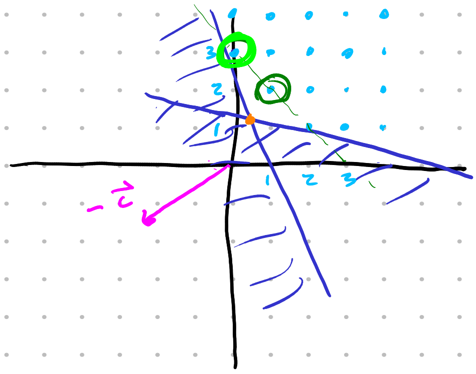
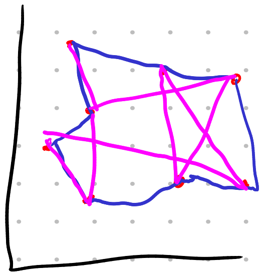
Discrete Optimization

Integer Linear program (ILP)

$$\begin{aligned} \underset{\vec{x}}{\text{minimize}} \quad & \vec{c}^T\vec{x} \\ \text{subject to} \quad & A\vec{x} \le b \\ & \vec{x} \ge 0 \\ & \vec{x} \in \mathbb{Z}^n \end{aligned}$$



Travelling Salesman Problem (TSP)



Find shortest "tour" (path passing through all points)

Formulate as ILP

$$\begin{aligned} \underset{\vec{x}\,\vec{u}}{\text{minimize}} \quad & \sum_{ij} d_{ij} x_{ij} \\ \text{subject to} \quad & x_{ij} \in \{0,1\} \\ & \sum_j x_{ij} = 1 \quad \forall i \quad \text{"exit"} \\ & \sum_i x_{ij} = 1 \quad \forall j \quad \text{"entry"} \\ & u_i \in \{2, \dots, n\} \\ & u_i - u_j + 1 \le (n-1)(1 - x_{ij}) \quad \forall i,j \end{aligned}$$

$x_{ij} \in \{0,1\}$

$= \begin{cases} 1 & \text{if path contains } i \to j \\ 0 & \text{o.w.} \end{cases}$

$\sum_{ji} x_{ij} = n$

$\forall i$

$\forall i \quad x_{ii} = 0$

Miller-Tucker-Zemlin

want if $x_{ij} = 1$
$\underline{u_j - u_i = 1}$

if $x_{ij} = 1$

$u_i - u_j + 1 \le 0$

$u_i - u_j \le -1$

$u_j - u_i \ge 1$

when applied to all i,j forces

$u_j - u_i = 1$

if $x_{ij} = 0$

$u_i - u_j + 1 \le n - 1$

worst case

$u_i = n, \quad u_j = 2$

does not constrain u

MILP "Mixed Integer LP"

$$\underset{\vec{x}}{\text{minimize}} \quad \vec{c}^T\vec{x}$$

$$A\vec{x} \le \vec{b} \quad \leftarrow \quad A\vec{x} = \vec{b}$$

$$\vec{x} \ge 0$$

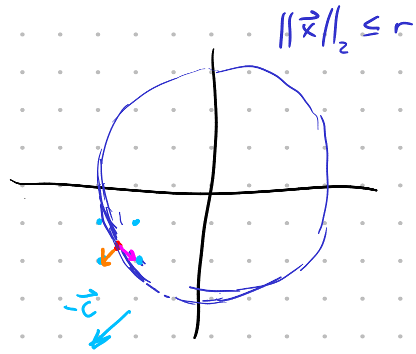
$$\vec{x}_D \in \mathbb{Z}^n$$

$\boxed{\text{Rounding}}$

relax $\vec{x}_D \in \mathbb{Z}^n$ constraint to $\vec{x}_D \in \mathbb{R}^n$

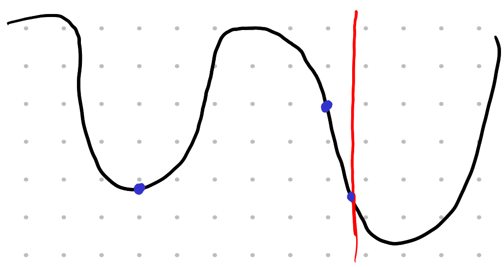
solve

round to nearest integer $\vec{x}_D$



Problems

1) result might be infeasible
2) nearest feasible integral solution might be much worse than optimal



Sometimes there are gaurantees on how close the solution is

If $A \in \mathbb{Z}^{n \times m}$

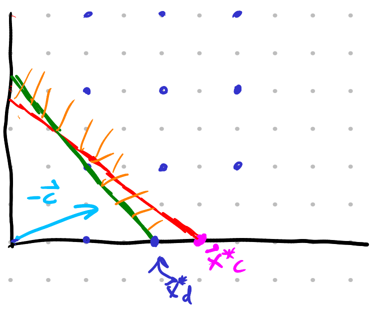
$$\| x_d^* - x_c^* \| \le n \times \text{max absolute value of determinants of submatrices of } A$$

x_c^*: relaxed solution

x_d^*: optimal discrete solution

S.O.T.A. "branch + cut"

Cutting Plane



Introduce new constraint that

1) excludes x_c^*
2) includes all other discrete solutions

Partition $\vec{x}_c^*$ into (B, V) $\quad \vec{x}_B^* \quad \vec{x}_V^* = 0$

$\vec{x}_B^*$: all of the non-integral components

For each $b \in B$ where x_b^* is non-integral introduce a new constraint

$$\underbrace{x_b^* - \lfloor x_b^* \rfloor}_{\geq 0} - \underbrace{\sum_{v \in V} \left(\bar{A}_{bv} - \lfloor \bar{A}_{bv} \rfloor\right) x_v}_{0} \leq 0 \qquad \bar{A}_{bv} = A_B^{-1} A_V$$

$$x_t + \sum_{v \in V} \left(\lfloor \bar{A}_{bv} \rfloor - \bar{A}_{bv}\right) x_v = \lfloor x_b^* \rfloor - x_b^*$$

$$x_t \in \mathbb{N}$$

Consider the integer program:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & 2x_1 + x_2 + 3x_3 \\ \text{subject to} \quad & \begin{bmatrix} 0.5 & -0.5 & 1.0 \\ 2.0 & 0.5 & -1.5 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 2.5 \\ -1.5 \end{bmatrix} \\ & \mathbf{x} \geq \mathbf{0} \quad \mathbf{x} \in \mathbb{Z}^3 \end{aligned}$$

x_c^* $V = \{2\}$

The relaxed solution is $\mathbf{x}^* \approx [0.818, 0, 2.091]$, yielding:

$B = \{1, 3\}$

$$\mathbf{A}_{\mathcal{B}} = \begin{bmatrix} 0.5 & 1 \\ 2 & -1.5 \end{bmatrix} \quad \mathbf{A}_{\mathcal{V}} = \begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix} \quad \bar{\mathbf{A}} = \begin{bmatrix} -0.091 \\ -0.455 \end{bmatrix}$$

From equation (19.7), the constraint for x_1 with slack variable x_4 is:

$$\begin{aligned} x_4 + (\lfloor -0.091 \rfloor - (-0.091))x_2 &= \lfloor 0.818 \rfloor - 0.818 \\ x_4 - 0.909x_2 &= -0.818 \end{aligned}$$

The constraint for x_3 with slack variable x_5 is:

$$\begin{aligned} x_5 + (\lfloor -0.455 \rfloor - (-0.455))x_2 &= \lfloor 2.091 \rfloor - 2.091 \\ x_5 - 0.545x_2 &= -0.091 \end{aligned}$$

The modified integer program has:

$$\mathbf{A} = \begin{bmatrix} 0.5 & -0.5 & 1 & 0 & 0 \\ 2 & 0.5 & -1.5 & 0 & 0 \\ 0 & -0.909 & 0 & 1 & 0 \\ 0 & -0.545 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 2.5 \\ -1.5 \\ -0.818 \\ -0.091 \end{bmatrix} \quad \mathbf{c} = \begin{bmatrix} 2 \\ 1 \\ 3 \\ 0 \\ 0 \end{bmatrix}$$

Solving the modified LP, we get $\mathbf{x}_c^* \approx [0.9, 0.9, 2.5, 0.0, 0.4]$. Since this point is not integral, we repeat the procedure with constraints:

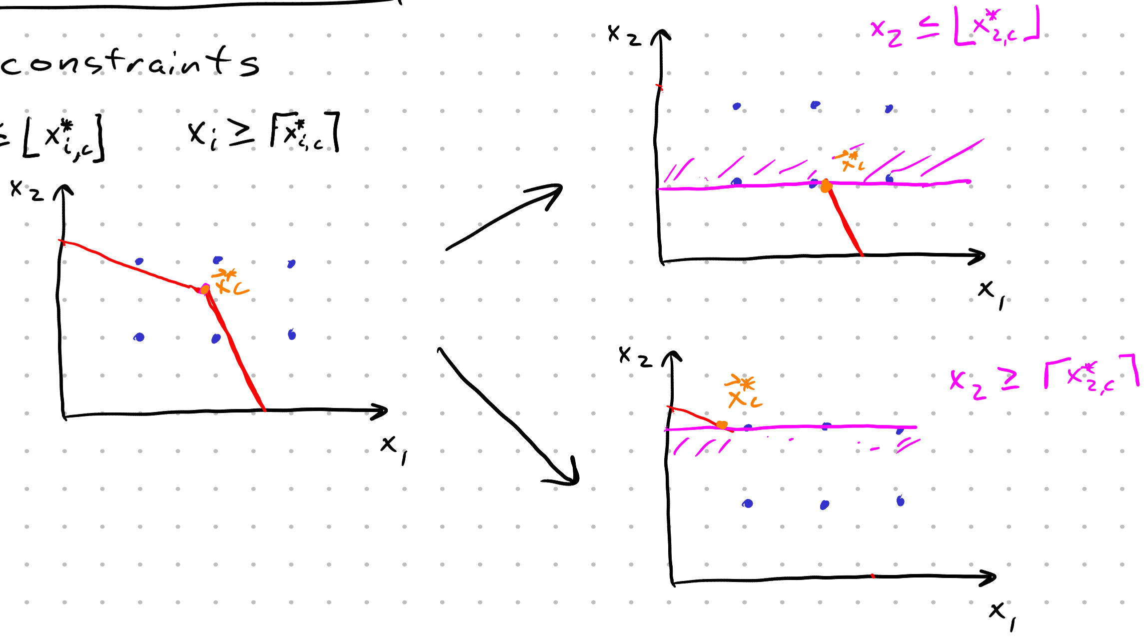
$$\begin{aligned} x_6 - 0.9x_4 &= -0.9 \qquad & x_7 - 0.9x_4 &= -0.9 \\ x_8 - 0.5x_4 &= -0.5 \qquad & x_9 - 0.4x_4 &= -0.4 \end{aligned}$$

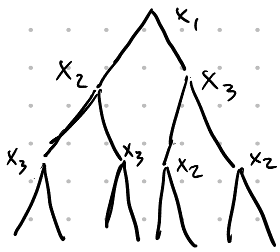
and solve a third LP to obtain: $\mathbf{x}^* = [1, 2, 3, 1, 1, 0, 0, 0, 0]$ with a final solution of $\mathbf{x}_i^* = [1, 2, 3]$.

Branch and Bound

add constraints

$x_i \le \lfloor x^*_{i,c} \rfloor \qquad x_i \ge \lceil x^*_{i,c} \rceil$





or Priority Queue ranked by lower bound

Which i to branch on next

Common heuristic $\quad \vec{x}^*_c = [0.1, 0.4, 0.9]$

```
function minimize_lp_and_y(LP)
    try
        x = minimize_lp(LP)
        return (x, x⋅LP.c)
    catch
        return (fill(NaN, length(LP.c)), Inf)
    end
end
function branch_and_bound(MIP)
    LP = relax(MIP)
    x, y = minimize_lp_and_y(LP)
    n = length(x)
    x_best, y_best, Q = deepcopy(x), Inf, PriorityQueue()
    enqueue!(Q, (LP,x,y), y)
    while !isempty(Q)
        LP, x, y = dequeue!(Q)
        if any(isnan.(x)) || all(isint(x[i]) for i in MIP.D)
            if y < y_best
                x_best, y_best = x[1:n], y
            end
        else
            i = argmax([abs(x[i] - round(x[i])) for i in MIP.D])
            # x_i ≤ floor(x_i)
            A, b, c = LP.A, LP.b, LP.c
            A2 = [A zeros(size(A,1));
                  [j==i for j in 1:size(A,2)]' 1]
            b2, c2 = vcat(b, floor(x[i])), vcat(c, 0)
            LP2 = LinearProgram(A2,b2,c2)
            x2, y2 = minimize_lp_and_y(LP2)
            if y2 ≤ y_best
                enqueue!(Q, (LP2,x2,y2), y2)
            end
            # x_i ≥ ceil(x_i)
            A2 = [A zeros(size(A,1));
                  [j==i for j in 1:size(A,2)]' -1]
            b2, c2 = vcat(b, ceil(x[i])), vcat(c, 0)
            LP2 = LinearProgram(A2,b2,c2)
            x2, y2 = minimize_lp_and_y(LP2)
            if y2 ≤ y_best
                enqueue!(Q, (LP2,x2,y2), y2)
            end
        end
    end
    return x_best
end
```

Shortest Path

travel from node a to b in minimum distance

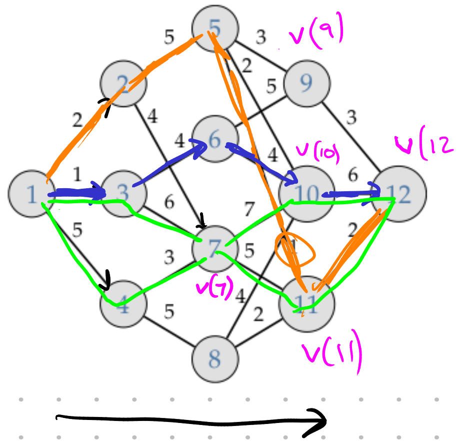
$$\text{minimize} \quad \sum_{ij} c_{ij} x_{ij}$$

$$\text{subject to} \quad x_{ij} \in \{0, 1\}$$

$$\sum_{j \in \text{children}(1)} x_{1j} = 1 \qquad \text{"start"}$$

$$\sum_{i \in \text{parents}(n)} x_{in} = 1 \qquad \text{"end"}$$

$$\sum_{i \in \text{parents}(k)} x_{ik} - \sum_{j \in \text{children}(k)} x_{kj} = 0$$



$$\underset{x_1, \dots x_n}{\text{minimize}} \left(c(s_1, x_1) + c(s_2, x_2) \dots c(s_n, x_n) \right)$$

$$s_{k+1} = f(s_k, x_k)$$

$$c(s_k, x_k)$$

Greedy Algorithm (not always optimal)

$$\underset{x_i}{\text{minimize}} \; c(s_i, x_i)$$

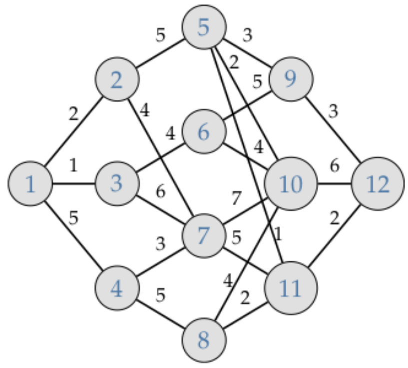
Bellman's Principle of Optimality

"Every sub-path of an optimal path is optimal"

$$v(s_i) = \underset{x_i \dots x_n}{\text{minimize}} \left(c(s_i, x_i) + \underbrace{c(s_{i+1}, x_{i+1}), \dots c(s_n, x_n)} \right)$$

$$= \underset{x_i}{\text{minimize}} \left(c(s_i, x_i) + v(s_{i+1}) \right)$$

$$= \underset{x_i}{\text{minimize}} \left(c(s_i, x_i) + v(f(s_i, x_i)) \right)$$


5
5
3
2
2
9
5
2
4
4
6
4
3
3
1
1
10
6
12
6
7
7
2
5
5
1
3
4
4
11
5
2
8