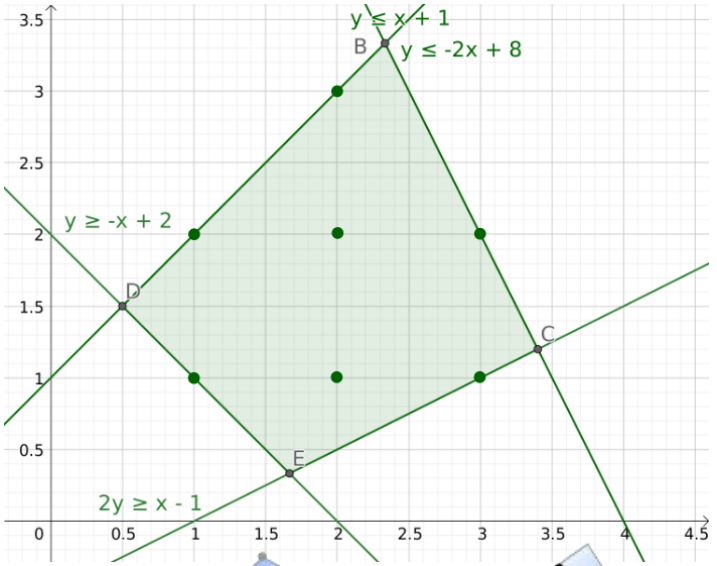
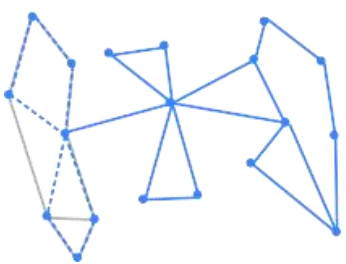
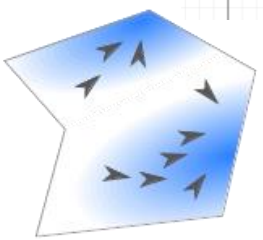
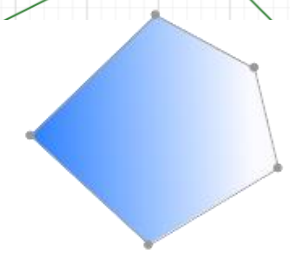
Mixed-Integer Programming 1

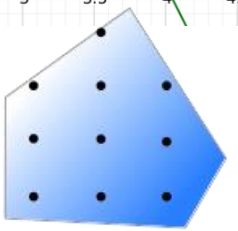
COMP4691 / 8691

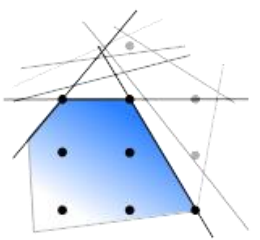


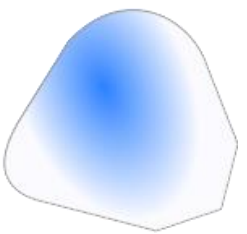












Course Outline

- Linear Programming
- **Mixed-Integer Programming** ← next 3 lectures
- Convex Optimisation
- Decomposition
- Local Search
- Metaheuristics

The content in this section draws inspiration from the textbook:

Optimization over Integers, Dimitris Bertsimas and Robert Weismantel 2005

Mixed-Integer Programming

Integer Linear Programming (ILP)

LP where all variables are restricted to being integers

Mixed-Integer Linear Programming (MILP)

LP where variables are either integers or continuous

IP, MIP

More general forms of the above, but generally used synonymously with the above

We will be focusing on ILP / MILP. Large parts of the theory and solving techniques are the same or similar for both ILP and MILP.

MILP **greatly extends** over LP the **range of problems** we can solve:

e.g., combinatorial problems

This expressiveness comes at a cost: **ILP is NP-hard.**

Knapsack ILP

$$\max_x \; v^\intercal x$$

$$\text{s.t. } w^\intercal x \leq \bar{w}$$

$$x \in \{0,1\}^n$$

Value of each item

Maximum knapsack weight

Weight of each item

Binary decision variables: 1 take item, 0 leave

Knapsack is NP-hard.

See knapsack.py

Knapsack Model

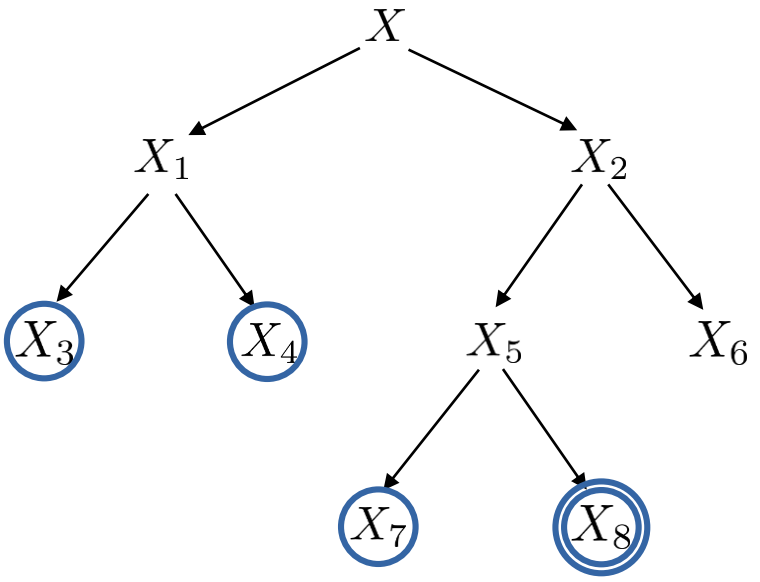
PuLP is able to model MILPs, and call backend solvers, e.g,. **cbc,** glpk, gurobi, cplex.

$$\max_x \ v^\intercal x$$
$$\text{s.t. } w^\intercal x \leq \bar{w}$$
$$x \in \{0, 1\}$$

```
def knapsack (items, max_weight):
     m = pulp. LpProblem(sense = pulp. LpMaximize)

     # Binary variables (cat = 'Continuous' | 'Binary' | 'Integer')
     xs = [Var('x_ {} '. format(i), cat = 'Binary') for i in range (len
(items))]

     m += sum (w * x for x, (w, _) in zip (xs, items)) <= max_weight
     m += sum (v * x for x, (_, v) in zip (xs, items))

     m. solve()
     return (m. status, m. objective. value(), [x. value() for x in xs])

s, obj, xs = knapsack([(3, 4), (2, 5), (6, 6), (3, 5)], 10)
```

MILP Topic Outline

- **Linear relaxation**
  - Complexity
  - Linear relaxation
  - Convex hull
  - Delivery drone example
- **Modelling and solving**
- Branch and bound
- Cutting plane methods
- Duality



Note on Complexity

While ILP is in general NP-hard, it is a worst-case result. Not all hope is lost:

- **Not all problems** we want to model **are NP-hard** just because we have integer variables (e.g., total unimodularity of some ILPs).
- Even if a problem is NP-hard, it doesn't mean we can't **solve specific instances or sub-classes** of the problem **efficiently.**
- We use clever formulations and solvers that can **identify sub-class / instance-specific structure,** to at least **delay the exponential** wall of death.

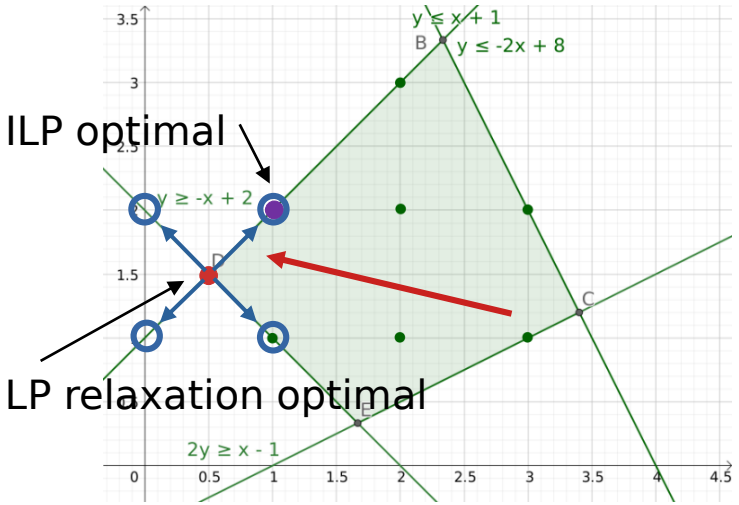
E.g., consider the subset of knapsack problems where all items have the same weight. Can you think of a simple algorithm to efficiently solve them?

Sort based on value, then pick first $\left\lfloor \frac{\bar{w}}{w} \right\rfloor$ of them

Note: the converse is also possible: we can accidentally create an exponential time algorithm (or ILP model) for a polynomial time problem!

Linear Relaxation

Taking an ILP / MILP and **relaxing** the integer requirements produces an LP



$$\min_x \; c^\intercal x \quad \text{s.t. } Ax \leq b, \; x \in \mathbb{Z}^n \longrightarrow \min_x \; c^\intercal x \quad \text{s.t. } Ax \leq b, \; x \in \mathbb{R}^n$$

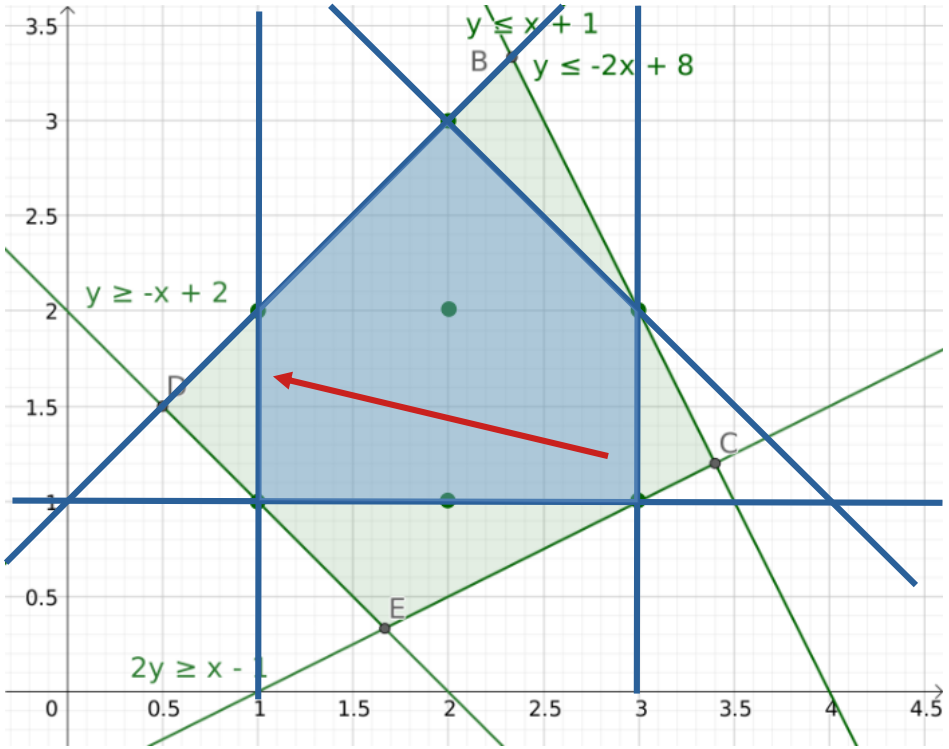
If the optimal LP solution is **feasible for the original ILP,** then we have an **optimal solution to the original problem** (why?).

Unsurprisingly this isn't too common (ILP is NP-hard, LP is P after all).

Rounding the LP solution: might get lucky, but also might be suboptimal or infeasible

Linear Relaxation

In ILP there are many ways to formulate the same problem: through choice of variables and / or constraints.



There are better and worse ways of doing this.

The blue region captures the same set of ILP feasible points, but its linear relaxation will produce an integer feasible (and hence optimal) solution.

We have formed the convex hull of the ILP feasible points. In doing so, the optimal ILP solution will be an extreme point and optimal solution of the linear relaxation.

Convex Hull

$$\min_x \; c^\intercal x \quad \text{s.t. } Ax \leq b, \; x \in \mathbb{Z}^n$$

X	set of ILP feasible points
$\text{conv}(X)$	convex hull of ILP feasible points
L	linear relaxation of ILP

convex polyhedra

We don't explicitly know X (NP-hard to find a single feasible point). It is similarly difficult to obtain and / or efficiently represent $\text{conv}(X)$ (otherwise we could solve all ILP's in polynomial time!).

$$\text{conv}(X) \subseteq L$$

Our formulation determines how **strong** the linear relaxation is. When not equal to the convex hull it it can still provide useful information about the ILP that can be leveraged in solvers (e.g., a lower bound on the ILP objective, and a location for where to look for ILP solutions).

Convex Hull

We want to formulate an ILP so its linear relaxation is as **strong** (close to the convex hull) as **reasonably possible,** i.e. without:

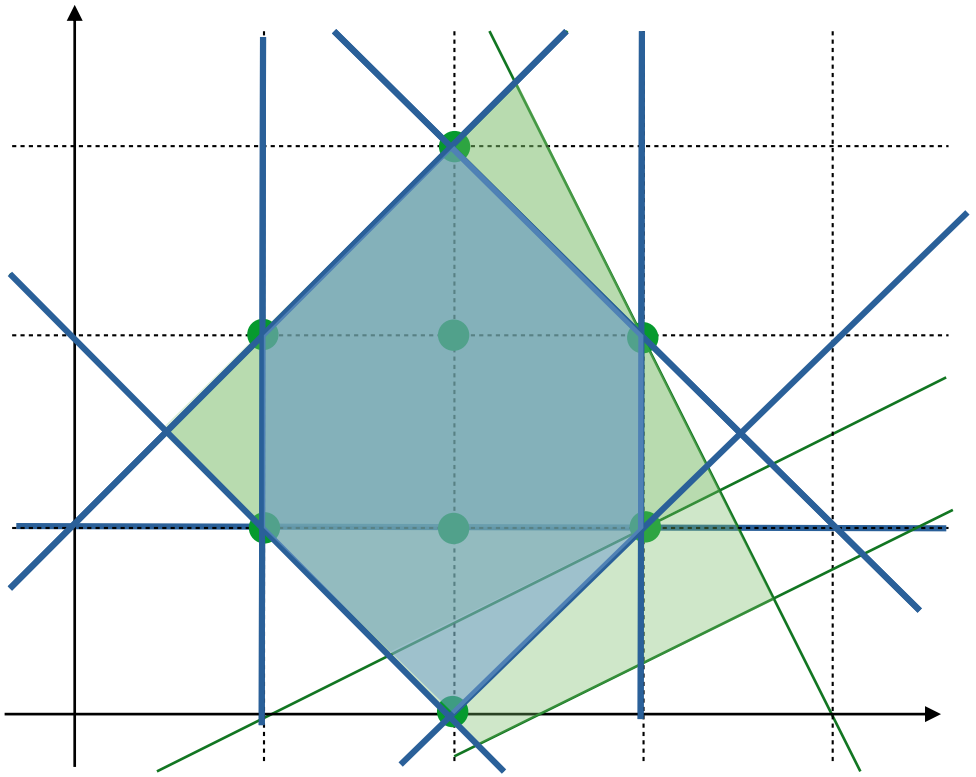
- solving an NP-hard problem in the process, or
- blowing up the formulation with too many constraints

To be useful, we typically want a formulation that is strong for a class of problems, e.g., all knapsack problems, rather than just a narrow instance.

Formulations will typically have to weaken the more general they become.

Unfortunately the **convex hull changes** not just as the class of problem changes, but also **as parameter values change.**

Changing Hull

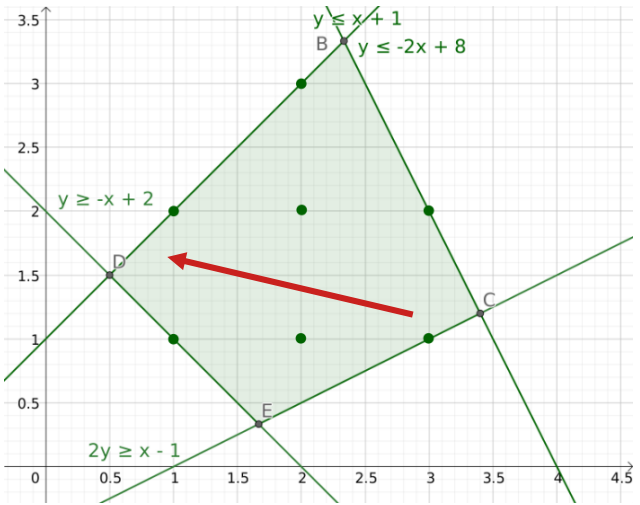


Changing “b” coefficient of one constraint changes convex hull in a way where a new facet / inequality is required.

Solver Strengthening

Good News: we don’t have to do **all** of the work ourselves:

Solvers can employ automated techniques to further strengthen the linear relaxation online with more instance-specific information, focusing the attention on regions where optimal solutions are likely to be, rather than the whole feasible region.



Pigeonhole Principle

We want to write an ILP to allocate pigeons to pigeonholes. Then demonstrate the pigeonhole principle: it is not possible to fit **p** pigeons in **q < p** pigeonholes (while only keeping one pigeon per pigeonhole).

Parameters?

$$p, q \in \mathbb{N}$$

Variables?

$$x_{i,j} \in \{0,1\} \quad \forall i \in \{1,\ldots,p\} \quad \forall j \in \{1,\ldots,q\}$$

Objective?

0

Constraints?

$$\sum_{j=1}^{q} x_{i,j} = 1 \quad \forall i \in \{1,\ldots,p\} \qquad \sum_{i=1}^{p} x_{i,j} \leq 1 \quad \forall j \in \{1,\ldots,q\}$$

Pigeonhole Principle

This seems OK. What about replacing the second constraint as so:

$$\sum_{i=1}^{p} x_{i,j} \leq 1 \quad \forall j \in \{1, \ldots, q\}$$

with

$$x_{i,j} + x_{k,j} \leq 1 \quad \forall i \neq k \in \{1, \ldots, n\} \quad \forall j \in \{1, \ldots, q\}$$

Each pigeon hole can contain at most one of the pigeons.

Are these equivalent?

They encode the same ILP (same set of integer feasible solutions).

Pigeonhole Principle

What accounts for the difference?

The first formulation is **stronger** than the second: the linear relaxation is closer to the convex hull of integer feasible solutions; the linear relaxation feasible region is smaller.

Consider the continuous solution: $x_{i,j} = \frac{1}{q}$

$$\sum_{i=1}^{p} x_{i,j} \leq 1 \implies \frac{p}{q} \leq 1$$

infeasible when p > q

$$x_{i,j} + x_{k,j} \leq 1 \implies \frac{2}{q} \leq 1$$

feasible when p > q (q ≥ 2)

It is unlikely you would have come up with the second formulation for this problem; but that is not always so!

Food Delivery Drone

Example of a **Capacitated Facility Location** problem.

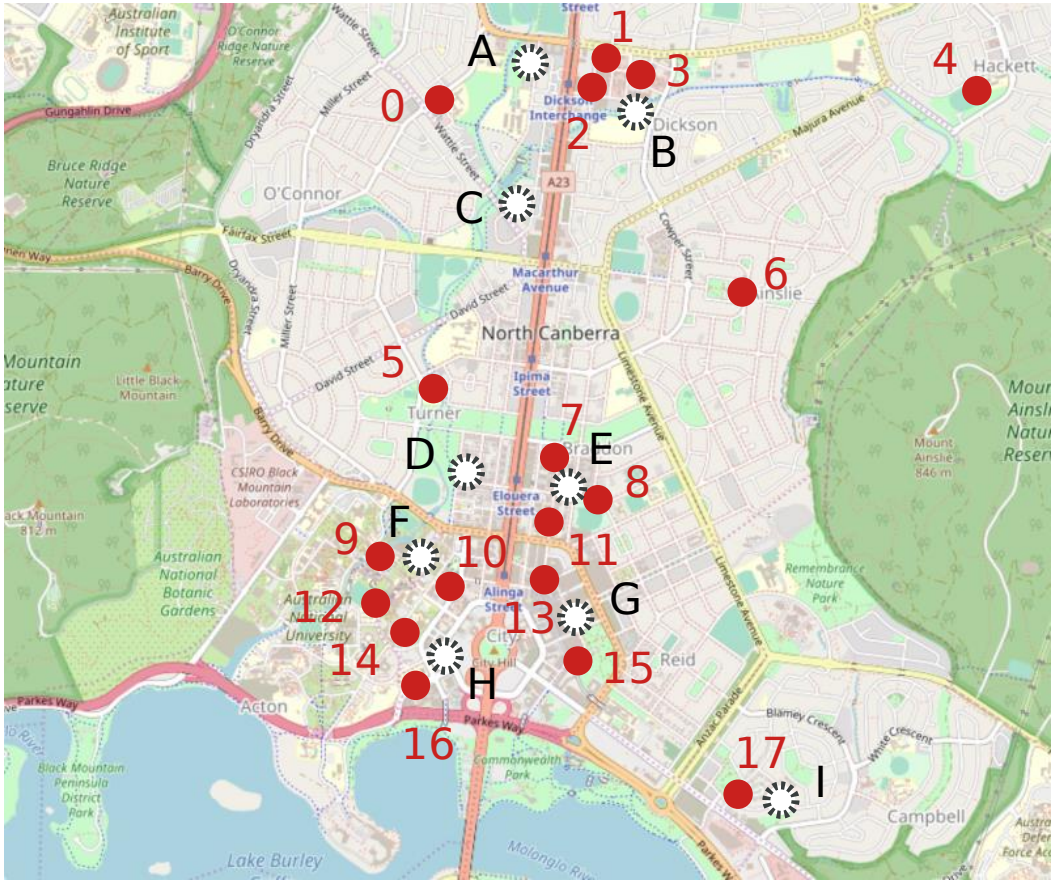
Locate hangers where drones can recharge and rest between jobs.

Should be close to restaurants.

Each hanger has a limited capacity, a cost to build, and a cost to serve each restaurant.

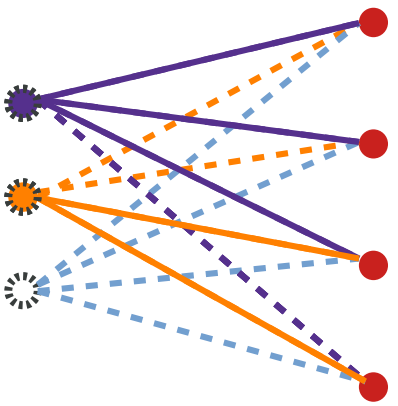
Must supply all restaurant contracted demand.

Food Delivery Drone



- Restaurants
- Hanger Options

Bipartite Graph



Food Delivery Drone

$$\min_{h,d} \sum_{s \in S} p_s h_s + \sum_{s \in S} \sum_{r \in R} p_{s,r} d_{s,r}$$

Cost to build and meet trips

$$\text{s.t.} \sum_{r \in R} d_{s,r} \leq c_s h_s \quad \forall s \in S$$

Capacity of built hangers

$$\sum_{s \in S} d_{s,r} = t_r \quad \forall r \in R$$

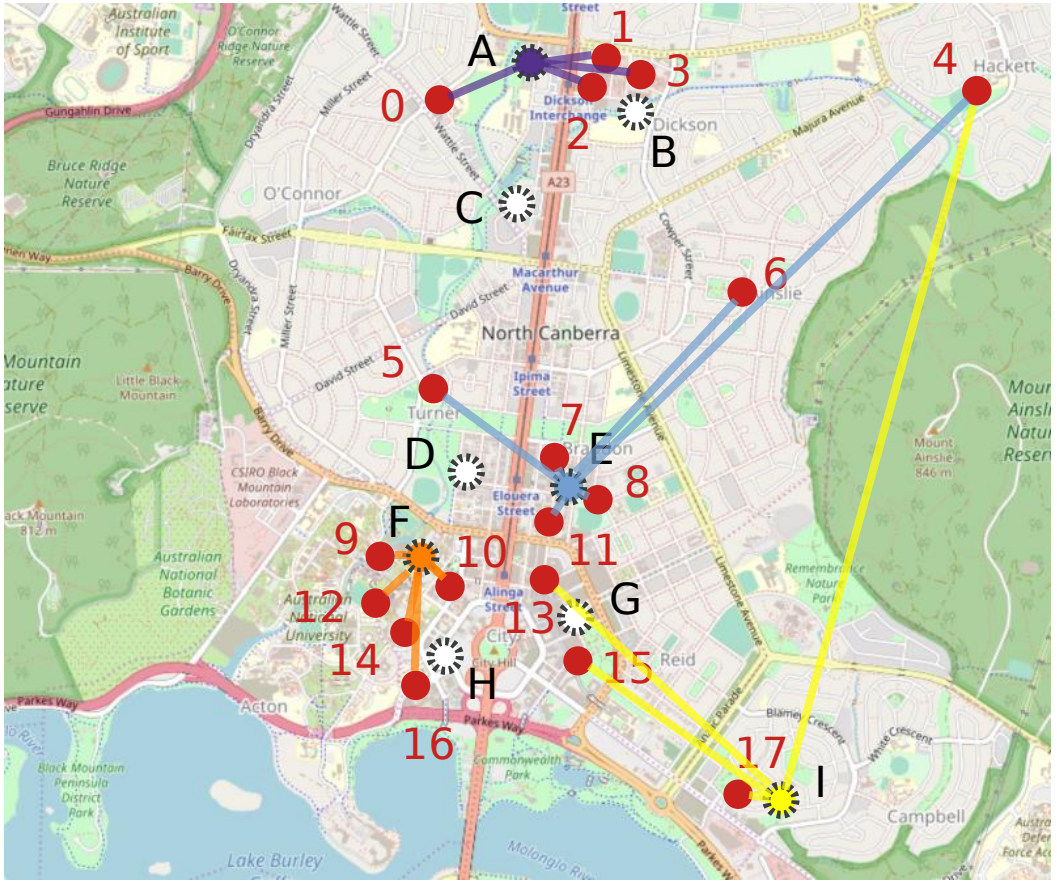
All trips must be met

$$d_{s,r} \geq 0$$

$$h_s \in \{0, 1\}$$

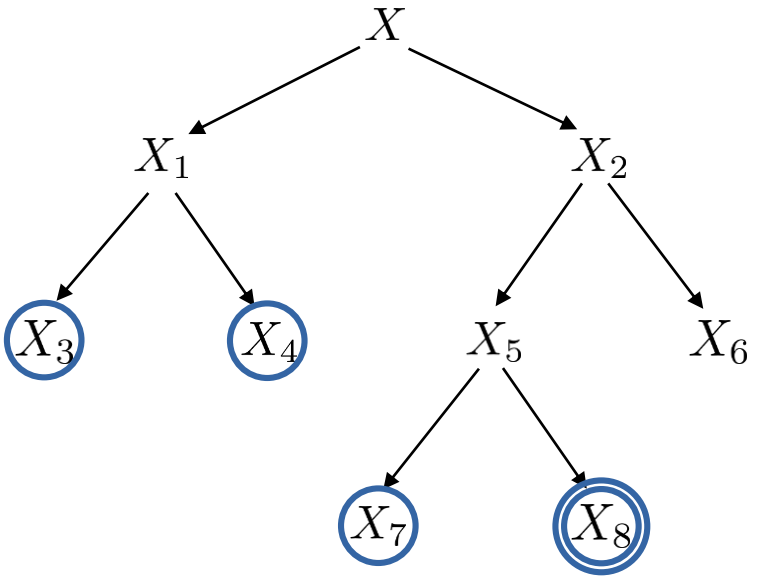
See drone_hangers.py

Food Delivery Drone



MILP Topic Outline

- **Linear relaxation**
- **Modelling and solving**
  - Common MILP constraints
  - Network expansion example
- Branch and bound
- Cutting plane methods
- Duality



Common Constraints

Coming up with ILP constraints can be difficult:

It requires thinking in a very ILP frame of mind that takes time and practice to develop

We will go through some of the common constraint encodings to give an idea of what is possible.

See the following online document for more ideas:

FICO 2009 – MIP Formulations and Linearizations

Choice Constraints

$x_j \in \{0, 1\}, \quad y \in \mathbb{R}$

Binary choice:

At most p items selected:

$$\sum_j x_j \leq p$$

E.g., in knapsack problem:

$$\begin{aligned} \max_x \ & v^\intercal x \\ \text{s.t. } & w^\intercal x \leq \bar{w} \\ & x \in \{0, 1\} \end{aligned}$$

Variable y can take on one value from set $S = \{s_1, s_2, \ldots, s_k\}$:

$$y = \sum_j s_j x_j \qquad \sum_j x_j = 1$$

Logical Constraints

$$x, y, z \in \{0, 1\}$$

$x \implies y$	$y \geq x$
$z = x \vee y$	$z \geq x, \quad z \geq y, \quad z \leq x + y$
$z = x \wedge y$	$z \leq x, \quad z \leq y, \quad z \geq x + y - 1$
$z = \neg x$	$z = 1 - x$

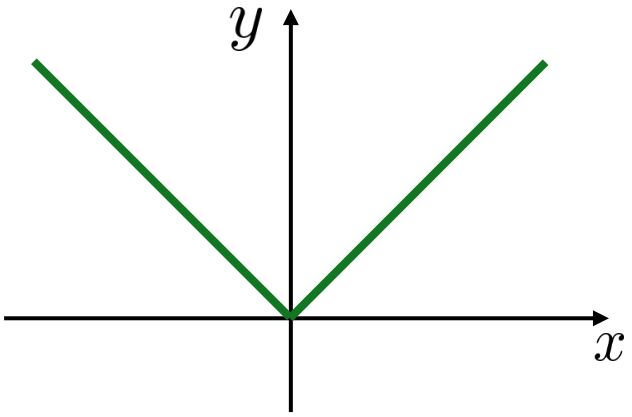
Nonlinear Functions

$x \in [\underline{x}, \overline{x}], \quad y \in \mathbb{R}$

$y = |x|$

auxiliary variable, only exists to formulate this constraint (vs decision variable)

$u \in \{0, 1\}$



$$0 \leq y - x \leq -2\underline{x}u \longrightarrow$$

$$0 \leq y + x \leq 2\overline{x}(1-u) \longrightarrow$$

$u = 0$

$y = x$

$0 \leq y + x \leq 2\overline{x}$

$\downarrow$

$0 \leq 2x \leq 2\overline{x}$

$u = 1$

$0 \leq y - x \leq -2\underline{x}$

$y = -x$

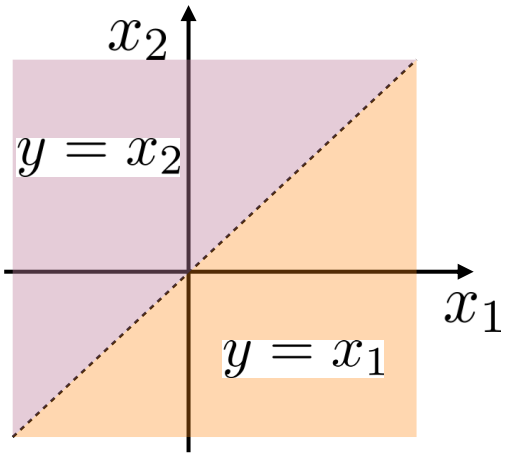
$\downarrow$

$0 \leq -2x \leq -2\underline{x}$

Nonlinear Functions

$x_j \in [\underline{x}_j, \overline{x}_j], \quad y \in \mathbb{R}$

$y = \max(x_1, x_2, \ldots, x_n)$



$u_j \in \{0, 1\}$

$y \geq x_j$

$y = x_j$ if $u_j = 1$

$y \leq x_j + (\max_i(\overline{x}_i) - \underline{x}_j)(1 - u_j)$ ⟶ $y \leq x_j$ for $j|u_j = 1$

$\sum_j u_j = 1$

$\forall j|u_j = 0$

$y \leq x_j + \max_i(\overline{x}_i) - \underline{x}_j$

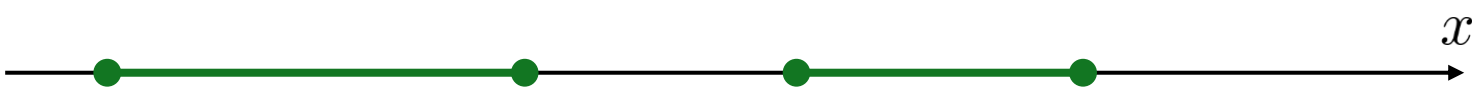
$x_j = \underline{x}_j$

$y \leq \max_i(\overline{x}_i)$

$x_j = \overline{x}_j$

$y \leq \overline{x}_j - \underline{x}_j + \max_i(\overline{x}_i)$

Disjunctive Constraints



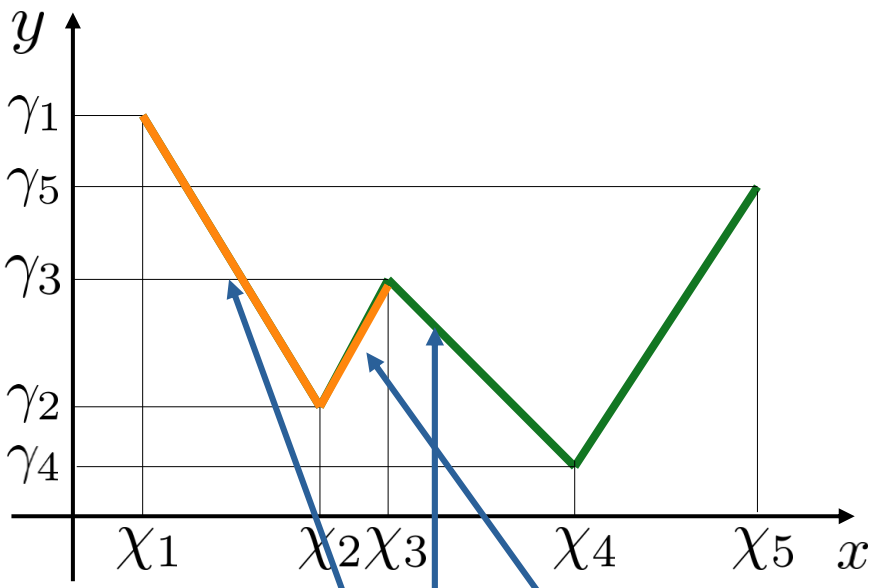
$x \in \mathbb{R}$

$(\underline{x}_1 \leq x \leq \overline{x}_1) \vee (\underline{x}_2 \leq x \leq \overline{x}_2)$

$u \in \{0, 1\}$

		$u = 0$	$u = 1$
$x \geq \underline{x}_1 + (\underline{x}_2 - \underline{x}_1)u$	$\longrightarrow$	$x \geq \underline{x}_1$	$x \geq \underline{x}_2$
$x \leq \overline{x}_1 + (\overline{x}_2 - \overline{x}_1)u$	$\longrightarrow$	$x \leq \overline{x}_1$	$x \leq \overline{x}_2$

Piecewise Linear Constraints



$w_3 = 0.75$

$w_4 = 0.25$

w_2 non-zero if $u_1 = 1$ or $u_2 = 1$

$w \in [0,1]^p$ weighting of each point

$$x = \sum_{k=1}^{p} \chi_k w_k \qquad y = \sum_{k=1}^{p} \gamma_k w_k$$

$$\sum_{k=1}^{p} w_k = 1$$

To stay in each line segment, at most 2 neighbouring weights can be non-zero.

$u \in \{0,1\}^{p-1}$ indicate if segment is active

$$\sum_{k=1}^{p-1} u_k = 1 \qquad w_1 \le u_1 \quad w_p \le u_{p-1}$$

$$w_k \le u_{k-1} + u_k, \quad \forall k \in \{2, \ldots, p-1\}$$

On-Off Constraints

$$x \in [\underline{x}, \overline{x}], \quad u \in \{0, 1\}$$

$$ux = 0 \qquad \text{i.e. } u = 1 \implies x = 0$$

$$\underline{x}(1-u) \le x \le \overline{x}(1-u)$$

Here (and in many of the previous slides) x could represent another expression that we can bound.

$$u = 1 \implies z = y$$

In above:

$$x = y - z$$

$$\underline{x} = \underline{y} - \overline{z}$$

$$\overline{x} = \overline{y} - \underline{z}$$

$$(\underline{y} - \overline{z})(1-u) \le y - z \le (\overline{y} - \underline{z})(1-u)$$

On-Off and Big-M

$$\underline{x}(1-u) \leq x \leq \overline{x}(1-u)$$

What if we can't easily derive bounds?

Make some up:

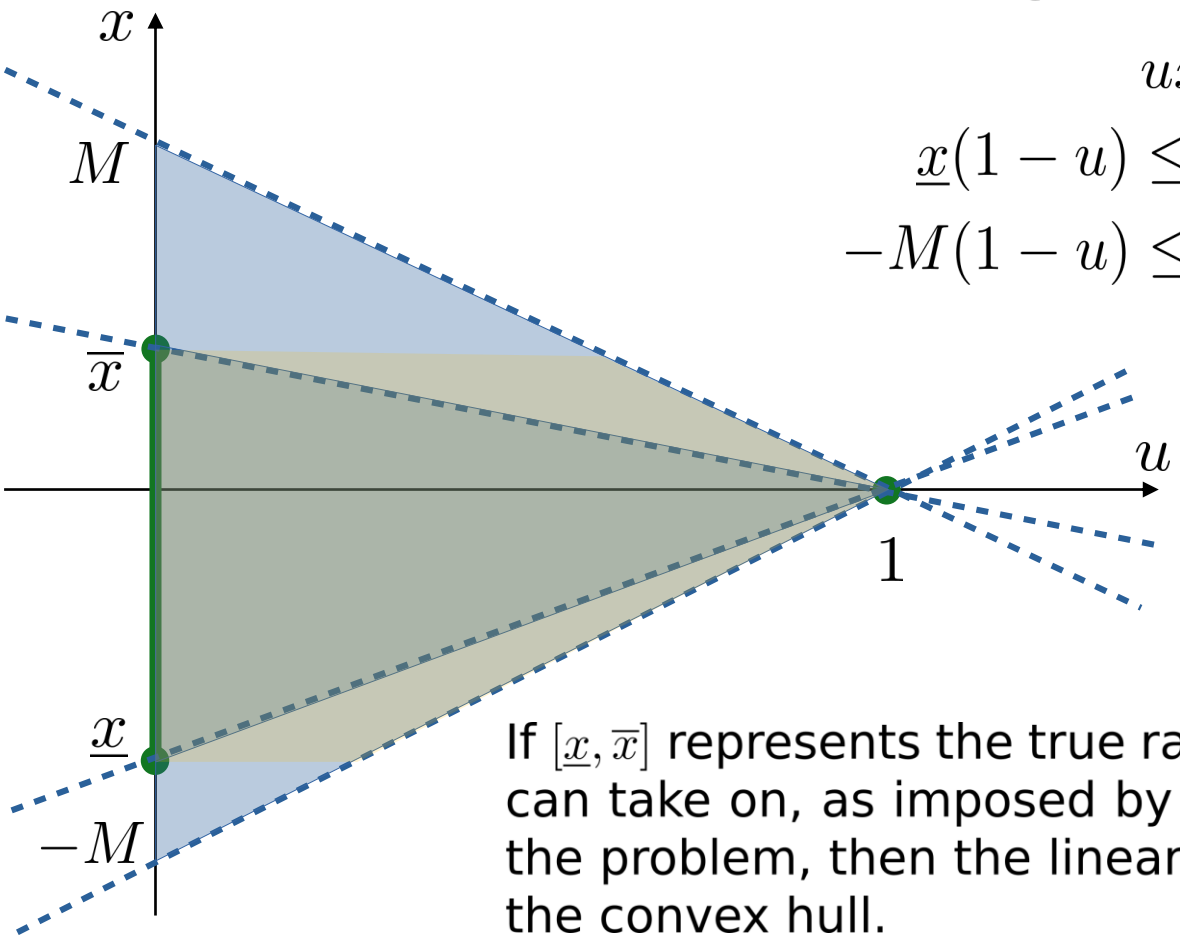
$$-M(1-u) \leq x \leq M(1-u)$$

Where M is some sufficiently large number (estimate the likely values x can take on).

After solving we can check whether or not these constraints are active at the optimal solution, and if so, maybe we need bigger M!

The linear relaxation gets poorer the larger M gets, so we generally want the bounds to be as tight as possible.

On-Off and Big-M



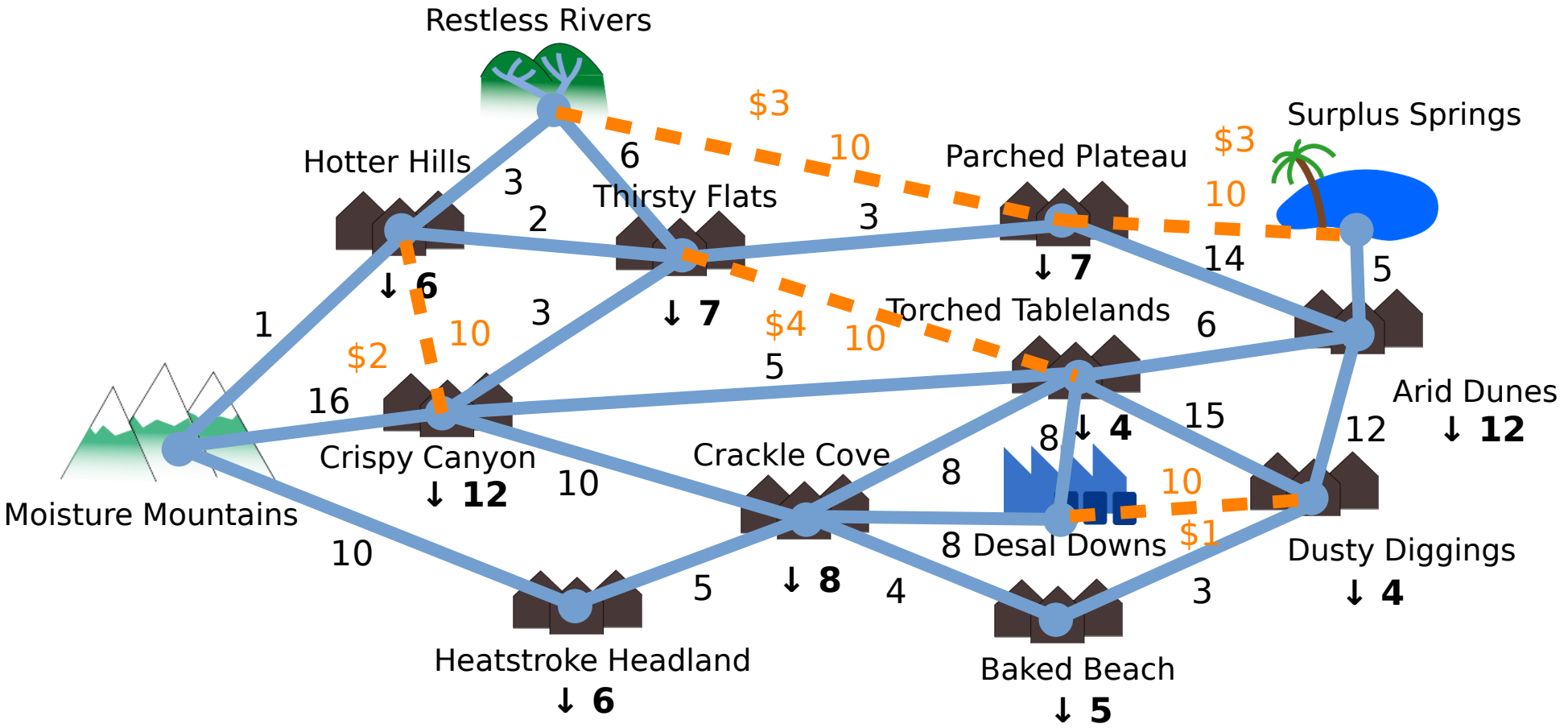
$$ux = 0$$

$$\underline{x}(1-u) \leq x \leq \overline{x}(1-u)$$

$$-M(1-u) \leq x \leq M(1-u)$$

If $[\underline{x}, \overline{x}]$ represents the true range of values that x can take on, as imposed by other constraints in the problem, then the linear relaxation will form the convex hull.

Water Network Expansion



Water Network Expansion

$$\min_{x,y,z} \sum_{(i,j)\in O} p_{i,j} z_{i,j}$$

Cost of new pipes

$$\text{s.t.} \sum_{(i,k)\in E} x_{i,k} - \sum_{(k,j)\in E} x_{k,j} + y_k - d_k = 0 \quad \forall k \in V$$

$$-c_{i,j} \le x_{i,j} \le c_{i,j} \quad \forall (i,j) \in E \setminus O$$

$$-c_{i,j} z_{i,j} \le x_{i,j} \le c_{i,j} z_{i,j} \quad \forall (i,j) \in O$$

$$y_k \ge 0 \quad \forall k \in V$$

$$z_{i,j} \in \{0,1\} \quad \forall (i,j) \in O$$

See water_expansion.py

Image Attributions

- By File:Exquisite-backpack.png, http://www.kde-look.org/content/show.php?content=14788, GPL, https://commons.wikimedia.org/w/index.php?curid=5851542
- By Al2 - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=2995931
- By Unknown - LSH 110188 (hm_dig24277), Public Domain, https://commons.wikimedia.org/w/index.php?curid=53731228
- By Szaaman - https://commons.wikimedia.org/wiki/File:Gold_bullion_1.jpg, Public Domain, https://commons.wikimedia.org/w/index.php?curid=35565318
- By Pigeons-in-holes.jpg by en:User:BenFrantzDale; this image by en:User:McKay - Transferred from en.wikipedia to Commons.; Original text : Edited from Image:Pigeons-in-holes.jpg, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4658682
- By 최광모 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=39329166