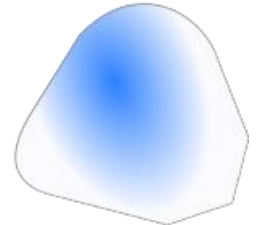
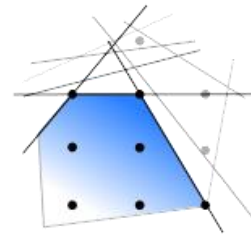
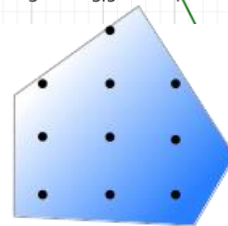
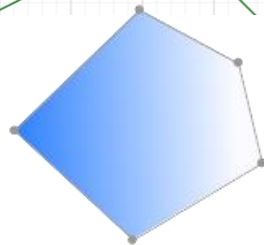
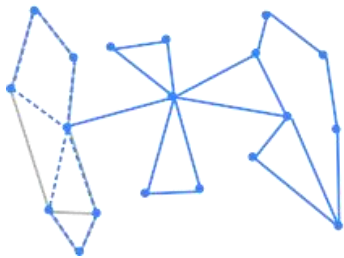
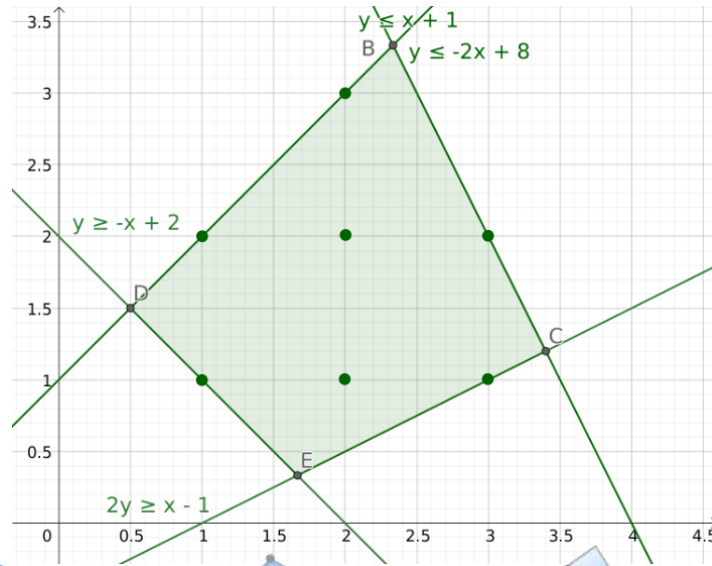


# Mixed-Integer Programming 1

COMP4691 / 8691



# Course Outline

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

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)**

← **MIP**

*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.**

FRACTIONAL

# Knapsack ILP



$$\max_x v^T x$$

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

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

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.

$$\max_x v^T x$$

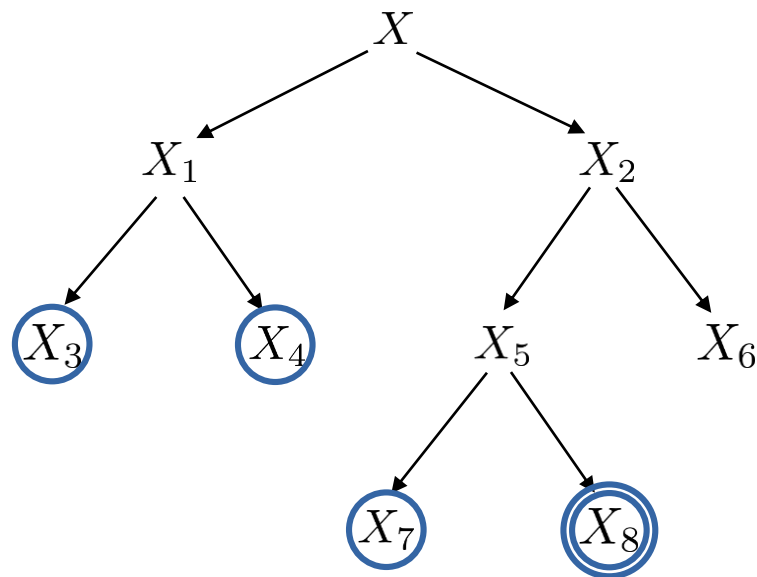
$$\text{s.t. } w^T x \leq \bar{w}$$

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

```
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  $\Delta-$
  - Linear relaxation  $\Delta-$
  - Convex hull  $\Delta-$
  - Delivery drone example  $\Delta-$
- **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.

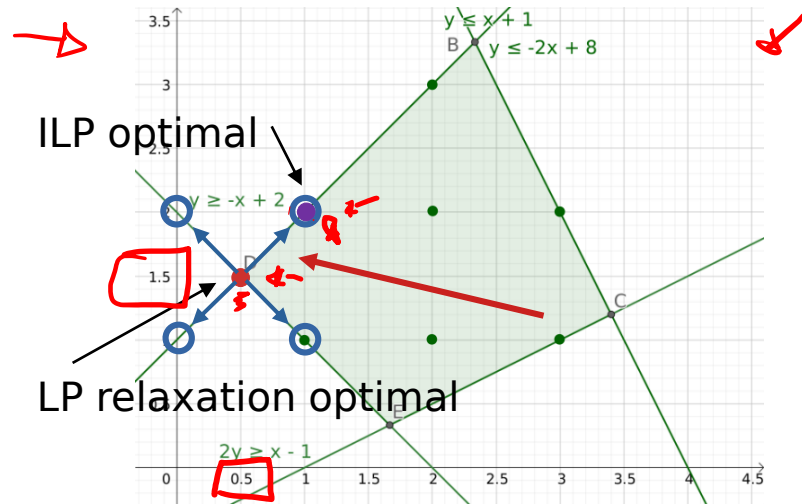
E.g., consider the subset of knapsack problems where all items have the same weight. Can you think of a simple algorithm to efficiently solve them?

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



MILP

$$\begin{aligned} \min & c^T x \\ \text{s.t.} & Ax \leq b \\ & x \in \mathbb{Z}^n \end{aligned}$$

FGAS REG  
GREEN DOTS

LP: RELAXED

$$\begin{aligned} \min & c^T x \\ \text{s.t.} & Ax \leq b \\ & x \in \mathbb{R}^n \end{aligned}$$

FEAS: GREEN AREA

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).

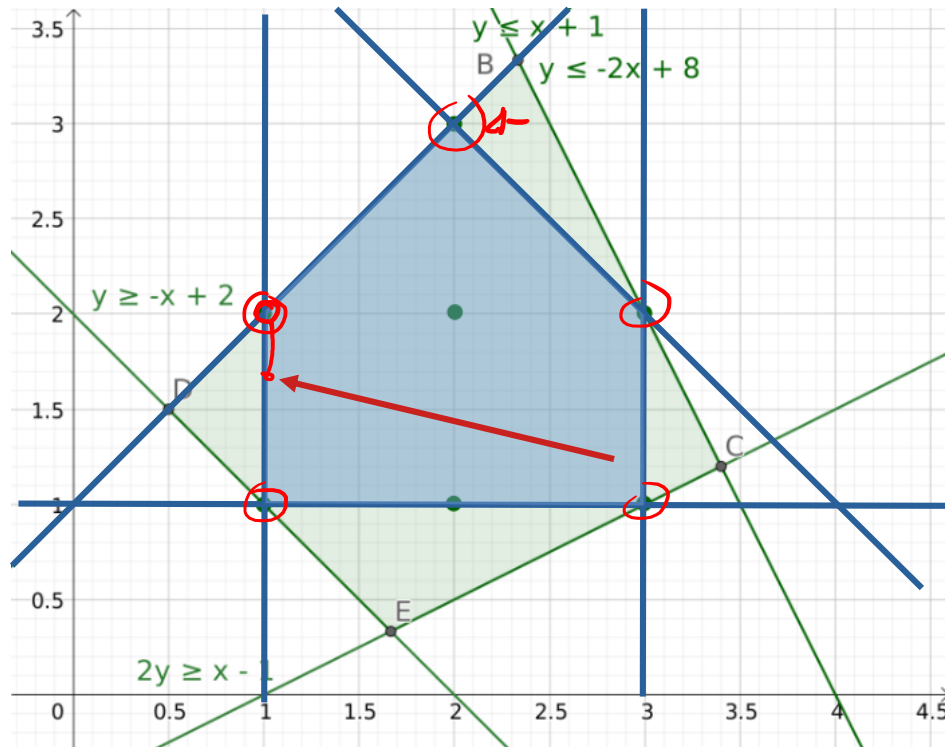
Rounding the LP solution: might get lucky, but also might be suboptimal or infeasible



# Linear Relaxation

ILP FEAS REG: GREEN  
DOTS

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

$$\left[ \begin{array}{ll} \min_x & c^\top x \\ \text{s.t.} & Ax \leq b \\ & x \in \mathbb{Z}^n \end{array} \right. \quad \begin{array}{ll} X & \text{set of ILP feasible points} \\ \text{conv}(X) & \text{convex hull of ILP feasible points} \\ L & \text{linear relaxation of ILP} \end{array} \quad \left| \quad \begin{array}{l} \text{convex} \\ \text{polyhedra} \end{array} \right.$$

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 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:

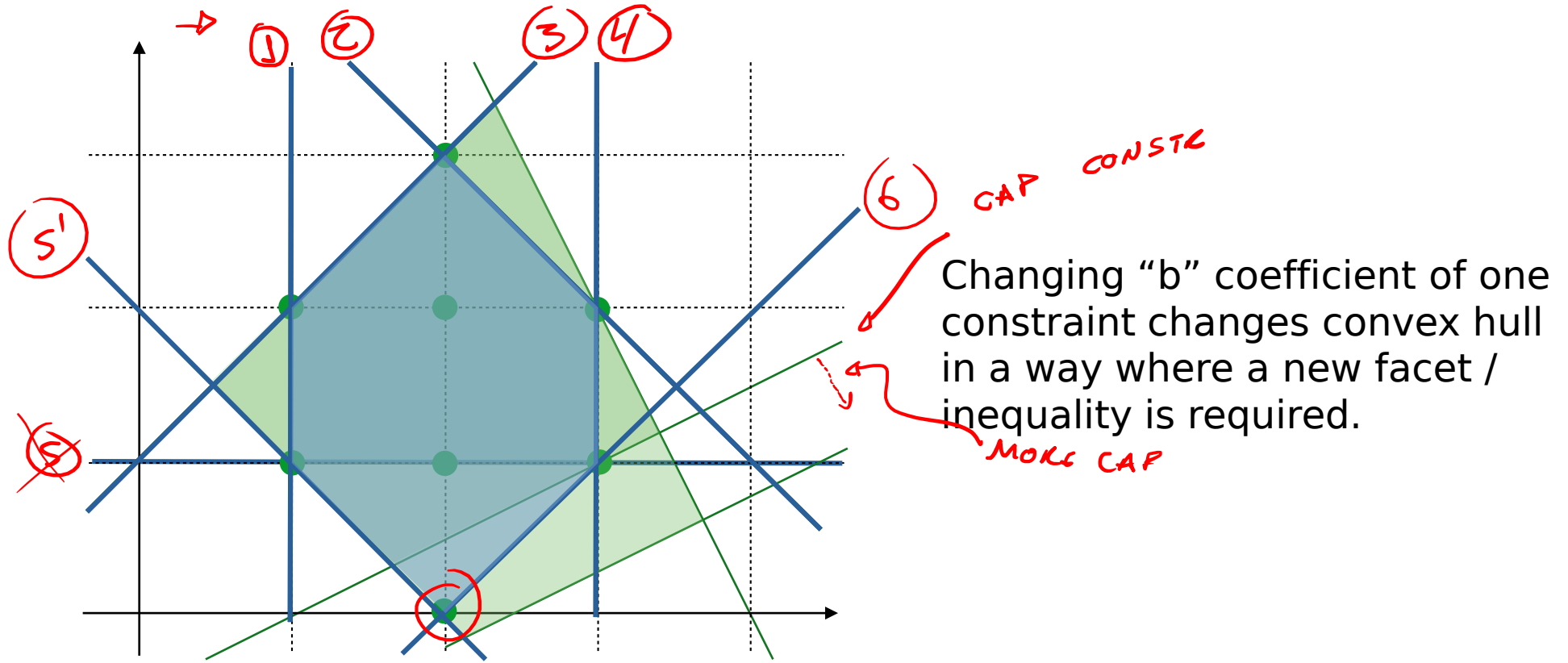
- solving an NP-hard problem in the process, or
- blowing up the formulation with too many constraints

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**.

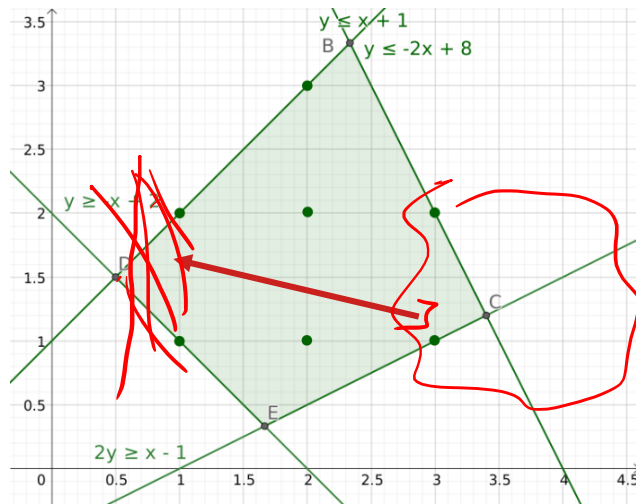
# Changing Hull



# 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?

$$\rightarrow x_{i,j} \in \{0, 1\} \quad \forall i \in \{1, \dots, p\}$$

Objective?

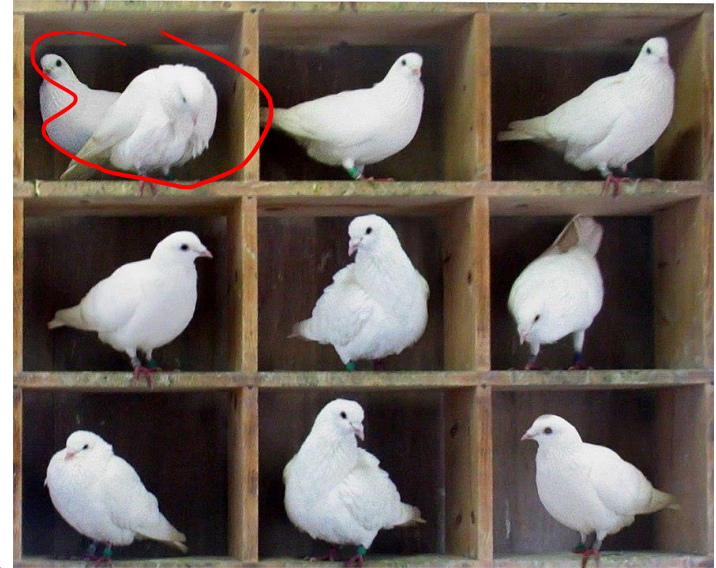
$$\forall j \in \{1, \dots, q\}$$

0

Constraints?

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

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



# Pigeonhole Principle

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

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

with

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

*2 PIGEONS THAT ARE DIF.*

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:  $x_{i,j} = \frac{1}{q}$

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

infeasible when p > q

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

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.

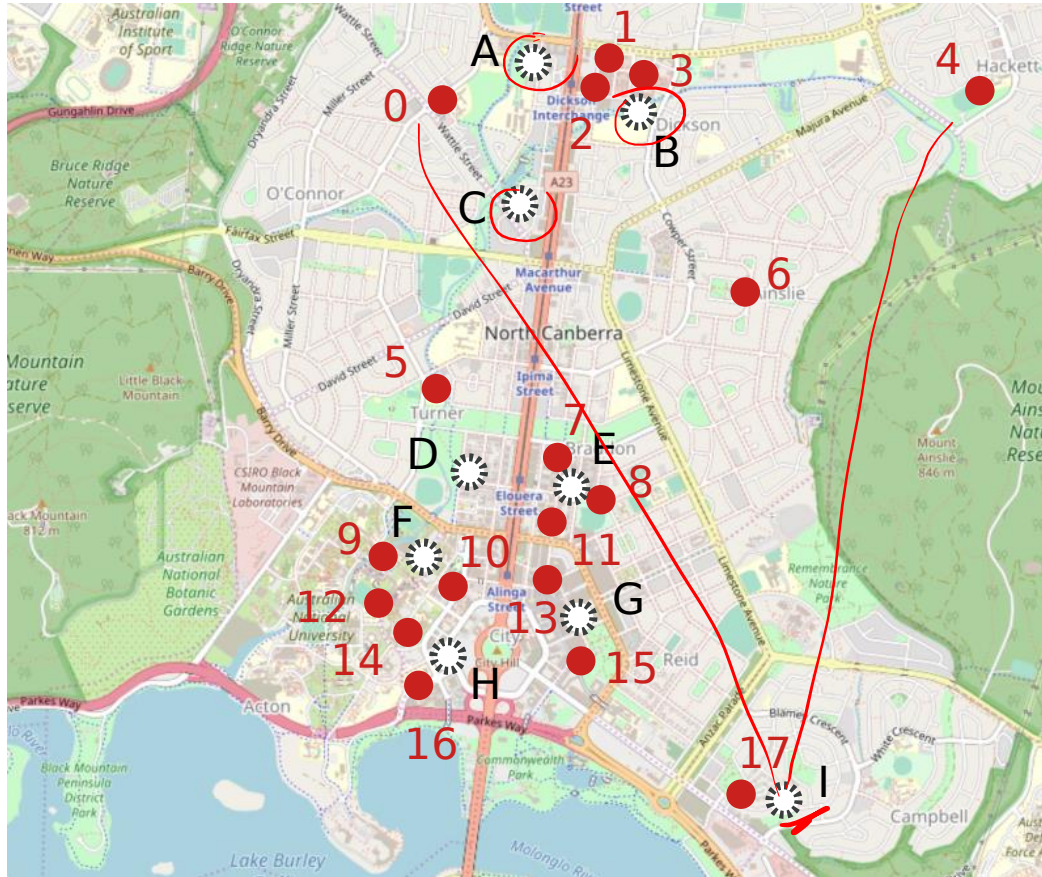
Locate hangers where drones can recharge and rest between jobs.

Should be close to restaurants.

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

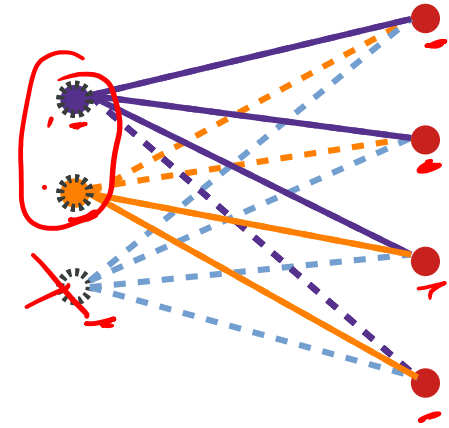
Must supply all restaurant contracted demand.

# Food Delivery Drone



- ● Restaurants *NUMBERS*
- ☼ Hanger Options *CATTERS*

## Bipartite Graph



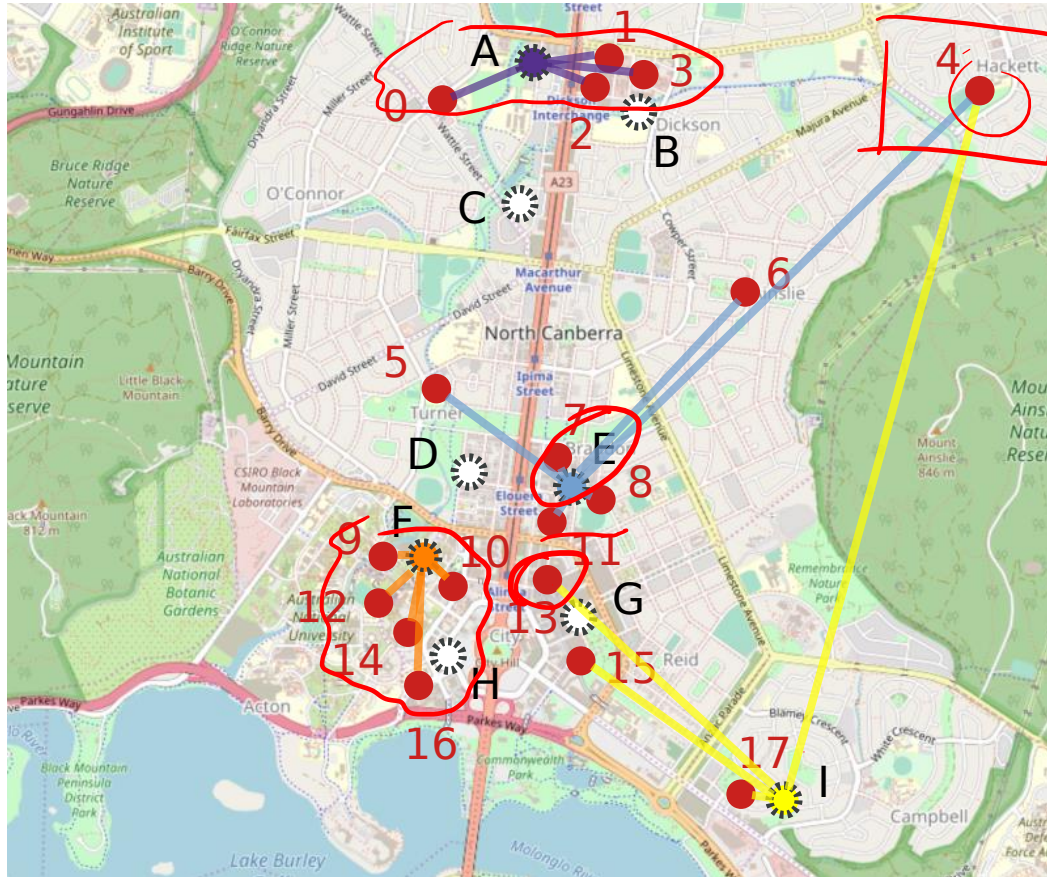
# Food Delivery Drone

$$\begin{aligned}
 & \min_{h,d} \sum_{s \in S} p_s h_s + \sum_{s \in S} \sum_{r \in R} p_{s,r} d_{s,r} \\
 & \text{s.t.} \quad \sum_{r \in R} d_{s,r} \leq c_s h_s \quad \forall s \in S \\
 & \quad \sum_{s \in S} d_{s,r} = t_r \quad \forall r \in R \\
 & \quad d_{s,r} \geq 0 \\
 & \quad h_s \in \{0, 1\}
 \end{aligned}$$

$\overbrace{p_s h_s}^{\text{BUILDING}}$   $\overbrace{\sum_{s \in S} \sum_{r \in R} p_{s,r} d_{s,r}}^{\text{COST OF CONTRACTS}}$  Cost to build and meet trips  
 $\overbrace{c_s}^{\text{PARAM}}$  Capacity of built hangers  
 All trips must be met  
 HANG. AT SITE 9

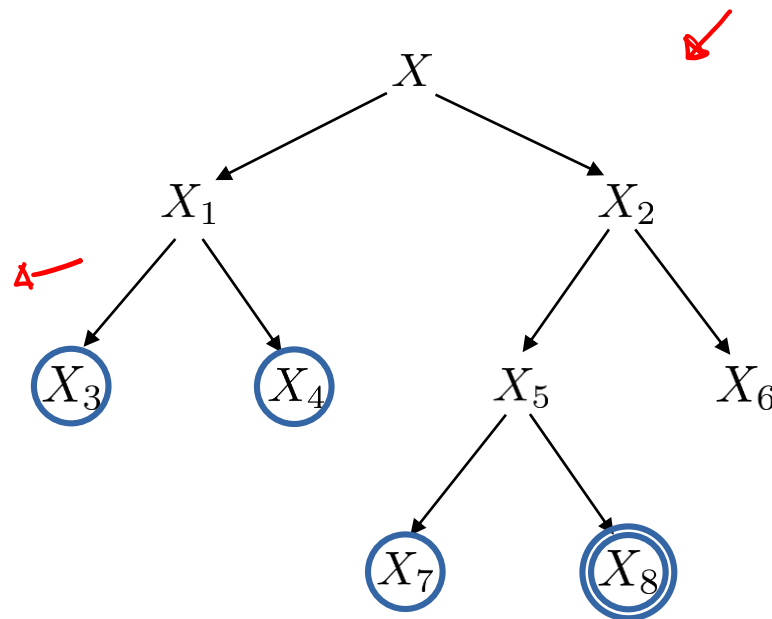
See drone\_hangers.py

# 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:

E.g., in knapsack problem:

$$\max_x \quad \underline{v^\top x}$$

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

$$\textcircled{x} \in \{0, 1\}$$

At most  $p$  items selected:

$$\sum_j x_j \leq p$$

$$S = \{2, 3, 4, 5, 6\} \quad s \in \mathbb{Z} \rightarrow S \geq 2 \quad s \leq 6$$

Variable  $\boxed{y}$  can take on one value from set  
 $S = \{s_1, s_2, \dots, s_k\} = \{2, 5, 7, 11\}$

$$\textcircled{y} = \sum_j s_j x_j$$

$$\sum_j x_j = 1$$

# Logical Constraints

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

$$x \overset{\text{IMPLIES}}{\implies} y$$

$$z = \underline{x} \vee \underline{y}$$

$$z = x \wedge y$$

$$z = \neg x$$

$$\underline{y} \geq x$$

$$\underline{z} \geq \underline{x}, \quad z \geq \underline{y},$$

$$\underline{z} \leq \underline{x}, \quad \underline{z} \leq \underline{y},$$

$$\underline{z} = \underline{1 - x}$$

Force Zero

$$z \leq \underline{x} + \underline{y}$$

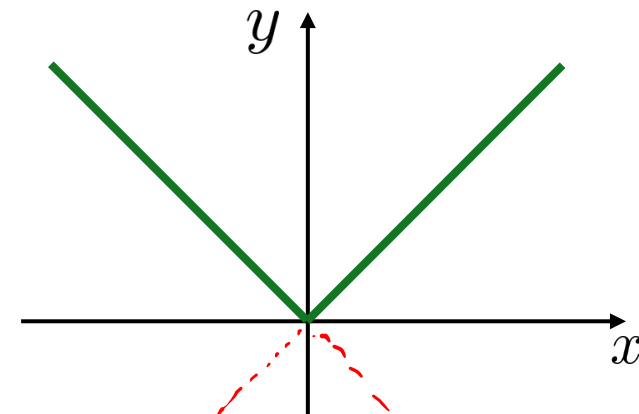
$$z \geq \underline{x + y - 1}$$



# Nonlinear Functions

$x \in [\underline{x}, \bar{x}], \quad y \in \mathbb{R}$   
 $y = |x|$

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



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

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

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

$u = 0$

$y = x$

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

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

$u = 1$

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

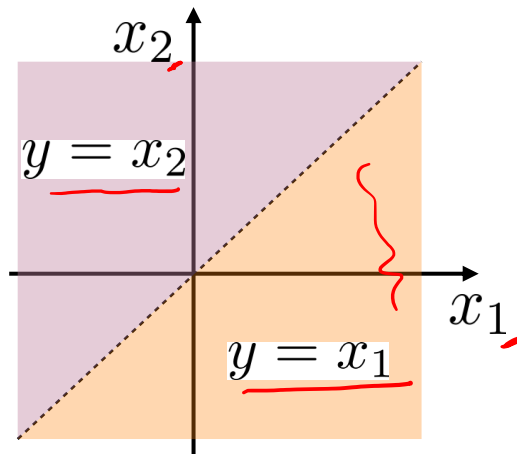
$y = -x$

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

# Nonlinear Functions

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

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



Min y

Max y

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

$$y \geq x_j$$

$$y = x_j \text{ if } u_j = 1$$

$$j | u_j = 1$$

$$\forall j | u_j = 0$$

$$y \leq x_j + (\max_i(\bar{x}_i) - \underline{x}_j)(1 - u_j) \rightarrow y \leq x_j$$

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

$$\sum_j u_j = 1$$

$$x_j = \underline{x}_j$$

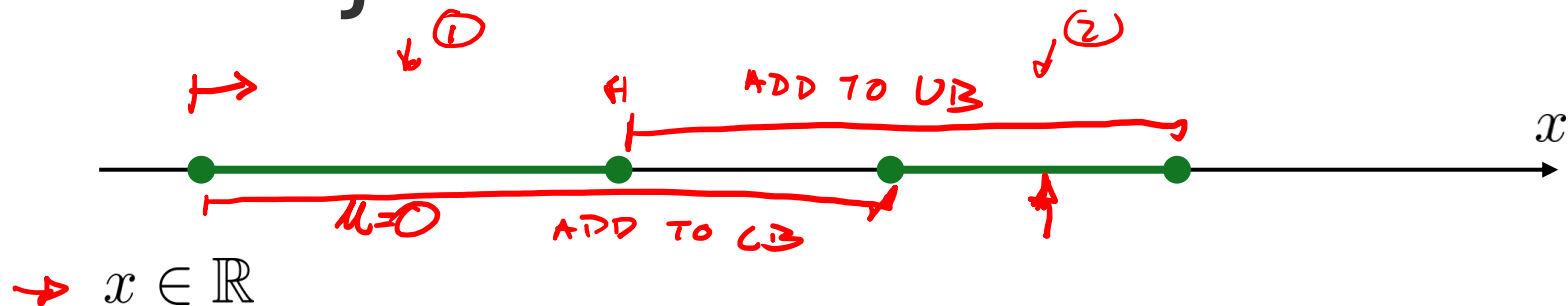
$$y \leq \max_i(\bar{x}_i)$$

$$x_j = \bar{x}_j$$

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

POSITIVE  
(RANGE OF  $x_j$ )

# Disjunctive Constraints



$$\underline{x}_1 \leq x \leq \bar{x}_1 \vee \underline{x}_2 \leq x \leq \bar{x}_2$$

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

$$\left. \begin{array}{l} \underline{x} \geq \underline{x}_1 + (\underline{x}_2 - \underline{x}_1)u \\ \underline{x} \leq \bar{x}_1 + (\bar{x}_2 - \bar{x}_1)u \end{array} \right\} \longrightarrow$$

$$u = 0$$

$$x \geq \underline{x}_1$$

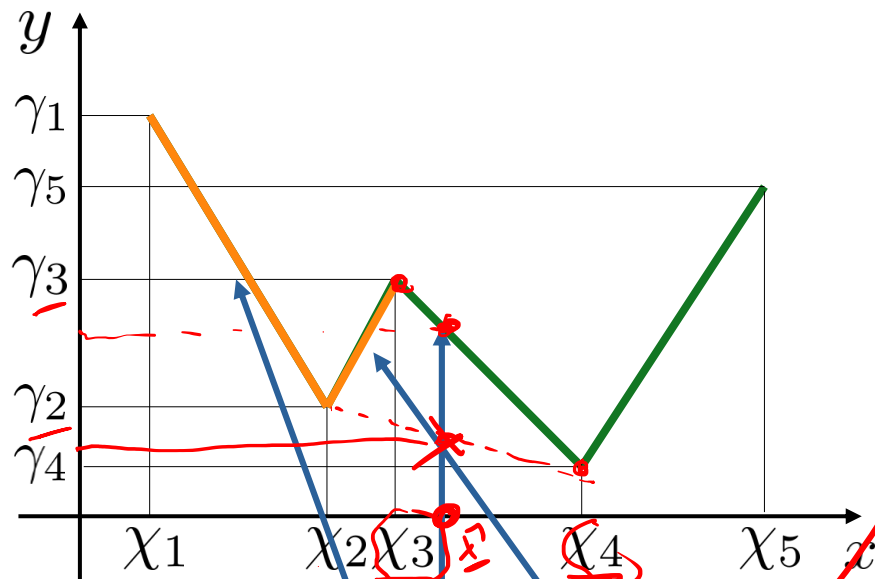
$$x \leq \bar{x}_1$$

$$u = 1$$

$$x \geq \underline{x}_2$$

$$x \leq \bar{x}_2$$

# Piecewise Linear Constraints



$w \in [0, 1]^p$   $\rightarrow$  5 points in the example  
weighting of each point 4-

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

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

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

$$\sum_{k=1}^{p-1} u_k = 1$$

$$w_1 \leq u_1 \quad u_p \leq u_{p-1}$$

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

ACTIVE SEG ON BOTH SIDES

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

# On-Off Constraints

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

$$ux = 0$$

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

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

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

$$\rightarrow \underline{u} = 1 \implies \underline{z} = \underline{y}$$

In above:  $\rightarrow \underline{x} = \underline{y} - \underline{z}$

$$\rightarrow \underline{x} = \underline{y} - \bar{z}$$

$$\rightarrow \bar{x} = \bar{y} - \underline{z}$$

$$(\underline{y} - \bar{z})(1 - u) \leq y - z \leq (\bar{y} - \underline{z})(1 - u)$$

# On-Off and Big-M

$$\underline{x}(1-u) \leq x \leq \bar{x}(1-u) \quad u=0 \quad x \text{ is BOUNDED } [\underline{x}, \bar{x}]$$

What if we can't easily derive bounds?

Make some up:

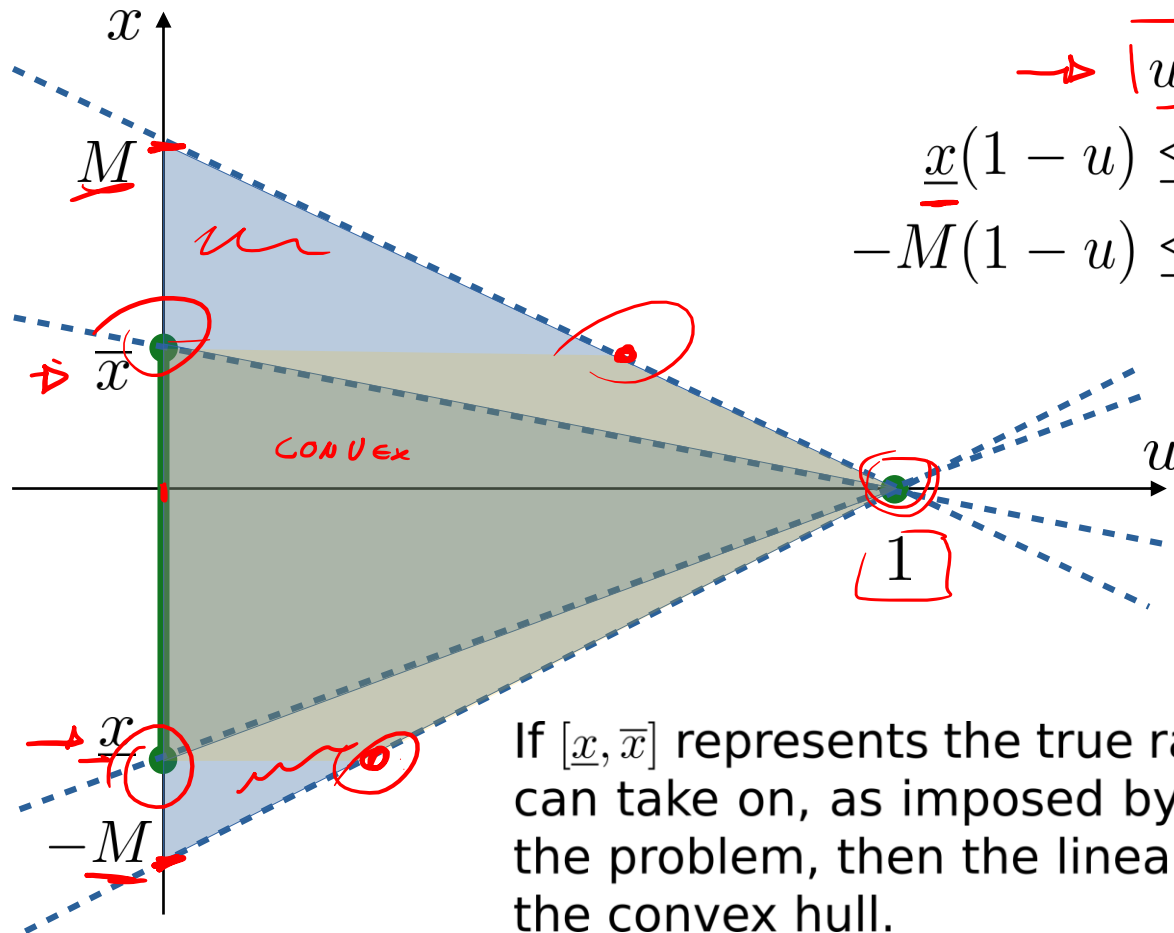
$$-M(1-u) \leq x \leq \underline{M}(1-u)$$

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



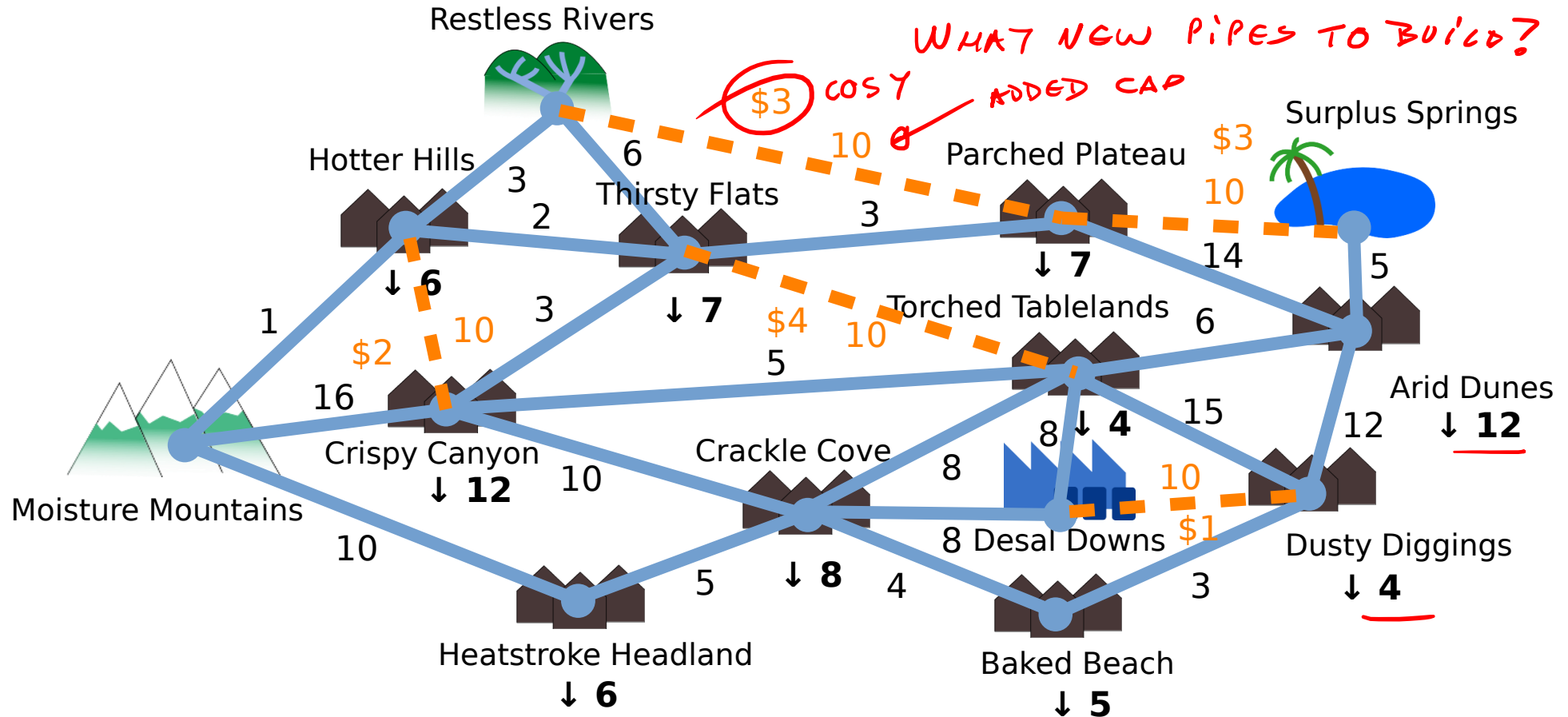
$$\rightarrow \boxed{ux = 0}$$

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

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

If  $[\underline{x}, \bar{x}]$  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

$$\begin{aligned}
 & \rightarrow \min_{x,y,z} \sum_{(i,j) \in O} p_{i,j} z_{i,j} \quad \leftarrow \text{Cost of new pipes} \\
 & \text{s.t.} \quad \sum_{(i,k) \in E} x_{i,k} - \sum_{(k,j) \in E} x_{k,j} + y_k - d_k = 0 \quad \forall k \in V \quad \leftarrow \text{PC OF FLOW} \\
 & \quad -c_{i,j} \leq x_{i,j} \leq c_{i,j} \quad \forall (i,j) \in E \setminus O \\
 & \quad -\underline{c_{i,j} z_{i,j}} \leq x_{i,j} \leq \underline{c_{i,j} z_{i,j}} \quad \forall (i,j) \in O \quad \leftarrow \text{OPTIONS} \\
 & \quad y_k \geq 0 \quad \forall k \in V \quad \leftarrow \text{PARAM} \\
 & \rightarrow z_{i,j} \in \{0, 1\} \quad \forall (i,j) \in O
 \end{aligned}$$

*Handwritten notes:*  
 - A red circle around  $z$  in the objective function has an arrow pointing to it.  
 - A red bracket above  $p_{i,j} z_{i,j}$  has an arrow pointing to the text "Cost of new pipes".  
 - A red arrow points from the text "SAMPLING OF REF" to the first constraint.  
 - A red arrow points from the text "FOR EXISTING NET." to the second constraint.  
 - A red box around  $\forall (i,j) \in O$  in the third constraint has an arrow pointing to the word "OPTIONS".  
 - A red arrow points from the text "PARAM" to the fourth constraint.

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 AI2 - 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](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>