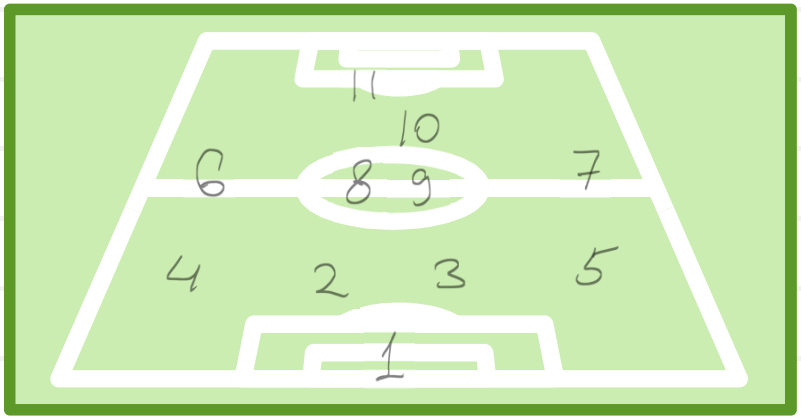
Lecture 9: Solving Integer Linear Programs

Recap

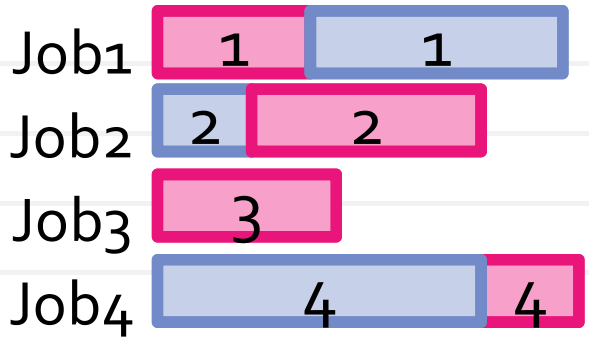


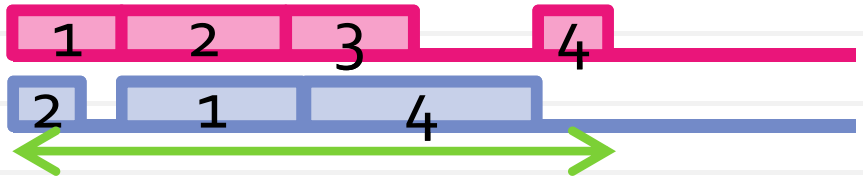
$$\min \; p^T x$$

$$s.t: \; Ax \leq b$$

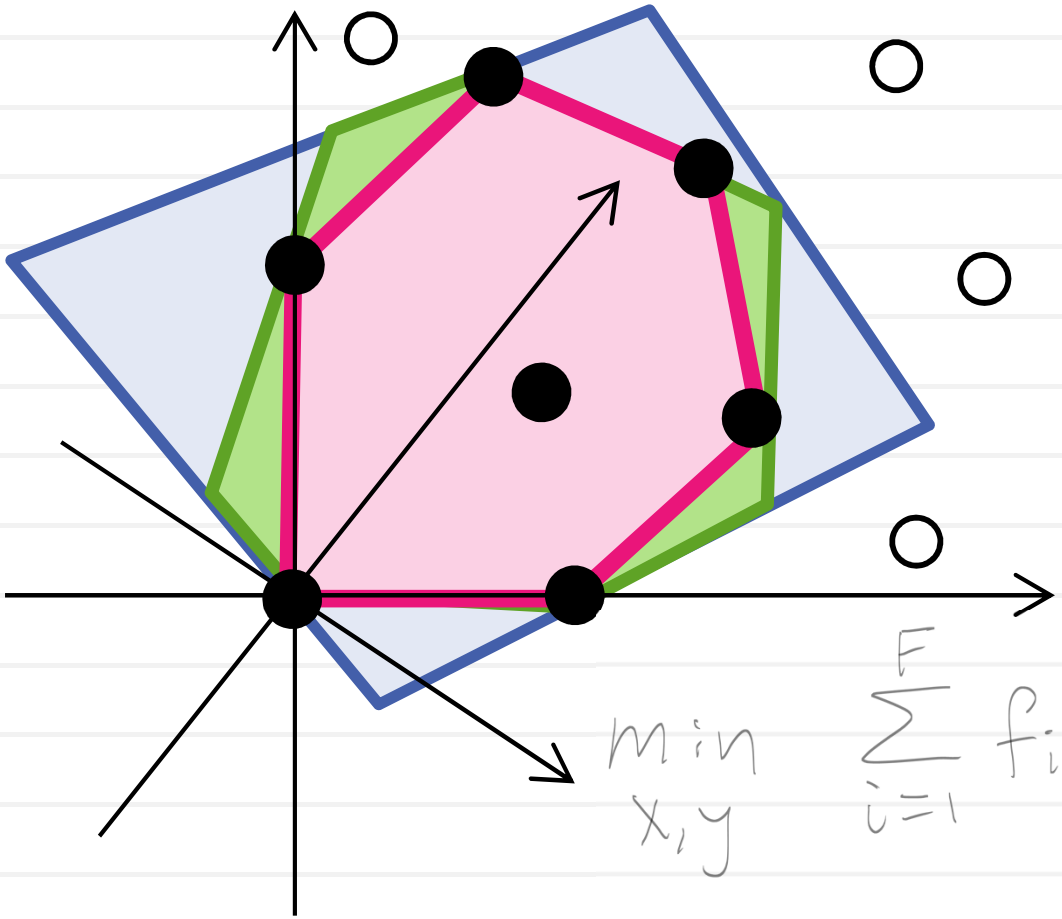
$$Cx = d$$

$$\forall i \in I \quad x_i \in \{0;1\}$$





Integer programs and polytopes



$$\min_{x,y} \sum_{i=1}^{F} f_i y_i + \sum_{i=1}^{F} \sum_{j=1}^{C} a_{ji} \cdot x_{ji}$$

$$x_i, y_i \in \{0;1\} \quad \forall j \sum_{i=1}^{F} x_{ji} = 1$$

$$\forall i,j \quad x_{ji} \le y_i \qquad \sum_{j=1}^{C} x_{ji} \le C \cdot y_i$$

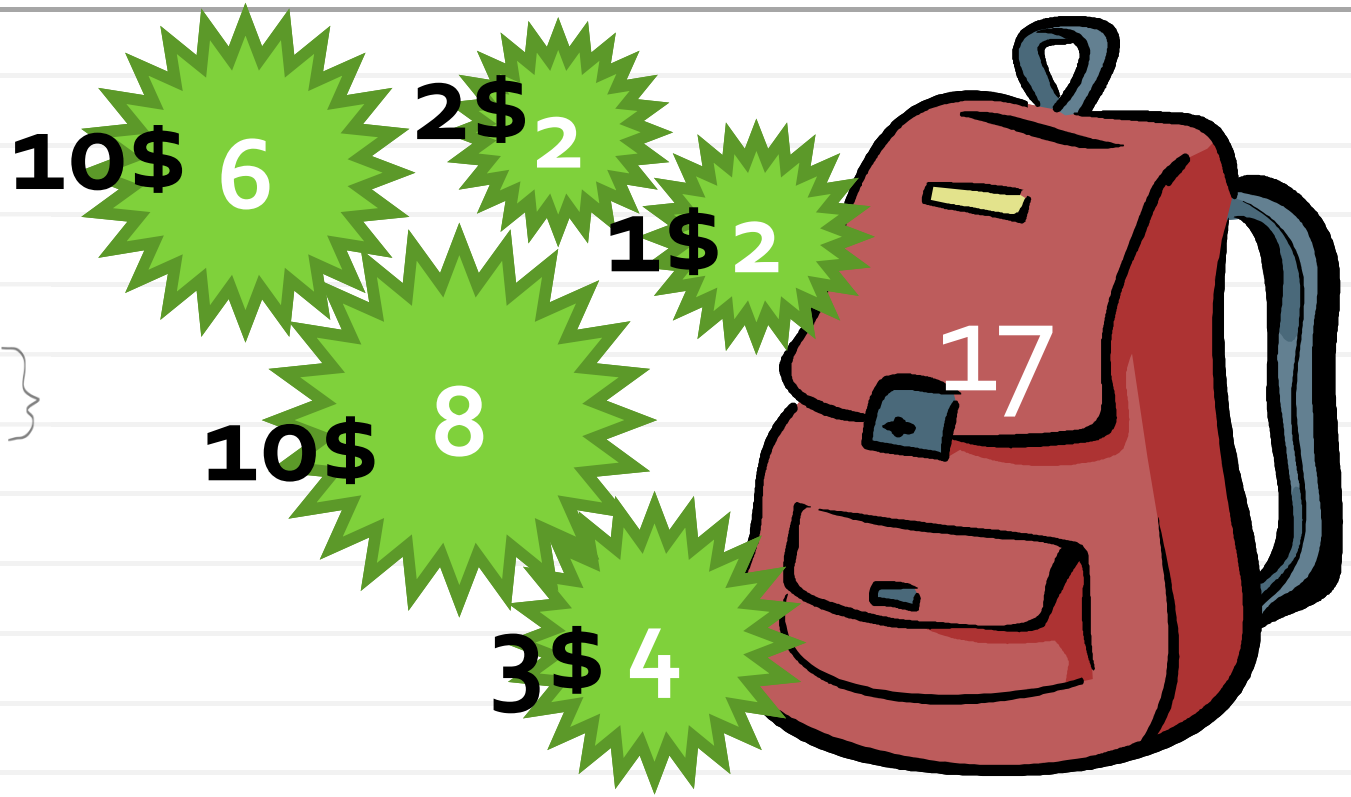
Now what?

We get some fractional solution. What to do next?

- We can round to the nearest integer solution (no feasibility guarantees!)
- We can use fractional values to guide greedy/local search
- More principled methods: branch-and-bound, cutting plane, branch-and-cut

Knapsack again

$$\min -c^T x$$
$$s.t: S^T x \le S$$
$$x_i \in \{0,1\}$$



Relaxation:

$$\min -c^T x$$
$$s.t: S^T x \le S$$
$$0 \le x \le 1$$

$$\Rightarrow$$

c = [10 2 3 1 10]

s = [6 2 4 2 8]

x = [1 1 0.25 0 1]

Obj = -22.75

Branching, bounding, pruning

{}

$x = [1\ \ 1\ \ 0.25\ \ 0\ \ 1]$

Obj = -22.75

{x3 = 0}

$x = [1\ \ 1\ \ 0\ \ 0.5\ \ 1]$

Obj = -22.5

{x3 = 1}

$x = [1\ \ 1\ \ 1\ \ 0.5\ \ 1]$

Obj = -21.75

pruned

{x3 = 0, x4=0}

$x = [1\ \ 1\ \ 0\ \ 0\ \ 1]$

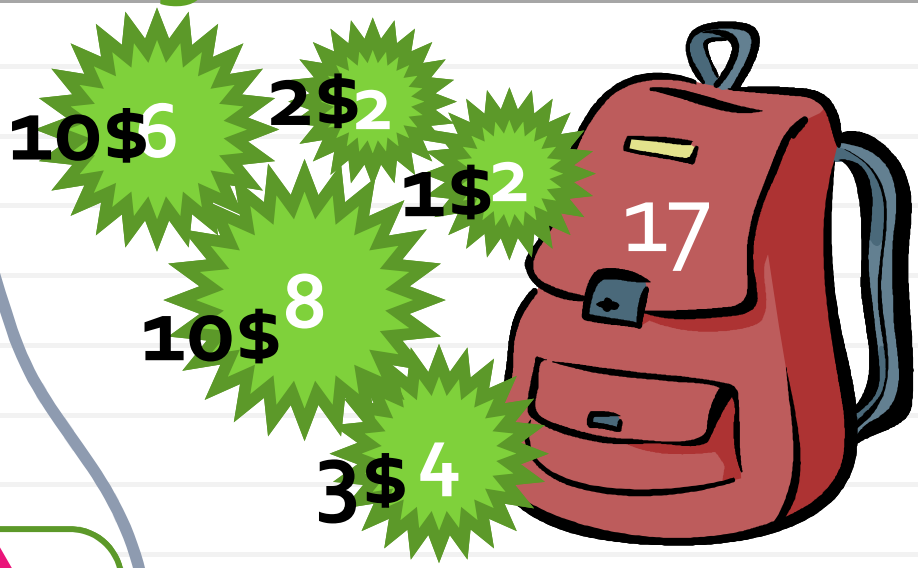
Obj = -22

{x3 = 0, x4=1}

$x = [1\ \ 1\ \ 0\ \ 1\ \ 1]$

Obj = -22

pruned



$c = [10\ \ 2\ \ 3\ \ 1\ \ 10]$

$s = [\ 6\ \ 2\ \ 4\ \ 2\ \ 8\ ]$

$\min -c^T x$

$s.t: s^T x \leq S$

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

$0 \leq x_i \leq 1$

Branch-and-bound parameters

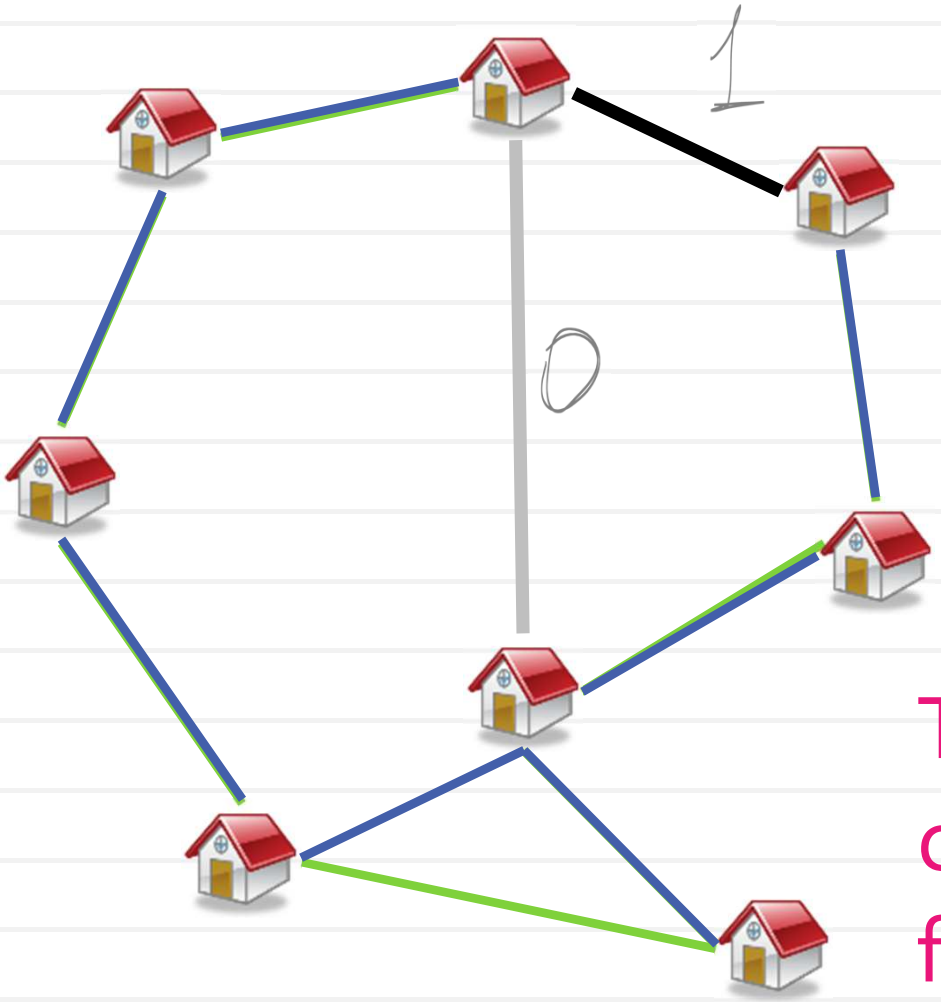
- Choice one: which variable to branch on
  Typical answer: the most fractional

- Choice two: which node to branch
  Typical answers: best-first or depth-first

Bound choices:

- Relaxation
- Lagrangian dual
- Customary bound for your task

Example: a bound for TSP

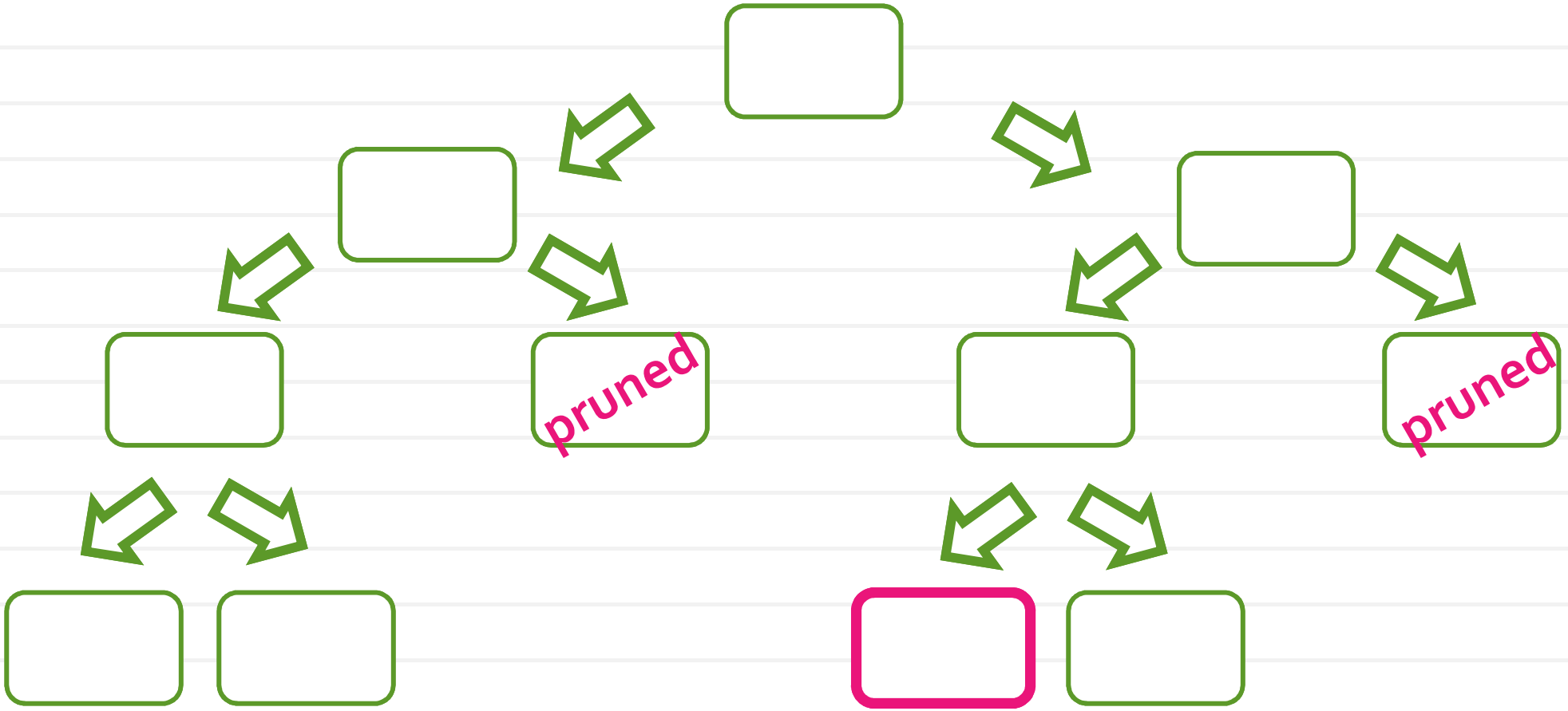


The weight of the optimal tree is a lower bound on the weight of the optimal tour

The MST algorithms give cheap and good bounds for TSP (can be made even better using Lagrange duality)

Best-first search

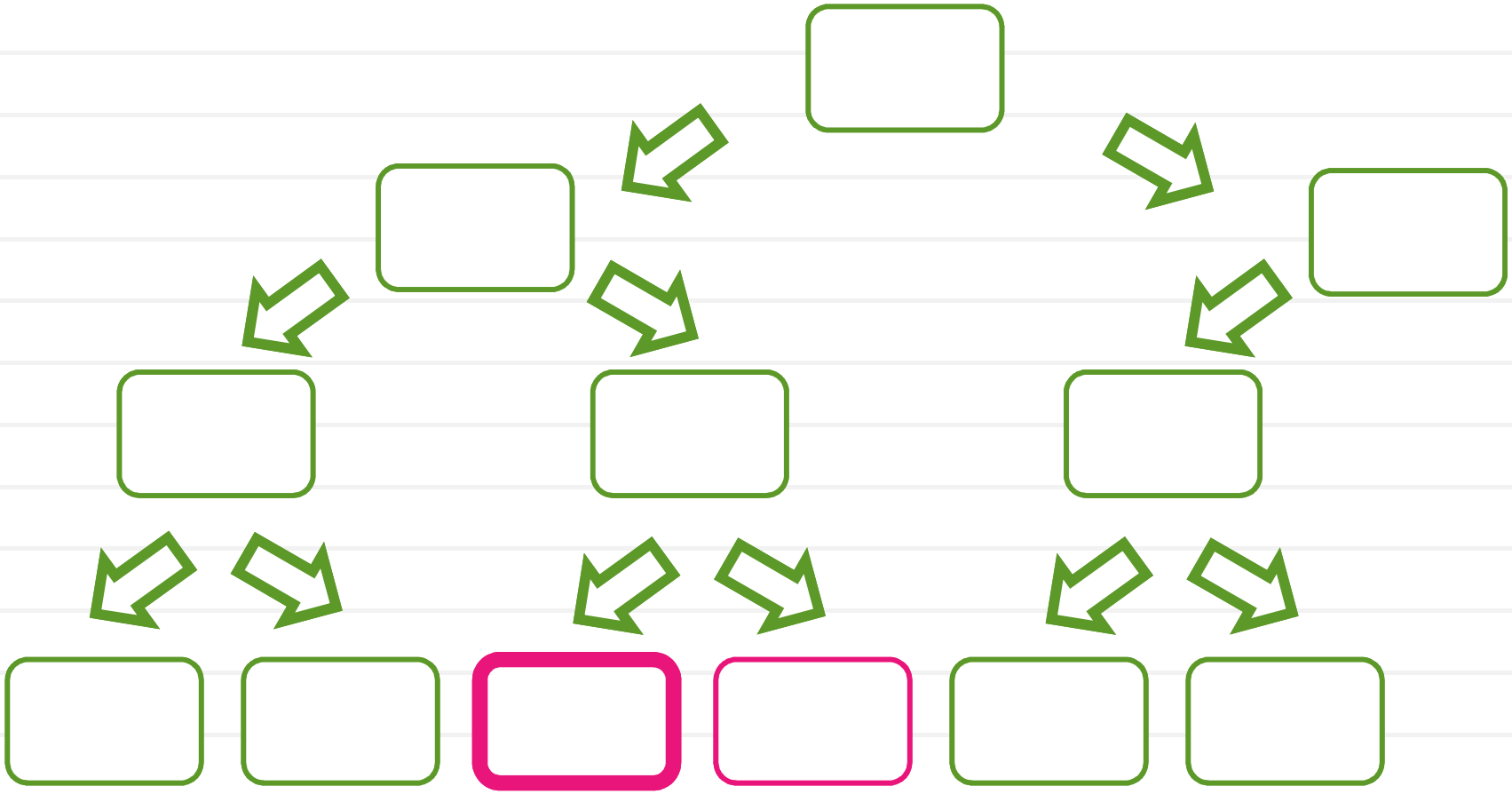
Idea: always explore the branch with the lowest bound



+: the most economical, only explores the nodes that we must to, to assert optimality

Depth-first search

Idea: always explore the most integer branch



+: finds some incumbents early-on. Can terminate early with an approximate solution.

- :if run to optimality, will explore more nodes than breadth-first

Branch-and-bound summary

Initialize the list *L* of the active nodes to the root node

incumbent_value = +∞

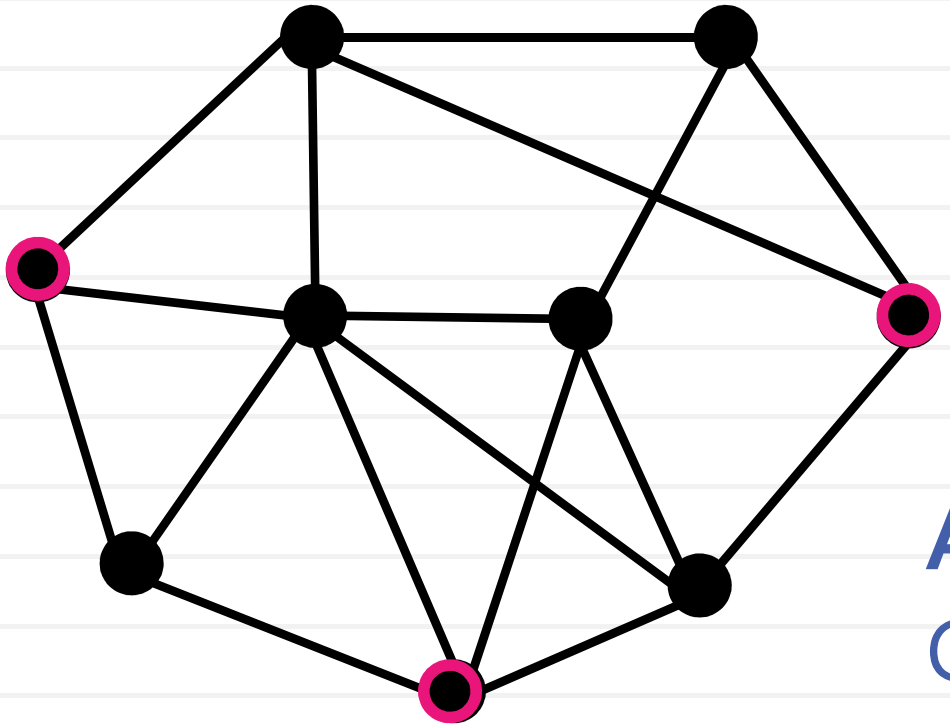
Until *L* is empty or time is out

- Pick the active node from *L (e.g. the deepest or with the smallest lower bound)*
- Pick the variable to branch
- **For each** of the two children nodes
  - *[x obj]* = Solve LP (node)
  - **If** *obj* ≥ *incumbent_value* **then continue;** //pruning
  - **If** *x* is integer **then**

    incumbent = *x*; *incumbent_value* = *obj*;
  - **Else** add node to *L*

MWIS problem

Maximum weight independent set (node packing) problem:



$$\max_{x} \; c^{\top} x$$

$$s.t.: \; \forall (i,j) \in E \quad x_i + x_j \leq 1$$

$$\forall i \quad x_i \in \{0, 1\}$$

Applications:

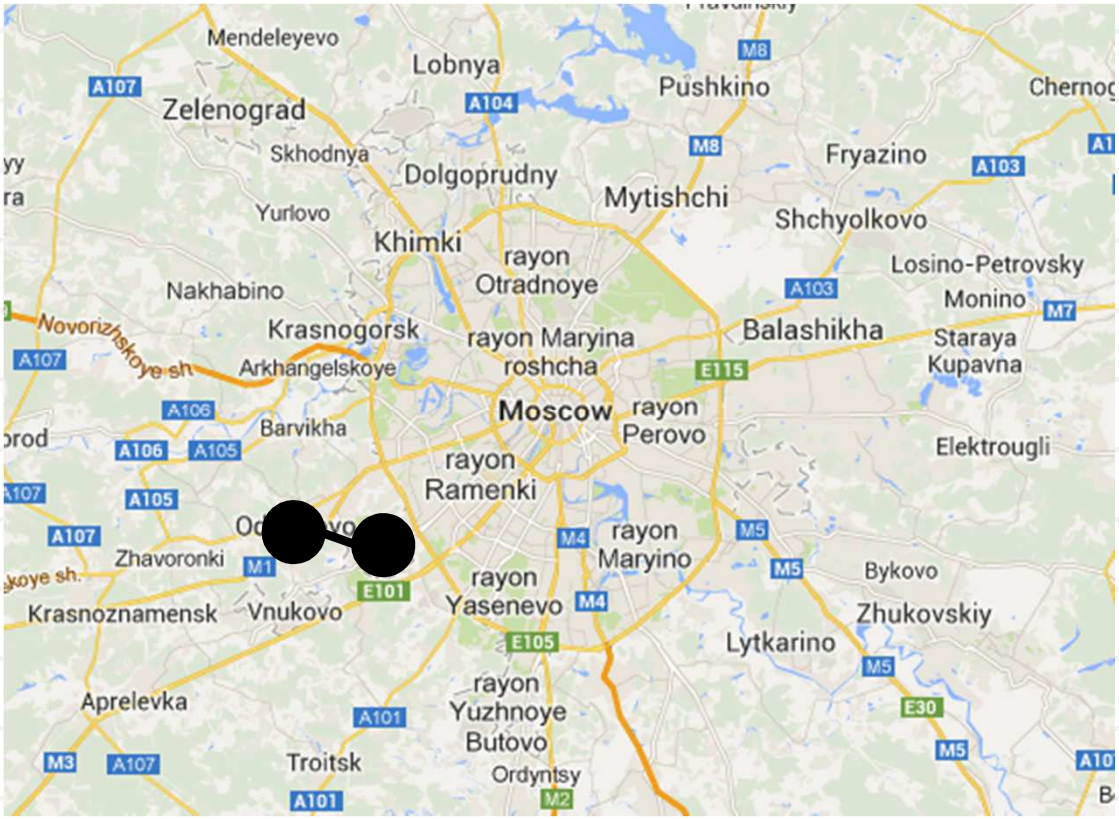
Choosing courses

Inviting people to a party

Map coloring

Map labeling

Map labeling

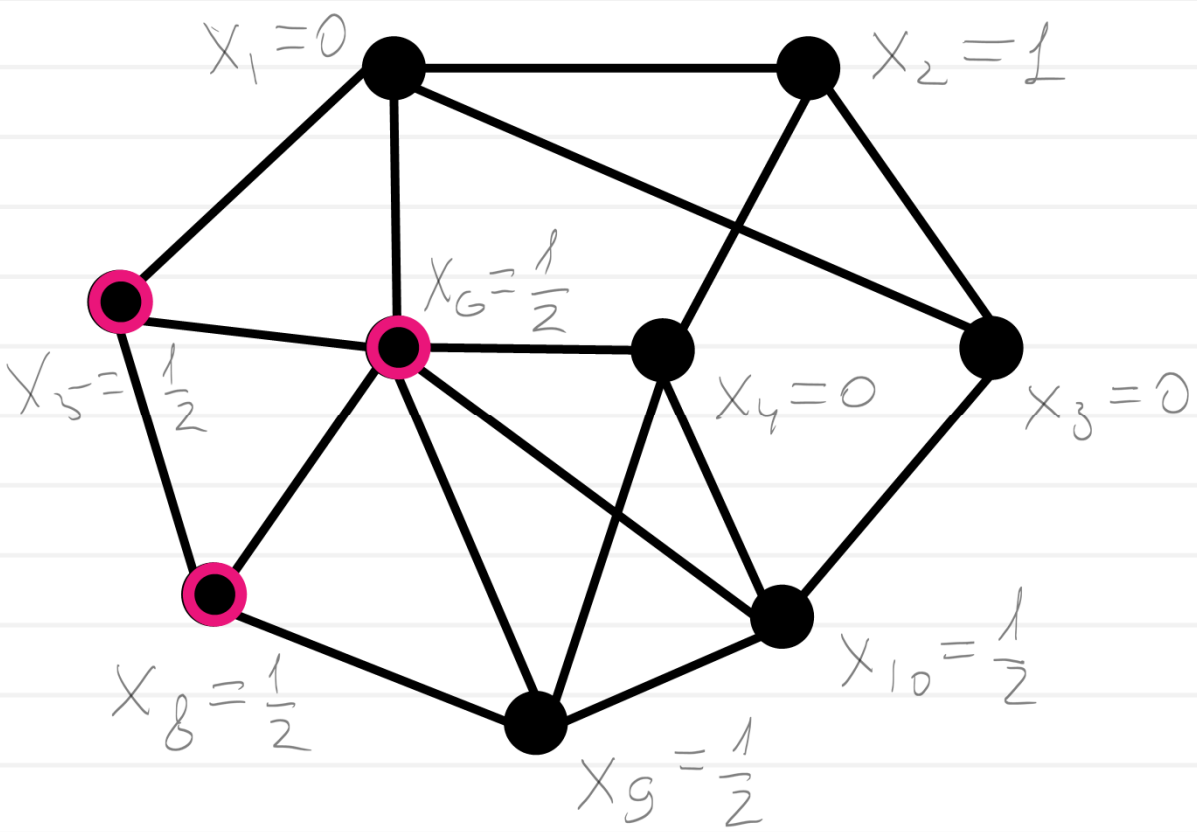


$$\max_{x} \; c^T x$$

$$s.t.: \; \forall (i,j) \in E \quad x_i + x_j \leq 1$$

$$\forall i \quad x_i \in \{0; 1\}$$

LP relaxation for MWIS



$$\min_x \; w^T x$$

$$s.t.: \; \forall (i,j) \in E \quad x_i + x_j \leq 1$$

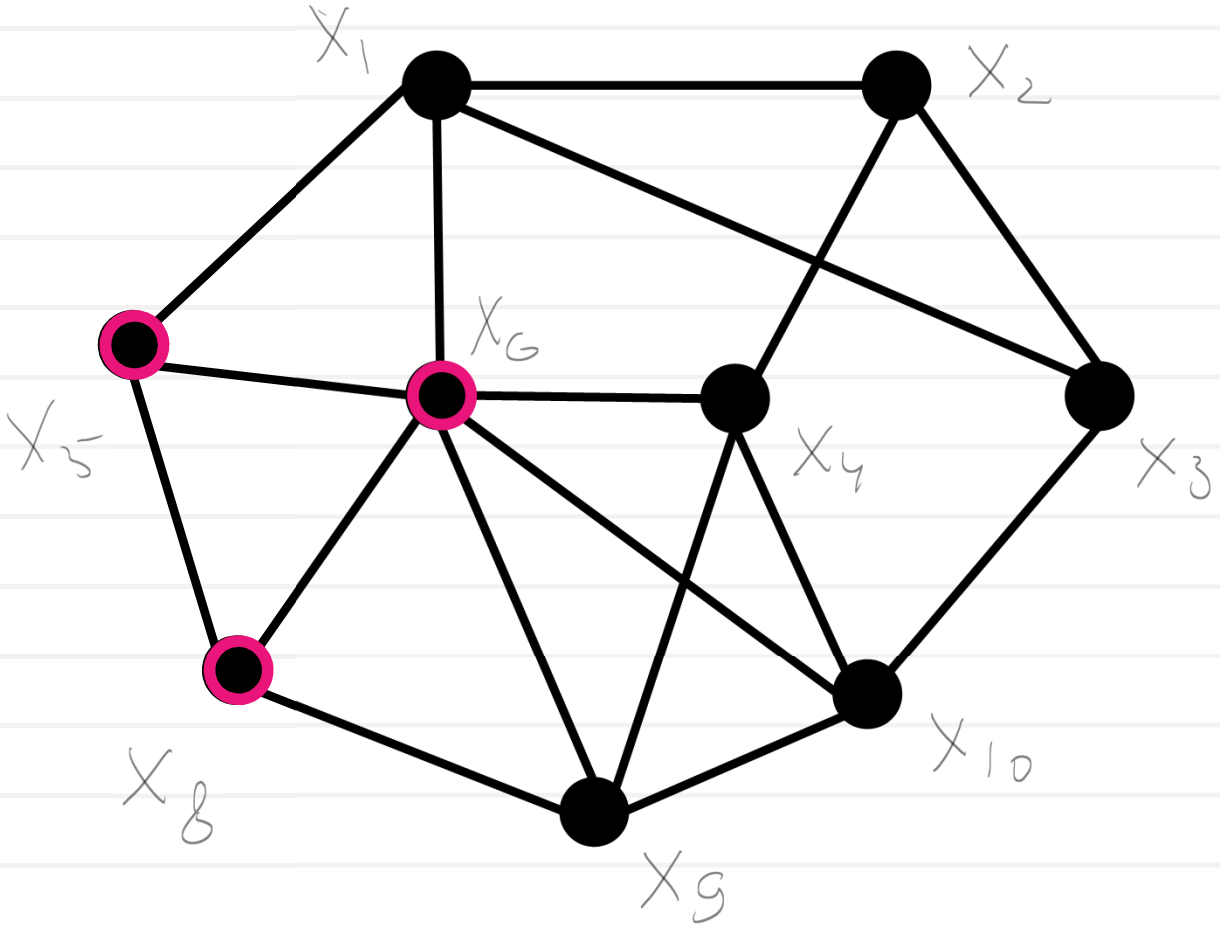
$$\forall i \quad 0 \leq x_i \leq 1$$

Is there a way to tighten the relaxation?

Yes!

$$x_5 + x_6 + x_8 \leq 1$$

MWIS problem



$$\min_{x} \; \omega^T x$$

$$s.t.: \; \forall (i,j) \in E \quad x_i + x_j \leq 1$$

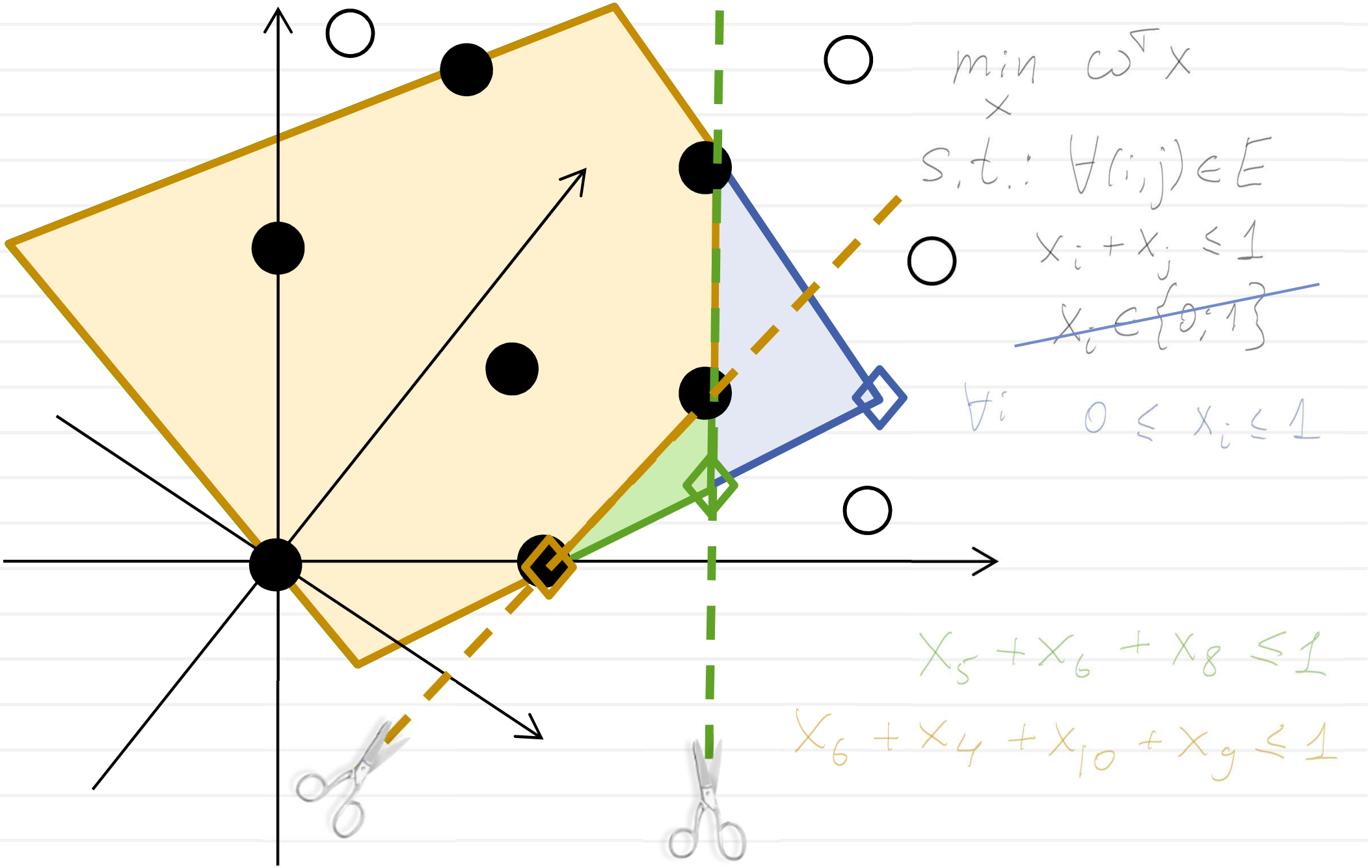
$$\forall i \quad x_i \in \{0; 1\}$$

Generally, any ***clique*** provides a *cutting plane*

$$x_5 + x_6 + x_8 \leq 1 \qquad x_6 + x_4 + x_{10} \leq 1$$

$$x_6 + x_4 + x_{10} + x_9 \leq 1 \qquad x_4 + x_9 + x_{10} \leq 1$$

What has happened



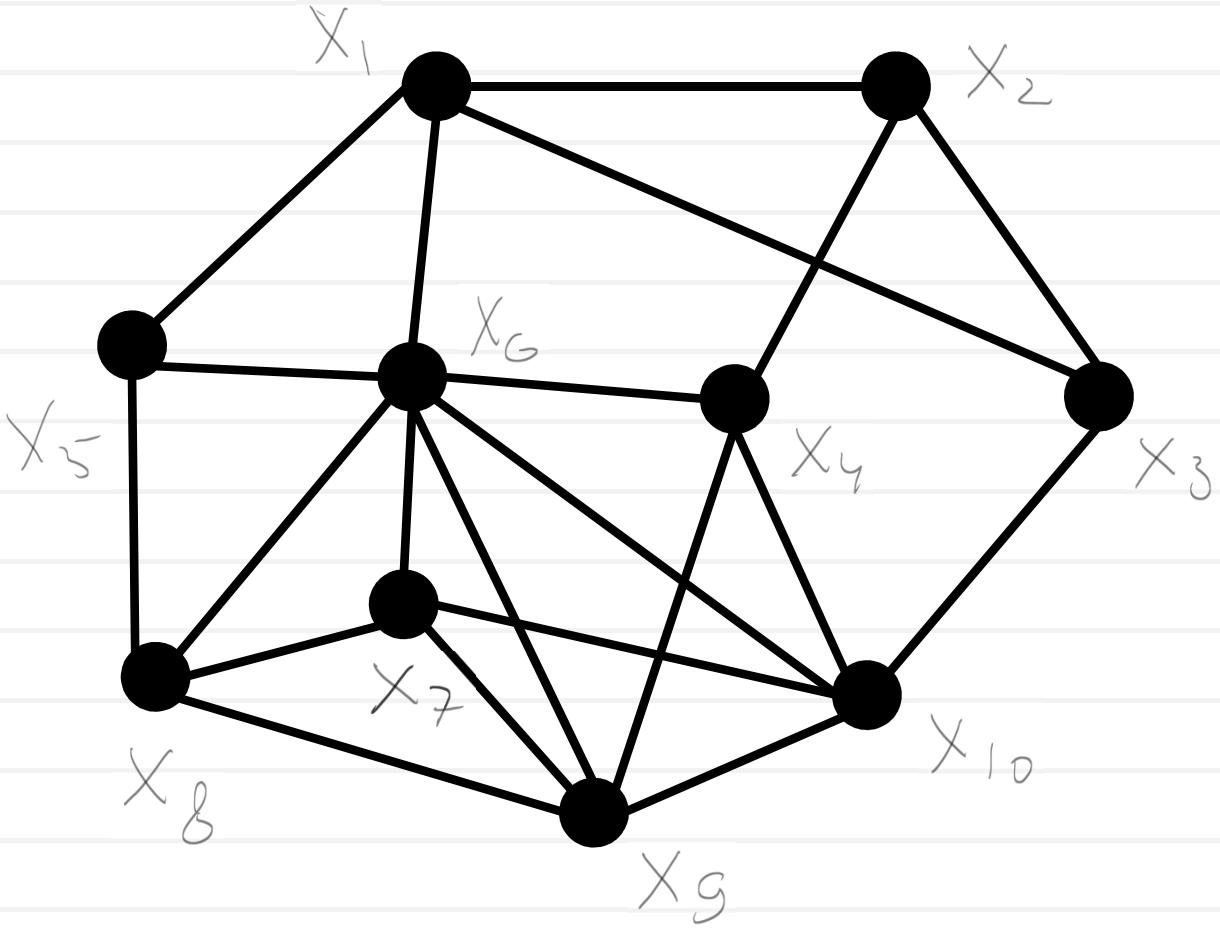
Cutting plane algorithm

$[p, D_{ILP}]$ = Formulate problem as an ILP
[p,A,b,C,d] = LPRelaxation(ILP)
x = SolveLP(p,A,b,C,d)

while x not integer
    $[A' \; b']$ = FindSeparatingCut(s) (x, D_{ILP})
    A = [A;A'], b = [b,b']
    x = SolveLP(p,A,b,C,d)
end

Separation oracle for MWIS

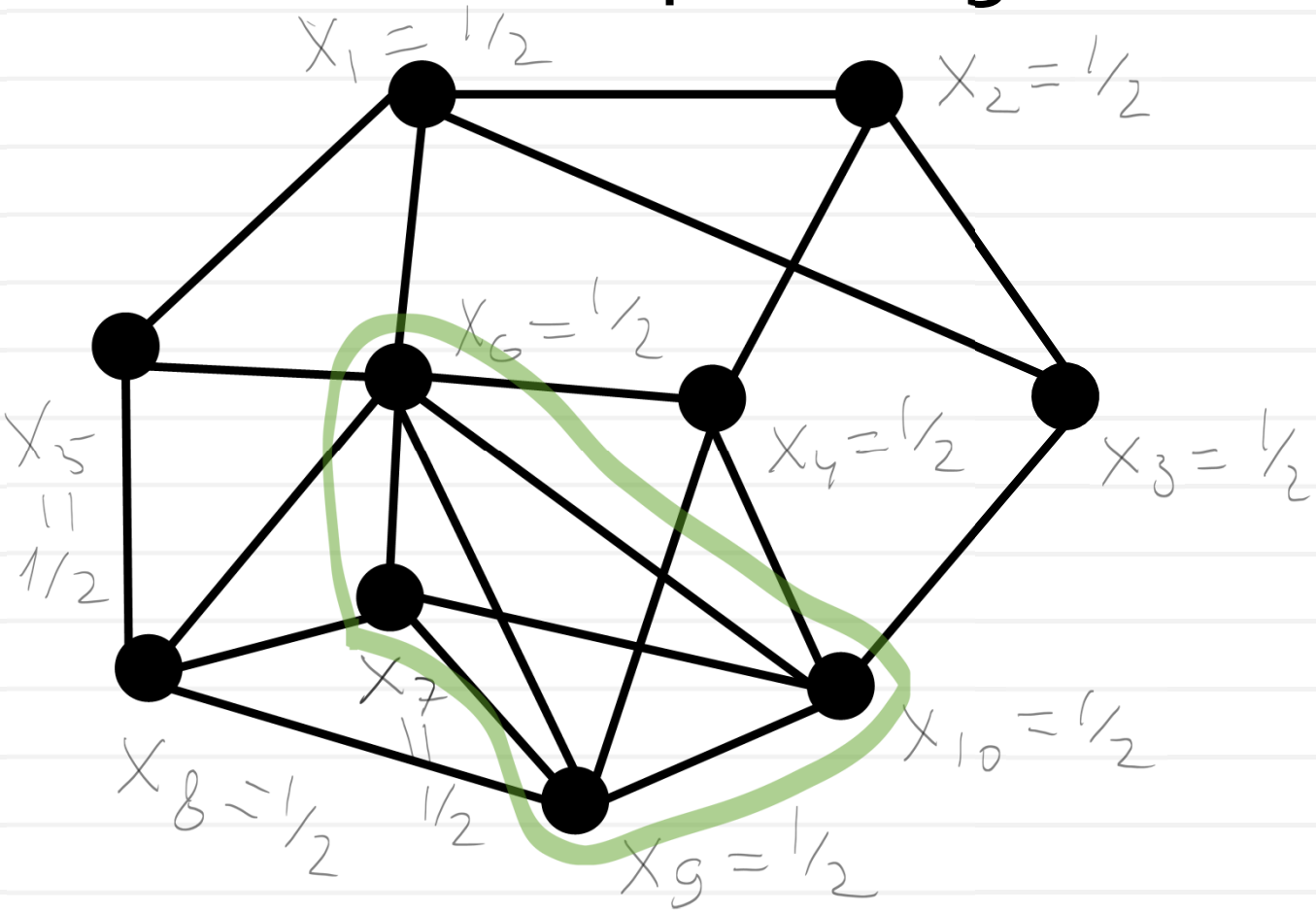
[A' b'] = FindSeparatingCut(s) (x, D_{ILP})



- Finding a clique with a sum > 1
- Can be done e.g. greedily

Separation oracle for MWIS

$[A' \; b'] = \text{FindSeparatingCut(s)}\,(x, D_{ILP})$



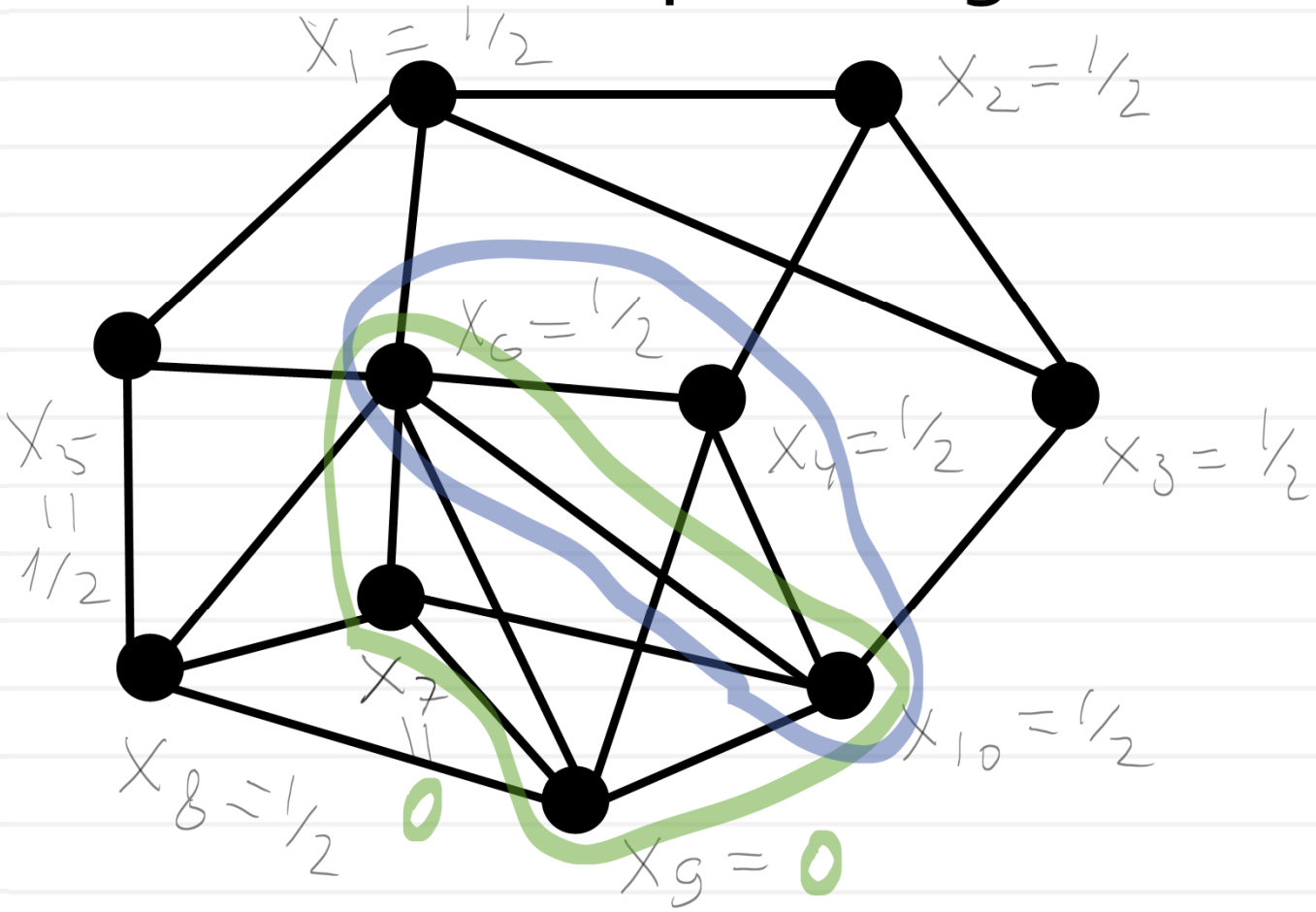
$$\min_x \; \omega^T x$$

$$s.t.: \; \forall (i,j) \in E \quad x_i + x_j \le 1$$

$$x_6 + x_7 + x_9 + x_{10} \le 1$$

Separation for MWIS

$[A' \; b'] = \text{FindSeparatingCut(s)}\,(x, D_{ILP})$



$$\min_{x} \; \omega^T x$$

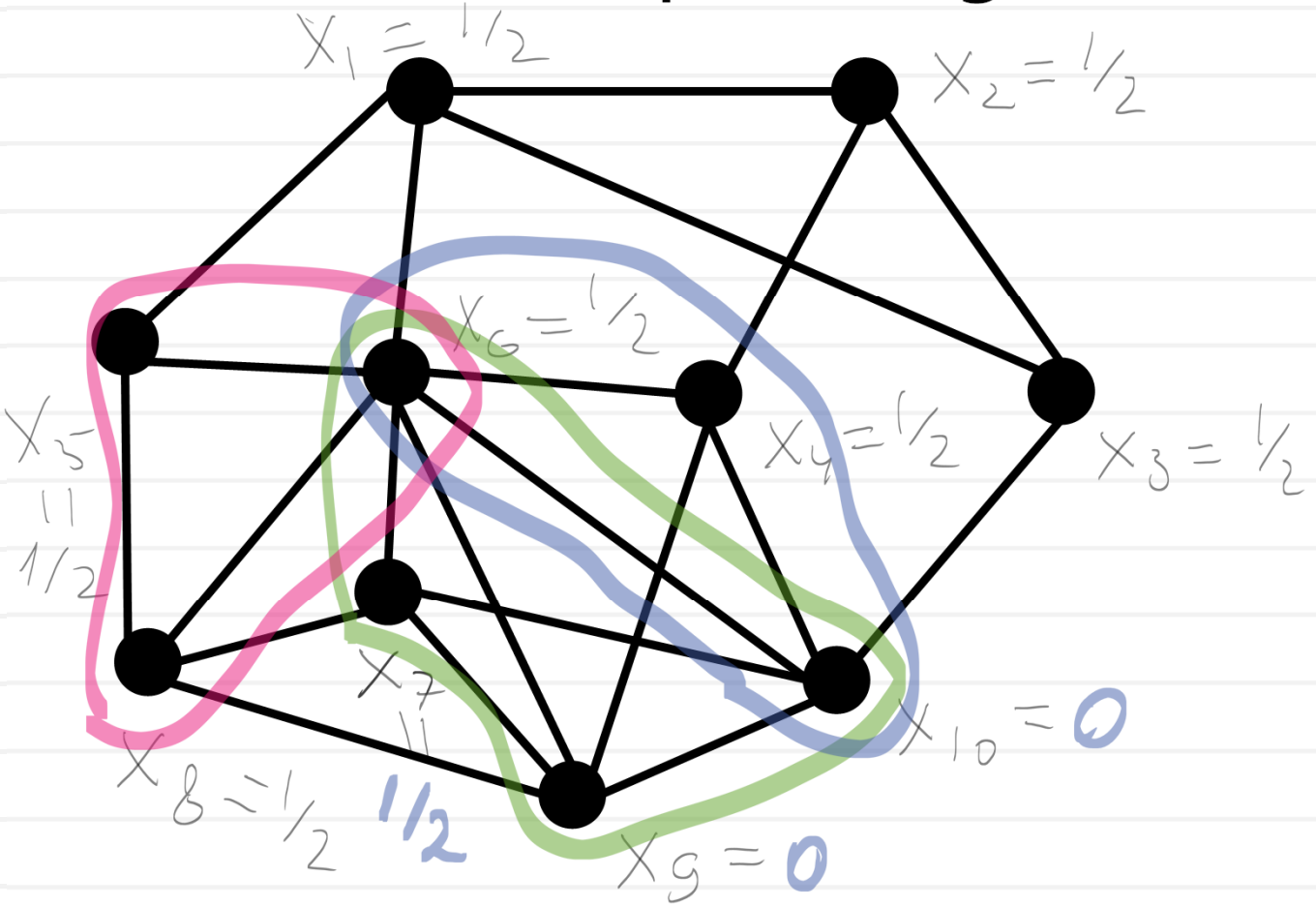
$$s.t.: \; \forall (i,j) \in E \quad x_i + x_j \leq 1$$

$$x_6 + x_7 + x_9 + x_{10} \leq 1$$

$$x_6 + x_4 + x_{10} \leq 1$$

Separation for MWIS

$[A' \; b'] = \text{FindSeparatingCut(s)}\,(x, D_{ILP})$



$$\min_x \; c^T x$$

$$s.t.: \; \forall (i,j) \in E \quad x_i + x_j \le 1$$

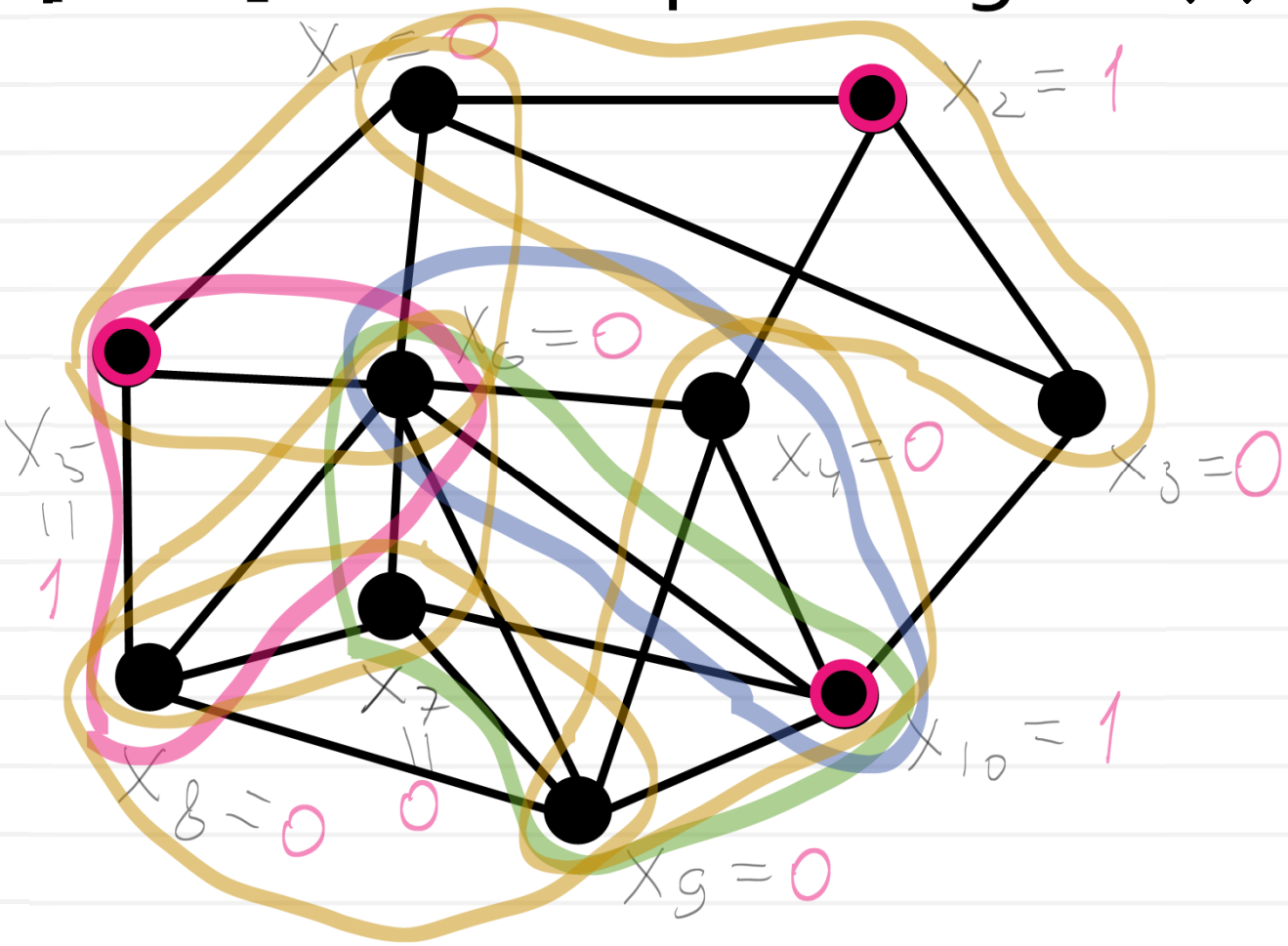
$$x_6 + x_7 + x_9 + x_{10} \le 1$$

$$x_6 + x_4 + x_{10} \le 1$$

$$x_5 + x_6 + x_8 \le 1$$

Separation for MWIS

$[A' \; b'] = \text{FindSeparatingCut(s)}\,(x, D_{ILP})$



$$\min_x \; c^T x$$

$$s.t.: \; \forall (i,j) \in E \quad x_i + x_j \leq 1$$

$$x_6 + x_7 + x_9 + x_{10} \leq 1$$

$$x_6 + x_4 + x_{10} \leq 1$$

$$x_5 + x_6 + x_8 \leq 1$$

Orange cliques represent non-trivial constraints that were not needed

Travelling salesman problem

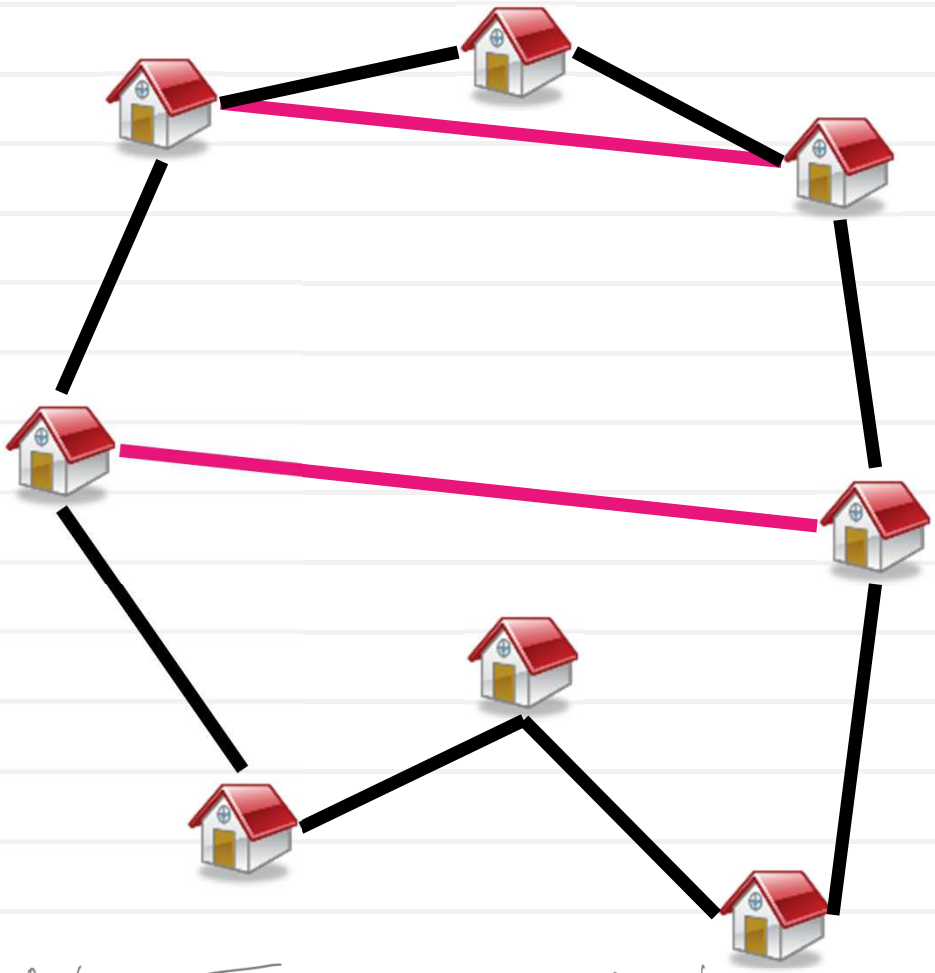
$$\min \omega_{ij} x_{ij}$$

$$s.t.: x_{ij} \in \{0;1\}$$

$$\forall i: \sum_j x_{ij} = 2$$

$$\sum_{i,j} x_{ij} = N$$

$$\forall C \subset \{1 \dots N\} \quad 0 < |C| < N: \sum_{\substack{i \in C \\ j \in C}} x_{ij} \leq |C| - 1$$



Exponentially many constraints!

(assuming that x_{ij} and x_{ji} are the same)

Travelling salesman problem

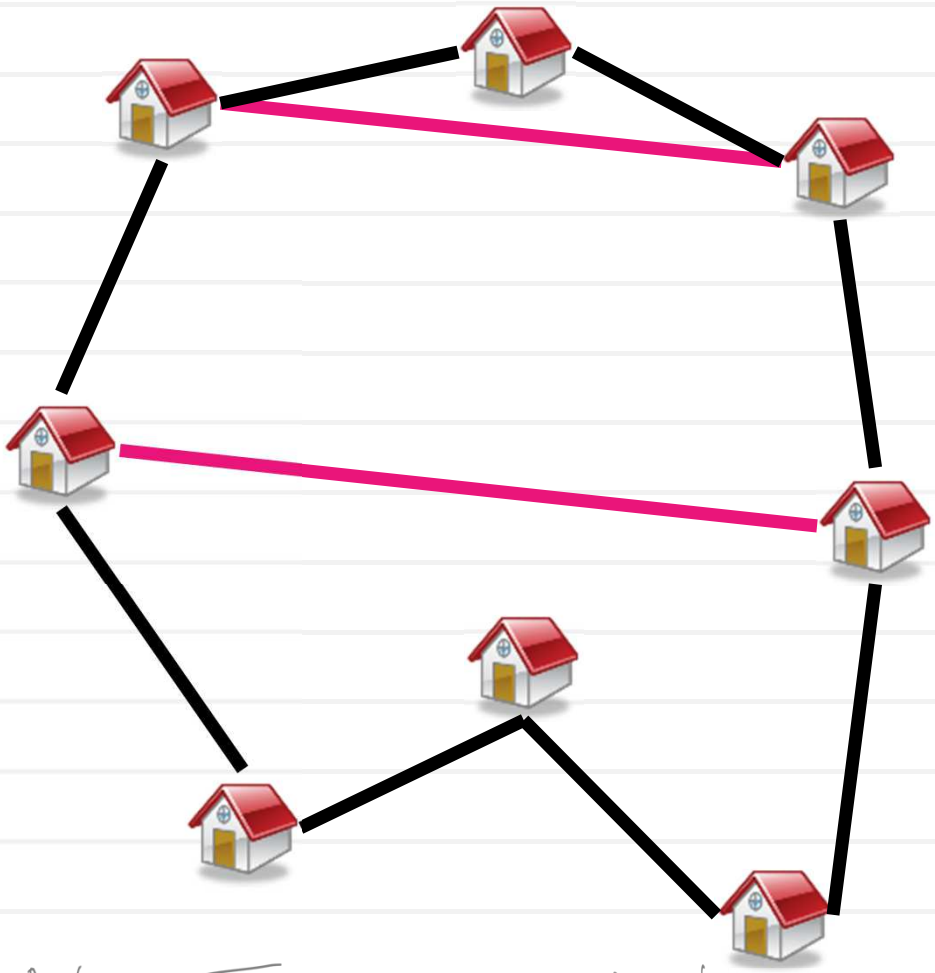
$$\min \omega_{ij} x_{ij}$$

$$s.t.: x_{ij} \in \{0;1\}$$

$$\forall i: \sum_j x_{ij} = 2$$

$$\sum_{i,j} x_{ij} = N$$

$$\forall C \subset \{1 \dots N\} \quad 0 < |C| < N: \sum_{\substack{i \in C \\ j \in C}} x_{ij} \leq |C| - 1$$



Exponentially many constraints!

(assuming that x_{ij} and x_{ji} are the same)

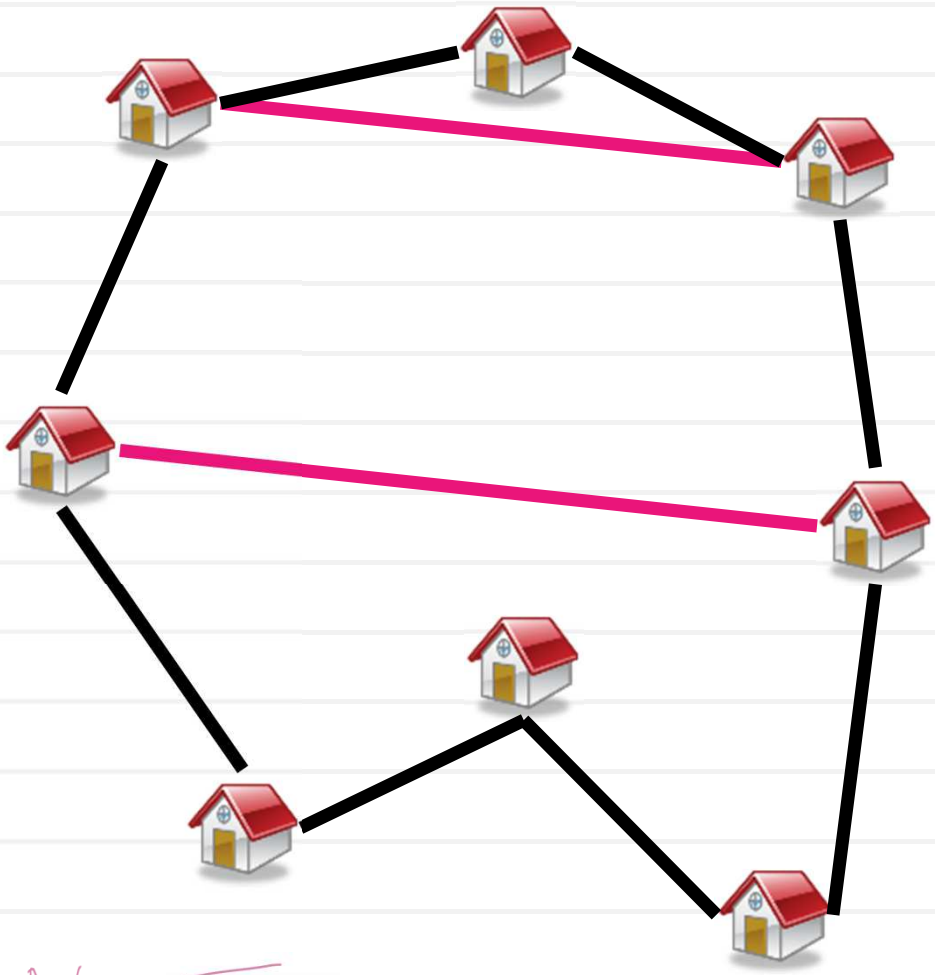
Travelling salesman problem

$$\min \omega_{ij} x_{ij}$$

$$s.t.: x_{ij} \in \{0;1\}$$

$$\forall i: \sum_j x_{ij} = 2$$

$$\sum_{i,j} x_{ij} = N$$



$$\forall C \subset \{1 \dots N\} \quad 0 < |C| < N: \sum_{\substack{i \in C \\ j \notin C}} x_{ij} \geq 2$$

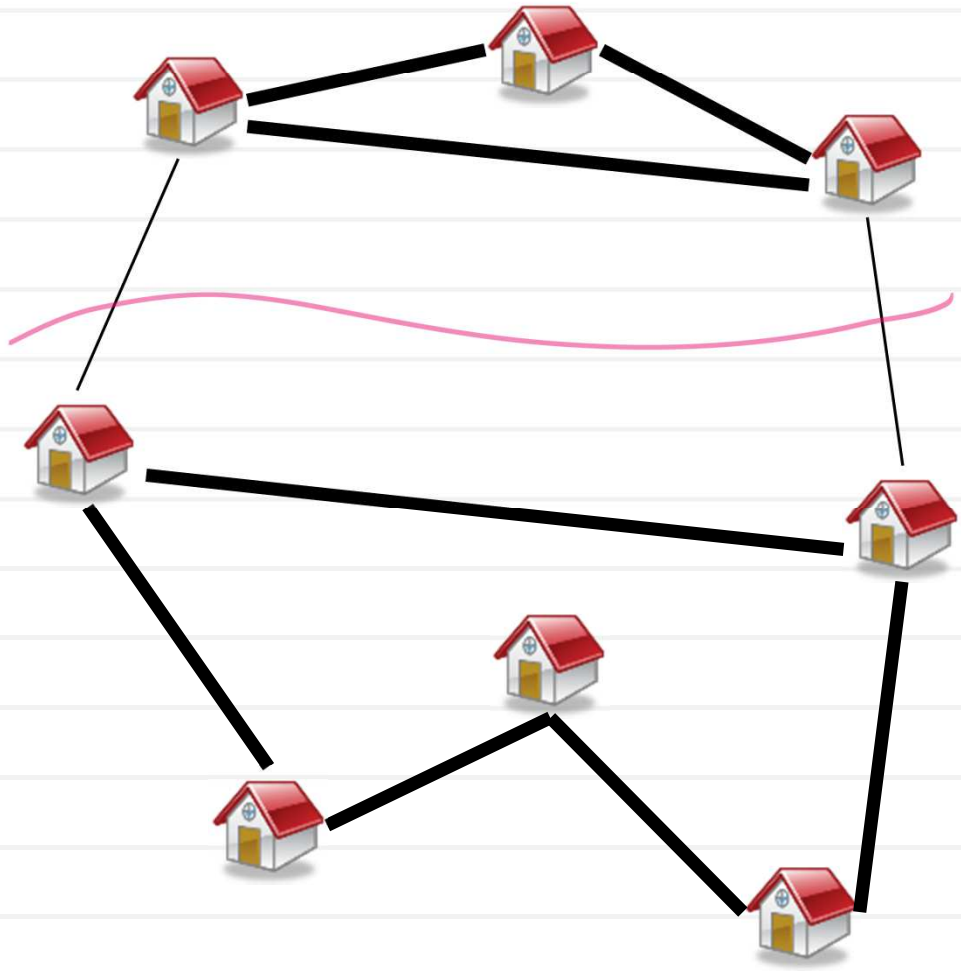
Exponentially many constraints!

(assuming that x_{ij} and x_{ji} are the same)

Finding Maximally Violated Constraint

$$\forall C \subset \{1 \dots N\} \quad 0 < |C| < N : \sum_{\substack{i \in C \\ j \notin C}} x_{ij} \geq 2$$

$$\min_{C} \sum_{\substack{i \in C \\ j \notin C}} x_{ij}$$



- (NP-)hard for arbitrary *xij*
- For non-negative *xij* polynomial algorithms exist

TSP vs MWIS

- In MWIS the integer program has a "small" number of constraints, we add extra constraint to tighten the LP relaxation
- In TSP, the integer program and its LP relaxation already has a huge number of constraints
- Still, we can use the cutting plane algorithm to solve the LP relaxation (*delayed constraint generation*)

Delayed constraint generation

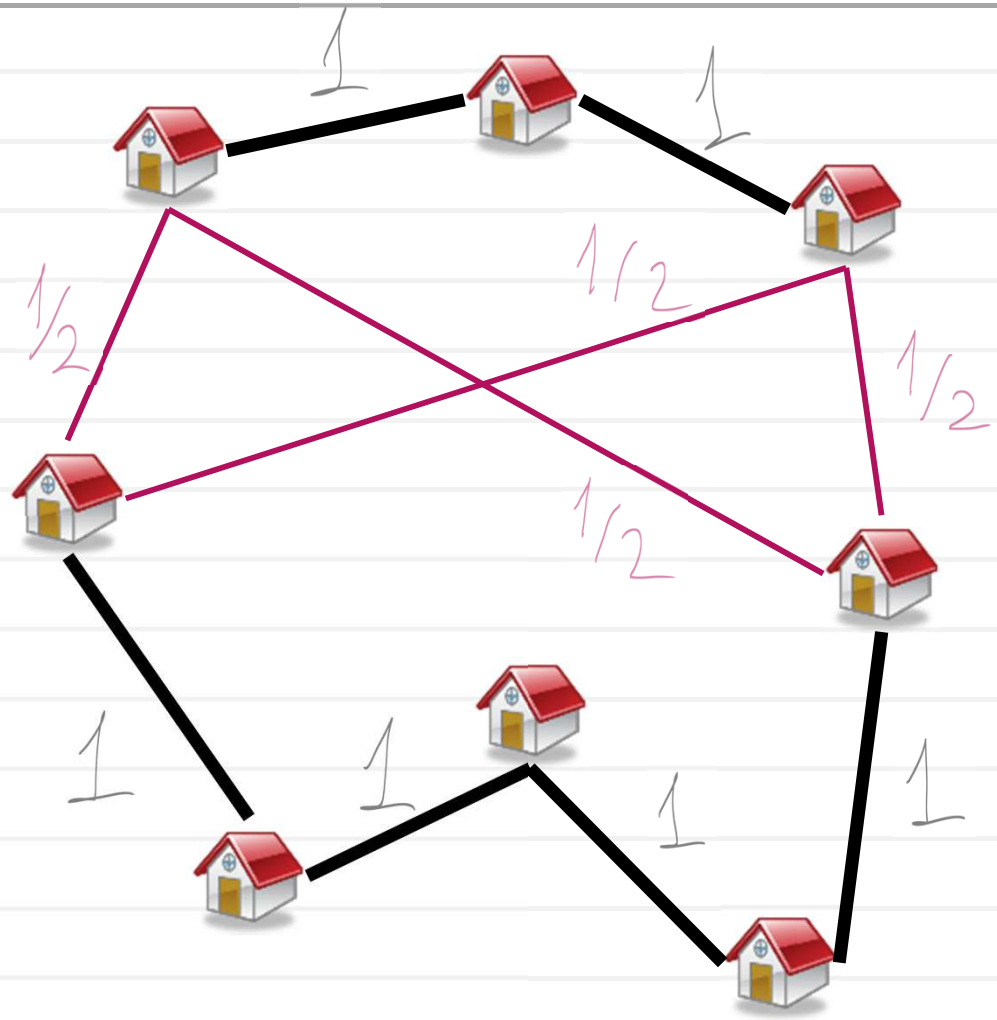
This pseudocode solves the LP relaxation of the TSP (not the TSP itself):

```
[p,A,b,C,d] = LPRelaxation(TSP(G),
                           some subtour constraints)
x = SolveLP(p,A,b,C,d)
while x not integer && x not a tour
        Set = MinCut(x, G)
        if cut(Set,x) ≥ 2
                break
        [A,b] = AddCutSetConstraint(A,b);
        x = SolveLP(p,A,b,C,d)
end
```

To solve the TSP, we then need to add further cutting planes that will tighten the relaxation (e.g. *comb* cuts)

LP relaxation

Even with all constraints in, the LP relaxation might give a fractional solution:



$$0 \le x_{ij} \le 1$$

$$\forall i: \sum_j x_{ij} = 2$$

$$\sum_{i,j} x_{ij} = N$$

$$\forall C \subset \{1 \dots N\}\ 0 < |C| < N: \sum_{\substack{i \in C \\ j \notin C}} x_{ij} \ge 2$$

Types of the tightening cuts

Problem specific (work best!):

- Clique cuts for MWIS
- (Subtour elimination constraints for TSP)
- *Comb* cuts for TSP
- ….

Generic cuts:

- Gomory cuts
- Cover cuts
- ….

Cover cut

Generic family of cutting planes, that can be derived directly from the problem

$$1x_1 + 5x_2 + 4x_3 + 3x_4 \leq 7 \quad (*)$$

$$x_i \in \{0;1\}$$

$5+4>7$

$5+3>7$

$$x_2 + x_3 \leq 1$$

$$x_2 + x_4 \leq 1$$

Furthermore: $x_1 + x_3 + x_4 \leq 2$

$\{2;3\}, \{2;4\}, \{1;3;4\}$ are *covers* w.r.t. (*)

Cover cut

Given original inequality:

$$1x_1 + 5x_2 + 4x_3 + 3x_4 \leq 7$$

Any $\alpha_1 \alpha_2 \alpha_3 \alpha_4$: $\alpha_1 + 5\alpha_2 + 4\alpha_3 + 3\alpha_4 \geq 8$

$\alpha_i \in \{0; 1\}$

α is a cover

Gives a cutting plane:

not all variables are on

$$\alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \alpha_4 x_4 \leq \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - 1$$

$$\alpha_1 (x_1 - 1) + \alpha_2 (x_2 - 1) + \alpha_3 (x_3 - 1) + \alpha_4 (x_4 - 1) \leq -1$$

Cover cut

Given original inequality:

$$1x_1 + 5x_2 + 4x_3 + 3x_4 \leq 7$$

Any $\alpha_1 \alpha_2 \alpha_3 \alpha_4 : \alpha_1 + 5\alpha_2 + 4\alpha_3 + 3\alpha_4 \geq 8$

$\alpha_i \in \{0; 1\}$ $\quad \beta_i = 1 - \alpha_i \in \{0; 1\}$

$$1 - \beta_1 + 5(1 - \beta_2) + 4(1 - \beta_3) + 3(1 - \beta_4) \geq 8$$

$$\beta_1 + 5\beta_2 + 4\beta_3 + 3\beta_4 \leq 13 - 8 = 5$$

Gives a cutting plane:

$$\alpha_1(x_1 - 1) + \alpha_2(x_2 - 1) + \alpha_3(x_3 - 1) + \alpha_4(x_4 - 1) \leq -1$$

$$(1 - x_1)\beta_1 + (1 - x_2)\beta_2 + (1 - x_3)\beta_3 + (1 - x_4)\beta_4 \leq 3 - \sum x_i$$

Separation oracle for cover cuts

If $\beta_1 + 5\beta_2 + 4\beta_3 + 3\beta_4 \leq 5$ Then:

$$(1-x_1)\beta_1 + (1-x_2)\beta_2 + (1-x_3)\beta_3 + (1-x_4)\beta_4 \leq 3 - \sum x_i$$

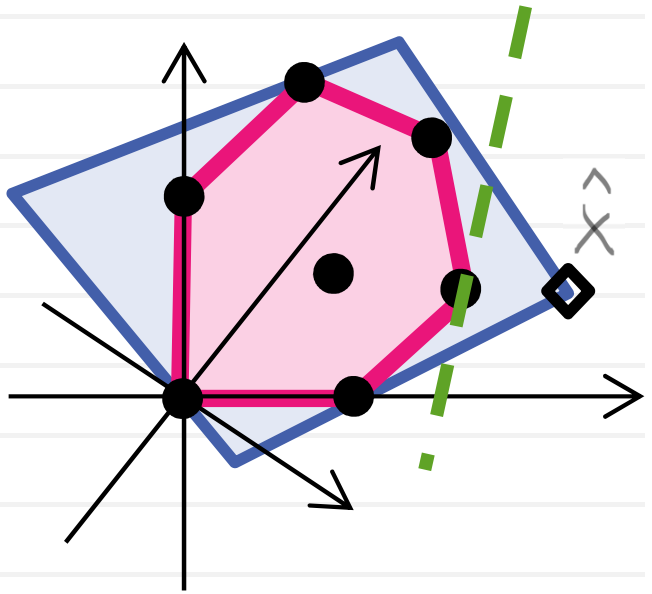
Let $\hat{x}$ be the current solution. Let us find the "deepest cut" that separates it from the bounding box:

$$\max_{\beta} (1-\hat{x}_1)\beta_1 + (1-\hat{x}_2)\beta_2 + (1-\hat{x}_3)\beta_3 + (1-\hat{x}_4)\beta_4$$

$$s.t.: \beta_1 + 5\beta_2 + 4\beta_3 + 3\beta_4 \leq 5$$

$$\beta_i \in \{0;1\}$$

If objective $\geq 3 - \sum \hat{x}_i$ then we have cut the current solution away



Gomory cut example

$$1.5\,x_1 + 4.3\,x_2 + 2.2\,x_3 + 7.5\,x_4 = 111.3$$

$$x_i \in \mathbb{Z}_+$$

$$1x_1 + 4x_2 + 2x_3 + 7x_4 +$$

$$(0.5\,x_1 + 0.3\,x_2 + 0.2\,x_3 + 0.5\,x_4) = 111 + 0.3$$

The red part cannot be less then 0.3, or otherwise the inequality cannot hold.

Gomory cut (in this case):

$$0.5\,x_1 + 0.3\,x_2 + 0.2\,x_3 + 0.5\,x_4 \geq 0.3$$

Gomory cuts: general formula

$$\alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \ldots + \alpha_n x_n = \beta$$

$$x_i \in \mathbb{Z}_+$$

$$\lfloor \alpha_1 \rfloor x_1 + \lfloor \alpha_2 \rfloor x_2 + \ldots \lfloor \alpha_n \rfloor x_n +$$

$$+ (\alpha_1 - \lfloor \alpha_1 \rfloor) x_1 + \ldots + (\alpha_n - \lfloor \alpha_n \rfloor) x_n =$$

$$= \lfloor \beta \rfloor + (\beta - \lfloor \beta \rfloor)$$

General Gomory cut:

$$(\alpha_1 - \lfloor \alpha_1 \rfloor) x_1 + \ldots + (\alpha_n - \lfloor \alpha_n \rfloor) x_n \geq \beta - \lfloor \beta \rfloor$$

Important: the equality can be a linear combination of initial equalities

Cutting plane algorithm

```
function x = SolveILPCuttingPlane(p, D_ILP)
[p,A,b,C,d] = LPRelaxation(ILP)
x = SolveLP(p,A,b,C,d)
while x not integer
        [A' b'] = FindSeparatingCut(s) (x, D_ILP)
        if [A' b'] empty
                break
        A = [A;A'],  b = [b,b']
        x = SolveLP(p,A,b,C,d)
end
```

Branch and Bound

Initialize the list *L* to the root node
incumbent_value = +∞
Until *L* is empty or time is out
    Pick the active node from *L*
    Pick the variable to branch
    For each of the two children nodes
        [x obj] = Solve LP (node)
        If *obj* ≥ *incumbent_obj* **then continue;**
        If *x* is integer **then**
            incumbent = *x*; *incumbent_value* = *obj*;
        Else add node to *L*
    end
end

Branch-and-Cut

Initialize the list *L* to the root node
incumbent_value = +∞
Until *L* is empty or time is out
    Pick the active node from *L*
    Pick the variable to branch
    For each of the two children nodes
        [x obj] = **SolveILPCuttingPlane**(node)
        If *obj* ≥ *incumbent_obj* **then continue;**
        If *x* is integer **then**
            incumbent = *x*; *incumbent_value* = *obj*;
        Else add node to *L*
    end
end