Why linear program? Answered!

The objective function must be a linear function of the decision variables and the constraints must be a linear inequality

This is a linear function

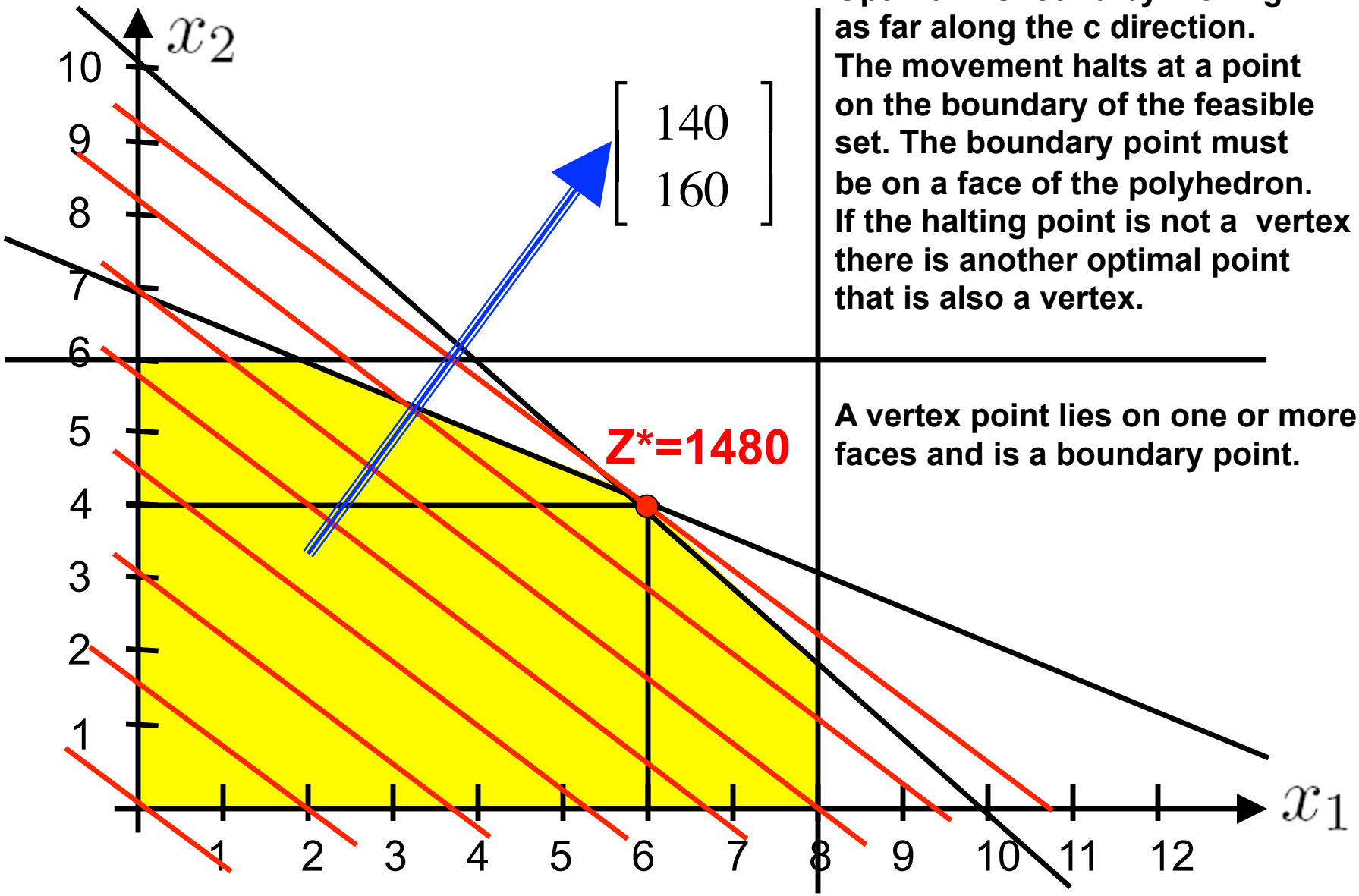
$$\min \ \langle c, x \rangle$$

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

Thus Linear Inequality.

EVERY LP HAS AN OPTIMAL SOLUTION AT A VERTEX OF ITS FEASIBLE POLYTOPE

All perpendicular to c = [140 160]^T



Optimum is found by moving as far along the c direction. The movement halts at a point on the boundary of the feasible set. The boundary point must be on a face of the polyhedron. If the halting point is not a vertex there is another optimal point that is also a vertex.

A vertex point lies on one or more faces and is a boundary point.

THE SIMPLEX METHOD FOR SOLVING AN LP JUMPS CLEVERLY FROM VERTEX TO VERTEX

Different cost functions

*M*inimize Total Inventory Level: $\min x_1 + x_2 + x_3$

*M*inimize Worst Case Inventory Level:

$\min \max \{|x_1|, |x_2|, |x_3|\}$

*M*inimize Sum of Deviations from an Inventory Level:

$\min \{|x_1 - d| + |x_2 - d| + |x_3 - d|\}$

$s_i \geq 0$

https://en.wikipedia.org/wiki/Taxicab_geometry
https://math.stackexchange.com/questions/2589887/how-can-the-infinity-norm-minimization-problem-be-rewritten-as-a-linear-program
https://math.stackexchange.com/questions/1639716/how-can-l-1-norm-minimization-with-linear-equality-constraints-basis-pu

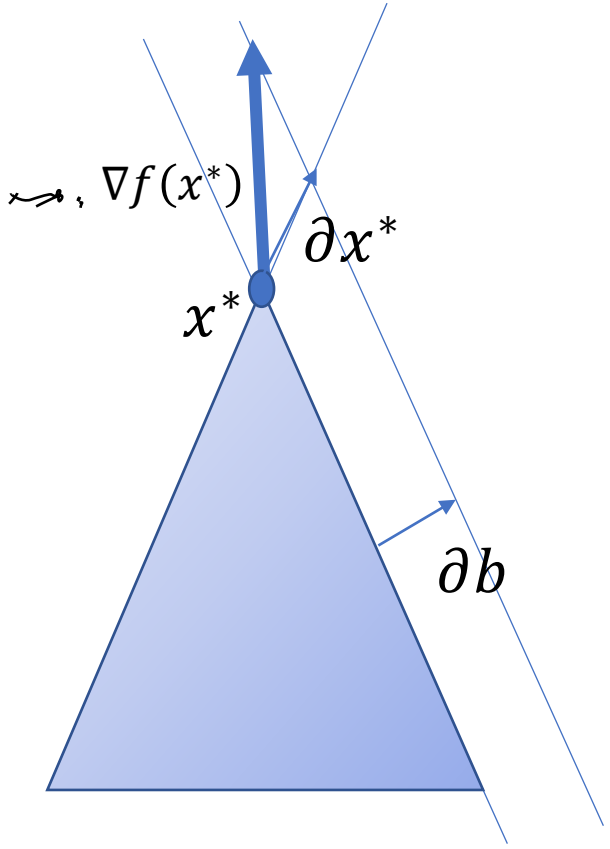
The shadow price of a constraint at the point of optimality is the value of its lagrange multiplier. By the KKT conditions the shadow price of an inactive constraint is zero.

$$\min f(x), \text{ s.t. } g_1(x) \leq b_1, g_2(x) \leq b_2, \ldots \ldots.$$

$$Shadow\ price\ for\ constraint\ 1 = p_1 = \left.\frac{\partial f}{\partial b_1}\right|_{x=x^*}$$

$$Shadow\ price\ for\ constraint\ 2 = p_2 = \left.\frac{\partial f}{\partial b_2}\right|_{x=x^*}$$

and so on.



The marginal improvement in the optimal cost for marginal relaxation of the constraint.

Relaxation: $x \geq -2\ constraint. x \geq -3\ is\ a\ relaxation. x \geq -1\ is\ not.$
You relax when you move the constraint in the direction of the cost function.

Shadow prices are always zero or negative for minimization problems.
Zero for inactive constraints.

Finding the minimum of f(x) by gradient descent

Start with a point (guess)

Repeat

- Determine a descent direction
- Choose a step size in the direction
- Update

Until stopping criterion is satisfied

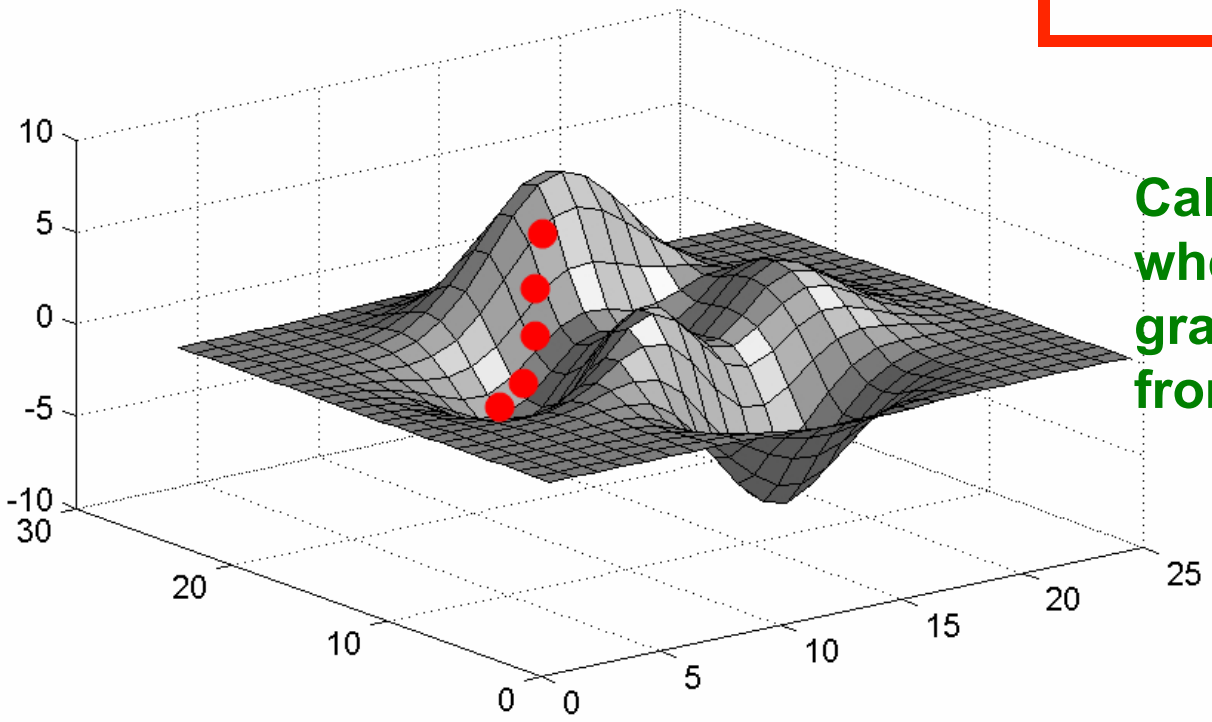
guess = x

direction = D

step = h > 0

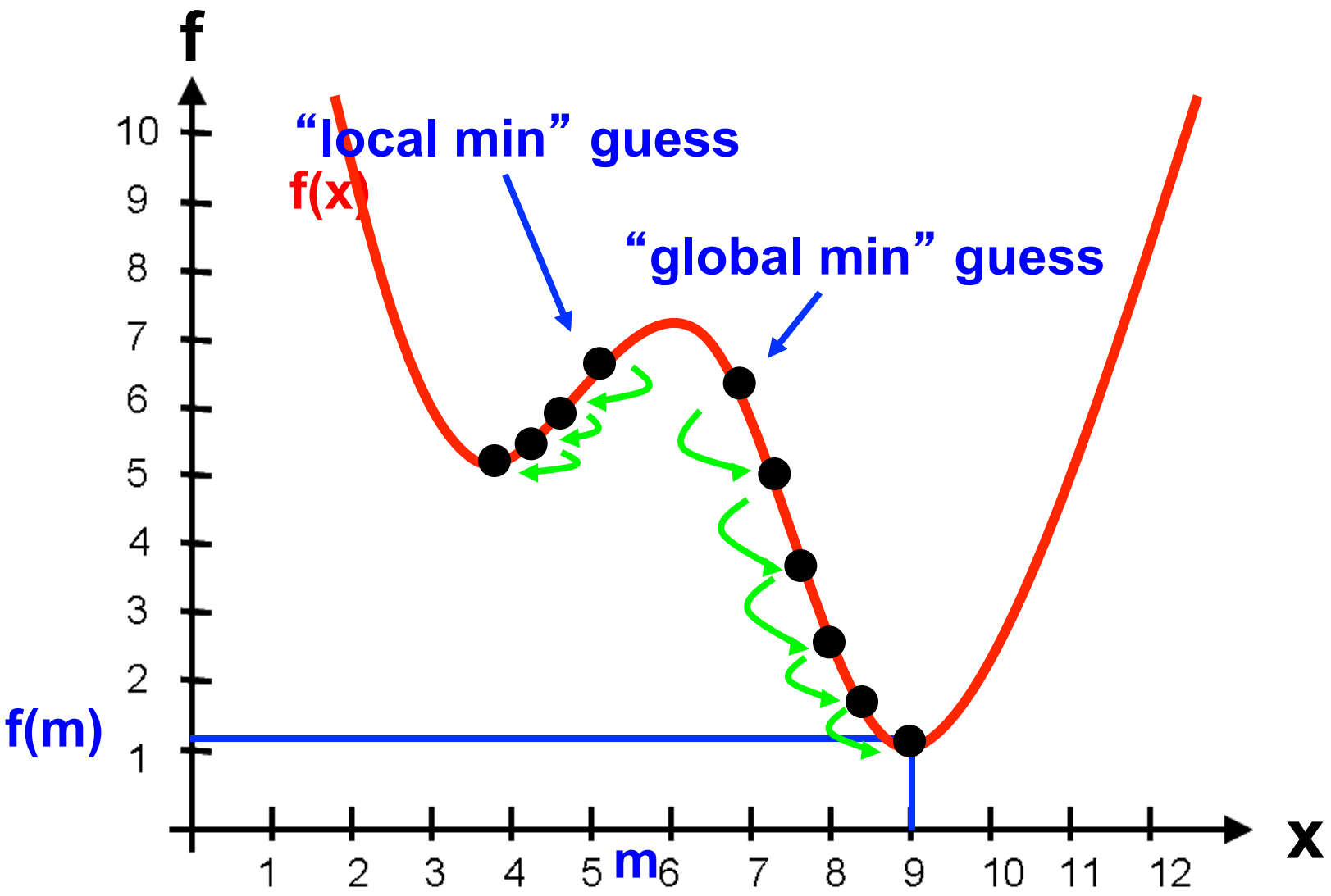
next **x:=x+hD**

||next x - x||~0



Called gradient descent when direction is the gradient or derived from it.

Problem 3: The right starting guess



Convex functions definition 1:

$f : \mathbf{R}^n \to \mathbf{R}$ is convex if $\mathbf{dom}\, f$ is a convex set and

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y)$$

for all $x, y \in \mathbf{dom}\, f$, $0 \leq \theta \leq 1$

- f is concave if $-f$ is convex
- f is strictly convex if $\mathbf{dom}\, f$ is convex and

$$f(\theta x + (1-\theta)y) < \theta f(x) + (1-\theta) f(y)$$

for $x, y \in \mathbf{dom}\, f$, $x \neq y$, $0 < \theta < 1$

[S. Boyd, L. Vandenberghe, Convex Convex Optimization lect. Notes, Stanford Univ. 2004]

Positive definite matrix

- $X^TQx > 0$ for all non-zero x : Positive definite
  - All eigenvalues positive
- $X^TQx < 0$ for all non-zero x : Negative definite
  - All eigenvalues negative
- Semidefinite when the inequality is not strict

https://www.gaussianwaves.com/2013/04/tests-for-positive-definiteness-of-a-matrix/

Convex function definition 2: First order conditions

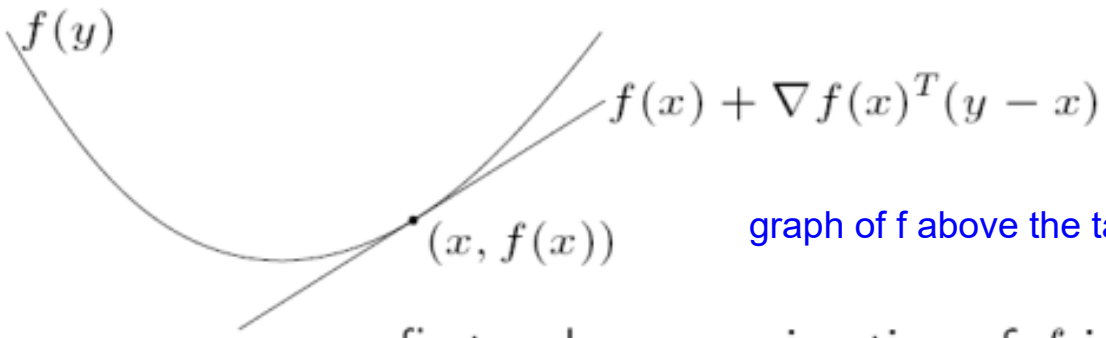
f is **differentiable** if $\mathbf{dom}\, f$ is open and the gradient

$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \ldots, \frac{\partial f(x)}{\partial x_n}\right)$$

exists at each $x \in \mathbf{dom}\, f$

1st-order condition: differentiable f with convex domain is convex iff

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) \quad \text{for all } x, y \in \mathbf{dom}\, f$$



graph of f above the tanget surface at (x,f(x))

first-order approximation of f is global underestimator

[S. Boyd, L. Vandenberghe, Convex Convex Optimization lect. Notes, Stanford Univ. 2004]

Convex function definition 3: Second order conditions

f is **twice differentiable** if $\mathbf{dom}\, f$ is open and the Hessian $\nabla^2 f(x) \in \mathbf{S}^n$,

$$\nabla^2 f(x)_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}, \quad i, j = 1, \ldots, n,$$

exists at each $x \in \mathbf{dom}\, f$

2nd-order conditions: for twice differentiable f with convex domain

- f is convex if and only if

$$\nabla^2 f(x) \succeq 0 \quad \text{for all } x \in \mathbf{dom}\, f$$

- if $\nabla^2 f(x) \succ 0$ for all $x \in \mathbf{dom}\, f$, then f is strictly convex

[S. Boyd, L. Vandenberghe, Convex Convex Optimization lect. Notes, Stanford Univ. 2004]

Newton step descent algorithm

General algorithm:

given a starting point $x \in \mathbf{dom}\, f$, tolerance $\epsilon > 0$.

repeat

1. *Compute the Newton step and decrement.*
   $\Delta x_{\mathrm{nt}} := -\nabla^2 f(x)^{-1} \nabla f(x)$; $\quad \lambda^2 := \nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x)$
2. *Stopping criterion.* **quit** if $\lambda^2/2 \leq \epsilon$.
3. *Line search.* Choose step size t by backtracking line search.
4. *Update.* $x := x + t\Delta x_{\mathrm{nt}}$.

Will find global optimum if x is in some set S in dom f s.t. f locally convex in S And S itself is a convex .

Choosing the step size t: Backtracking methods

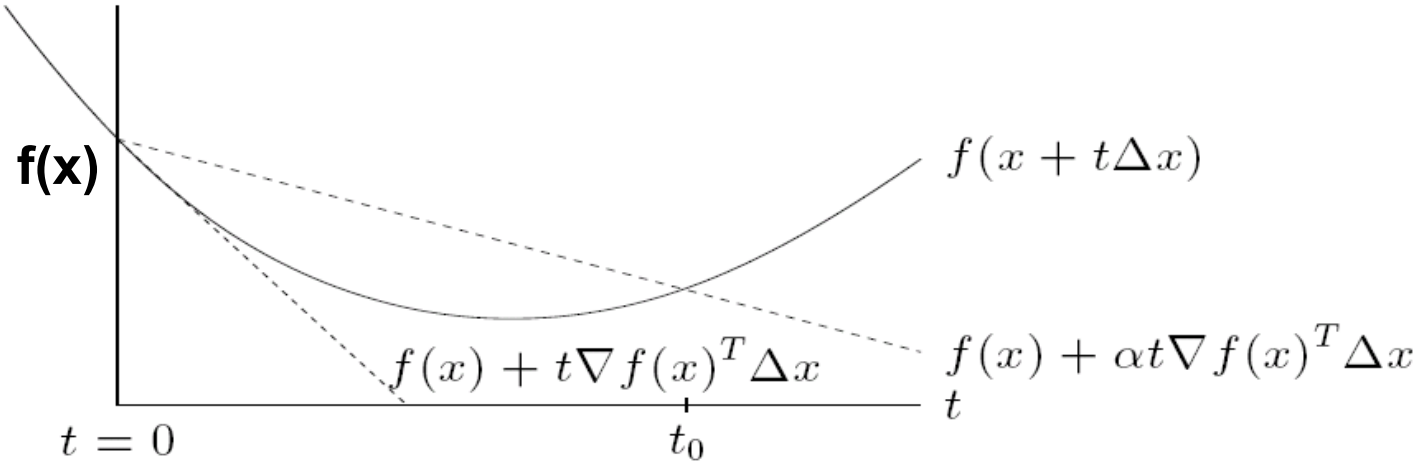
exact line search: $t = \text{argmin}_{t>0} f(x + t\Delta x)$

backtracking line search (with parameters $\alpha \in (0, 1/2)$, $\beta \in (0, 1)$)

- starting at $t = 1$, repeat $t := \beta t$ until

$$f(x + t\Delta x) < f(x) + \alpha t \nabla f(x)^T \Delta x$$

- graphical interpretation: backtrack until $t \leq t_0$



[S. Boyd, L. Vandenberghe, Convex Convex Optimization lect. Notes, Stanford Univ. 2004]

Karush-Kuhn-Tucker (KKT) Conditions

General Constrained Optimization Problem

$$\begin{array}{lll} \min & f(x) & \\ \text{s. to} & g_i(x) \leq 0, & i = 1, \cdots, m \\ & h_j(x) = 0, & j = 1, \cdots, l \end{array}$$

If x^* is a local minimum, then the following necessary conditions hold:

$$\nabla f(x^*) + \sum_{i=1}^{m} \mu^*_i \nabla g_i(x^*) + \sum_{j=1}^{l} \lambda^*_j \nabla h_j(x^*) = 0, \qquad \text{Stationarity} \tag{1}$$

$$g_i(x^*) \leq 0, \quad i = 1, \cdots, m \qquad \text{Feasibility} \tag{2}$$

$$h_j(x^*) = 0, \quad j = 1, \cdots, l \qquad \text{Feasibility} \tag{3}$$

$$\mu^*_i \geq 0, \quad i = 1, \cdots, m \qquad \text{Non-negativity} \tag{4}$$

$$\mu_i g_i(x^*) = 0, \quad i = 1, \cdots, m \qquad \text{Complementary slackness} \tag{5}$$

for some λ^*, μ^*

Remarks

- Non-zero μ_i indicates $g_i \leq 0$ is active (true with equality).
- Conditions are necessary, only.
- If problem is convex, then the conditions are necessary and sufficient.
- Lagrange multipliers λ, μ are sensitivity to perturbations in constraints
  - In economics, this is called the "shadow price"
  - In control theory, this is called the "co-state"

Go through KKT in lecture 8.

10.3 Projected Gradient Descent

So far, we were concerned with finding the optimal solution of an unconstrained optimization problem. In real life, optimization problems we are likely to come across constrained optimization problems. In this section, we discuss how to solve constrained optimization problem:

$$\min_{x \in X} f(x)$$

where f is a convex function and X is a convex set.

If we wish to use gradient descent update to a point $x_t \in X$, it is possible that the iterate $x_{t+1} = x_t - \frac{\nabla f(x_t)}{L}$ may not belong to the constraint set X. In the projected gradient descent, we simply choose the point nearest to $x_t - \frac{\nabla f(x_t)}{L}$ in the set X as x_{t+1} i.e., the projection of $x_t - \frac{\nabla f(x_t)}{L}$ onto the set X.

Definition 10.3 *The projection of a point y, onto a set X is defined as*

$$\Pi_X(y) = \operatorname*{argmin}_{x \in X} \frac{1}{2}\|x - y\|_2^2.$$

Projected Gradient Descent (PGD): Given a starting point $x_0 \in X$ and step-size $\gamma > 0$, PGD works as follows until a certain stopping criterion is satisfied,

$$x_{t+1} = x_t - \gamma \Pi_X(x_t - \nabla f(x_t)), \forall t \geq 1.$$

$x_{t+1} = \Pi_X(x_t + \gamma \nabla f(x_t))$

$t = 0, 1, 2, \ldots$

In this lecture, for an L smooth convex function, we fix the step-size to be $\gamma = \frac{1}{L}$.

The Mixed Integer Linear Program (MILP)

$$\min \; c^T x + c'^T x'$$

$$s.t.$$

$$Ax \leq b$$

$$A' x' \leq b'$$

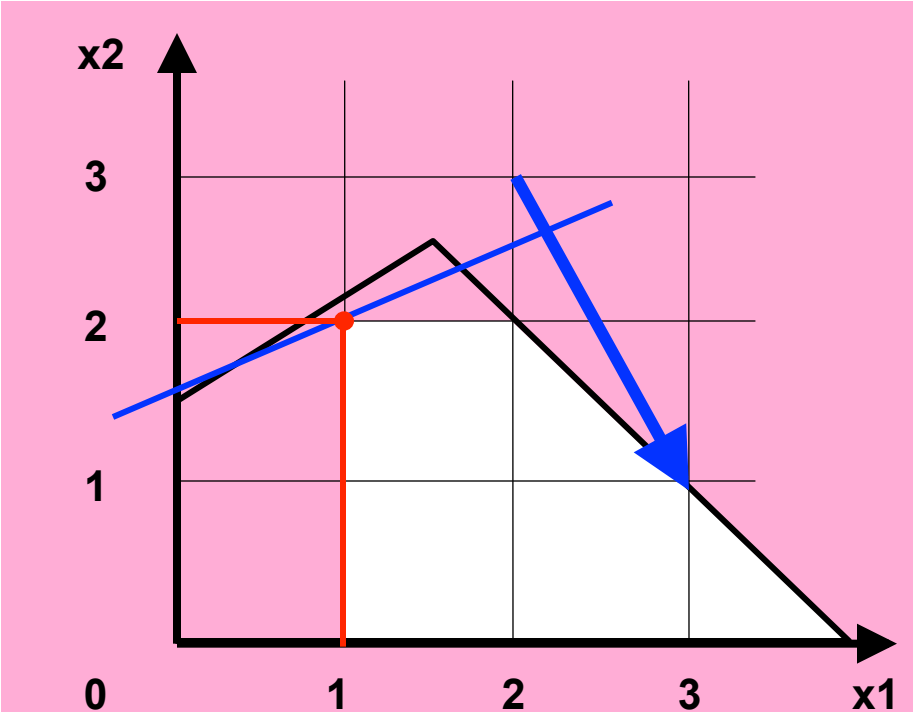
$$x' \in Integers$$

$$x \in Reals$$

Branch and Bound Procedure

- Problem = real valued relaxation + integer constraints
  - solution to relaxation is bound on solution to problem
- Solve real valued relaxation
  - if solution satisfies integer constraints, solution is optimal
- Otherwise, create sub-problems by branching.
  - Each sub-problem is problem plus some integer constraints
  - Sub-problems must be created intelligently. This is the art.
  - Union of feasible sets of sub-problems must be subset of feasible set of branched problem but superset of Problem
- Keep track of Best Solution found for Problem
  - Continue while there are active problems
- A sub-problem becomes inactive when
  - it is infeasible, or
  - Its solution solves the Problem
  - it is branched on
  - it can be pruned because it is worse than Best Solution

Example: summary



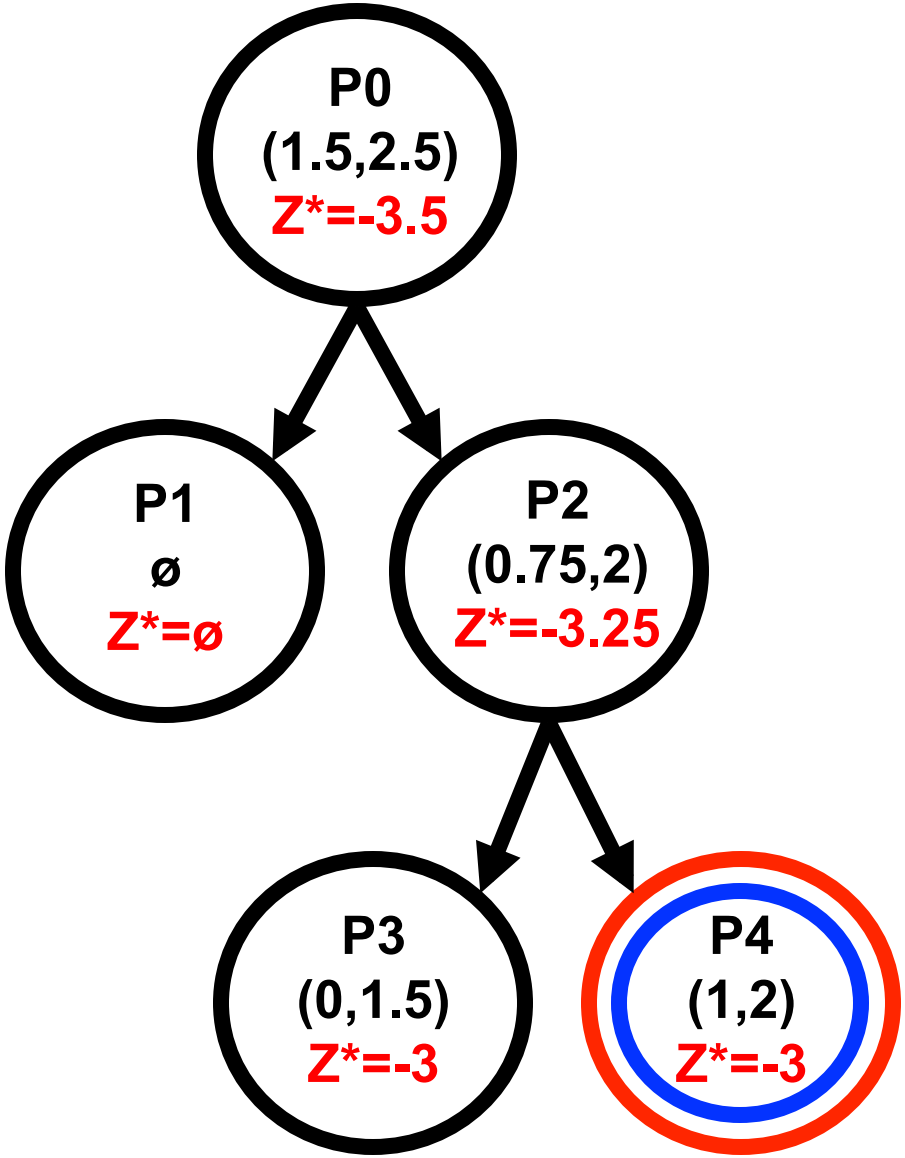
P0 $\min: \quad x_1 - 2x_2$

$\text{s.t.} \quad -4x_1 + 6x_2 \leq 9$

$x_1 + x_2 \leq 4$

$x_1 \geq 0$

$x_2 \geq 0$



Total Unimodularity of a Matrix

A matrix A is totally unimodular (TU) if every square submatrix has determinant -1, 0 or 1

If A is TU and b is integer all the vertices of the polyhedron Ax <= b are integral

https://en.wikipedia.org/wiki/Unimodular_matrix#Total_unimodularity

Does this give us a solution method for TM IP's?

If A is TU and b is integer all the vertices of the polyhedron Ax <= b are integral

Consider the optimization problem min <c,x>, Ax <= b,
x element of the Integers.

Does solving its LP relaxation solve the IP

YES – if we use simplex https://youtu.be/jh_kkR6m8H8

Is this a polynomial time method of solving TM IP's?

Shortest Path Integer Program

$$\text{minimize}: \quad Z = \sum_{i,j} c_{ij} x_{ij}$$

such that :

$$\sum_j x_{ij} - \sum_j x_{ji} = 0, \qquad i \in Nodes - \{A, B\}$$

$$\sum_j x_{Aj} - \sum_j x_{jA} = 1 \qquad \text{(origin constraint)}$$

$$\sum_j x_{Bj} - \sum_j x_{jB} = -1 \qquad \text{(destination constraint)}$$

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

Integer Constraint

LP

Assignment Problem: Mathematical Formulation

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

Minimize the total time to complete all tasks

with constraints

$$\sum_{i=1}^{n} x_{ij} = 1, \text{ for each job j}$$

Each task must have a worker

$$\sum_{j=1}^{n} x_{ij} = 1, \text{ for each engineer i}$$

Each worker must have a task

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

Symmetric TSP Formulation

$$Minimize \quad Z = \sum_{e \in E} c_e x_e$$

$$\sum_{e \in \delta(v)}^{n} x_e = 2, \text{ for all } v \in V$$

$$\sum_{e \in \delta(U)}^{n} x_e \geq 2, \text{ for all } U \subseteq V, 2 \leq |U| \leq |V| - 1$$

$$x_e \in \{0,1\}, \quad e \in E$$

Note: For a set K, |K| denotes the cardinality of K (the number of elements in K)

Asymmetric TSP Formulation

$$\min \sum_{j=1}^{m} \sum_{i=1}^{m} c_{ij} x_{ij}$$

$$s.t. \sum_{j=1}^{m} x_{ij} = 1 \qquad \text{for } i = 1,\ldots,m$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \text{for } j = 1,\ldots,m$$

$$\sum_{i \in K} \sum_{j \in K} x_{ij} \leq |K - 1|, \text{ for all } K \subset \{1,2,\ldots,m\}$$

$$x_{ij} = 0 \text{ or } 1 \qquad \text{for all } i, j$$

http://iris.gmu.edu/~khoffman/papers/trav_salesman.html

TSP MTZ formulation

https://medium.com/swlh/techniques-for-subtour-elimination-in-traveling-salesman-problem-theory-and-implementation-in-71942e0baf0c

The assignment problem constraints plus

$t_j > t_i - B(1 - x_{ij})$, i,j$\in \{2, \ldots, N\}$, *B large*

$t_i \in \{1, \ldots, N-1\}$

t_1=0

Traveling Salesman Problem

- Note that there are a tremendous number of constraints
- For the 20-city problem, there are 524,288 constraints
- For the 300-city problem, there are 1018517988167243043134222844204689080525734196832968125318070224677190649881668353091698688 constraints
- Try putting that into Matlab!
- Is there a more efficient way to solve this problem?