}$$

- Recall that $\sum_{i=1}^{N} v_i p_i = v_1 p_1 + v_2 p_2 + \cdots + v_N p_N$ and similarly $\sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \cdots + v_N w_N$. They can be computed using algorithm 1.
- Let us find the objective function and the constraints so that they follow the canonical formulation.

Exercise 5: Solution (continued)

- Objective function:

$$f(\mathbf{v}) = \sum_{i=1}^{N} v_i p_i$$

- Constraints:

$$g(\mathbf{v}) = \sum_{i=1}^{N} v_i w_i - W$$

- With these definitions, the canonical problem can be written

$$\begin{array}{ll} \text{maximise} & f(\mathbf{v}) \\ \text{subject to} & g(\mathbf{v}) \leq 0 \end{array}$$

Exercise 5: Solution (continued)

An algorithm to compute the objective function $f(\mathbf{v})$.

Algorithm 1: Compute $\sum_{i=1}^{N} v_i p_i$.

Input: Design variables $\mathbf{v} = (v_1, \ldots, v_N)$; profits $\mathbf{p} = (p_1, \ldots, p_N)$.
Output: *total*.

```
1 total ← 0;
2 for i = 1, ..., N do
3     total ← total + v_i p_i
4 return total
```

Exercise 6

Consider the routing problem formulation learned in Lecture 3.1, and in particular the formulation of the explicit constraint $h(\mathbf{x})$. It is possible to change the formulation of this constraint so that the constraint function will not just check whether the explicit constraint is or is not violated, but will also count the number of non-existent direct paths used by the solution. So, the function would retrieve 0 if all direct paths used by the solution are between neighboring cities, and it would retrieve the number inexistent direct paths otherwise. Create this function.

Exercise 6 (continued)

PS: For now, you may create it either in the form of pseudocode for the purpose of this exercise, or in the form of mathematical equations, as an aid to your learning of the topic. However, later on in assessments, you may be required to understand and write mathematical equations.
PS: You will see later on in the module that counting the number of non-existent direct paths used by the solution can be useful when we use optimisation algorithms to solve optimisation problems with constraints.

Exercise 6: Solution

In pseudocode, the function could be written as:

Algorithm 2: Violate neighbor Constraint

Input: $\mathbf{x}$, a, b, D
Output: num

```
1 num ← 0;
2 if D_{a,x_1} = −1 then
3     num ← num + 1
4 if D_{x_{size(x)},b} = −1 then
5     num ← num + 1
6 for i = 1 to size(x) − 1 do
7     if D_{x_i,x_{i+1}} = −1 then
8         num ← num + 1
9 return num
```

Exercise 6: Solution (continued)

As an equation, it could be written as:

$$h(\mathbf{x}) = \mathbf{1}(D_{a,x_1} = -1) + \mathbf{1}(D_{x_{size(\mathbf{x}),b}} = -1) + \sum_{i=1}^{size(\mathbf{x})-1} \mathbf{1}(D_{x_i,x_{i+1}} = -1)$$

where $\mathbf{1}(a)$ is a function called "indicator" function that returns 1 if the condition a is satisfied and 0 otherwise. You can see this function as "indicating" whether the condition is true or false by outputting 1 or 0, respectively.

Exercise 7

Consider the Hill-Climbing algorithm below:

Algorithm 3: Hill climbing (maximisation)

```
1 x ← generate initial candidate solution randomly;
2 repeat
3 |   u_1, ..., u_N ← generate neighbor solutions (differ from x by a single element);
4 |   u* ← highest quality solution among u_1, ..., u_N;
5 |   if quality(u*) ≤ quality(x) then
6 |   |   return x
7 |   else
8 |   |   x ← u*
9 until Stopping conditions are met;
```

Modify algorithm 3 so that it will minimise, rather than maximise the quality/objective function. Explain your answer.

Exercise 7: Solution

There are different ways to answer this question. Here is one:

Algorithm 4: Hill climbing (minimisation)

```
1 x ← generate initial candidate solution randomly;
2 repeat
3 |   u_1, ..., u_N ← generate neighbor solutions (differ from x by a single element);
4 |   u* ← lowest cost solution among u_1, ..., u_N;
5 |   if cost(u*) ≥ cost(x) then
6 |   |   return x
7 |   else
8 |   |   x ← u*
9 until Stopping conditions are met;
```

Exercise 7: Solution (continued)

First, you may wish to rename your quality function to cost function, to avoid confusion. Then, change line 4 to select the lowest cost neighbor. Change line 5 to use $\geq$ instead of $\leq$. This change is required because lower costs now means better cost. So, a "bad neighbor" is one with equal or higher cost.

Q&A

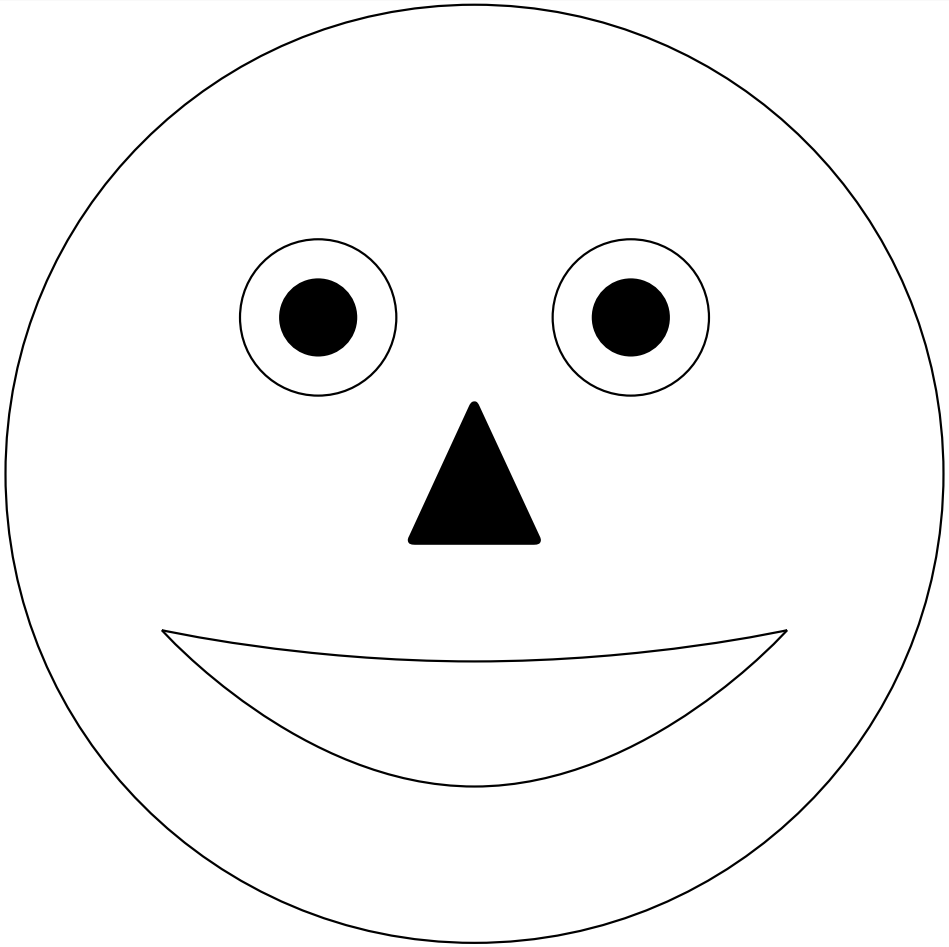
Any questions?

Some closing words...

—True optimisation is the revolutionary contribution of modern research to decision processes.

George Dantzig

Until the next time...



Thank you for your attention!