CE 191: Civil & Environmental Engineering Systems Analysis

LEC 17 : Final Review

Professor Scott Moura
Civil & Environmental Engineering
University of California, Berkeley

Fall 2014

Logistics

- **Date/Time:** Tuesday December 16, 2013, 3:00p-6:00p
- **Where:** 406 Davis Hall
- **Format/Rules:** See Practice Final (bCourses)
- **Topics Covered:** Everything

Topics Covered - 1

- Unit 1: Linear Programming
  - Formulation
  - Graphical Solutions to LP
  - Transportation & Shortest Path Problems
  - Applications (e.g. Water Supply Network)
- Unit 2: Quadratic Programming
  - Least Squares
  - Optimality Conditions
  - Applications (e.g. Energy Portfolio Optimization)
- Unit 3: Integer Programming
  - Dijkstra's Algorithm
  - Branch & Bound
  - Mixed Integer Programming and "Big-M" method
  - Applications (e.g. Construction Scheduling)

Topics Covered - 2

- Unit 4: Nonlinear Programming
  - Convex functions and convex sets
  - Local/global optima
  - Gradient Descent
  - Barrier Functions
  - KKT Conditions
  - Applications (e.g. WIFI tower location)
- Unit 5: Dynamic Programming
  - Principle of Optimality
  - Shortest Path Problems
  - Applications (e.g. knapsack, smart appliances, Cal Band)

Outline

Linear Program Formulation

"Matrix notation":

$$\begin{array}{rl} \text{Minimize:} & c^T x \\ \text{subject to:} & Ax \leq b \end{array}$$

where

$$\begin{aligned} x &= [x_1, x_2, \ldots, x_N]^T \\ c &= [c_1, c_2, \ldots, c_N]^T \\ [A]_{i,j} &= a_{i,j}, \quad A \in \mathbb{R}^{M \times N} \\ b &= [b_1, b_2, \ldots, b_M]^T \end{aligned}$$

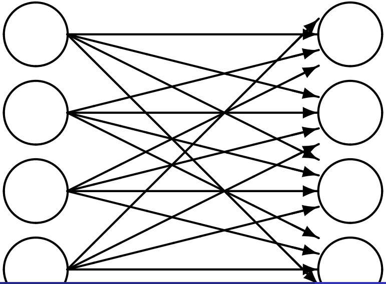
Ex 1: Transportation Problem - General LP Formulation

$$\text{min:} \quad \sum_{i=1}^{M} \sum_{j=1}^{N} c_{ij} x_{ij}$$

$$\text{s. to} \quad \sum_{i=1}^{M} x_{ij} = d_j, \quad j = 1, \cdots, N$$

$$\sum_{j=1}^{N} x_{ij} = s_i, \quad i = 1, \cdots, M$$

$$x_{ij} \geq 0, \quad \forall i, j$$



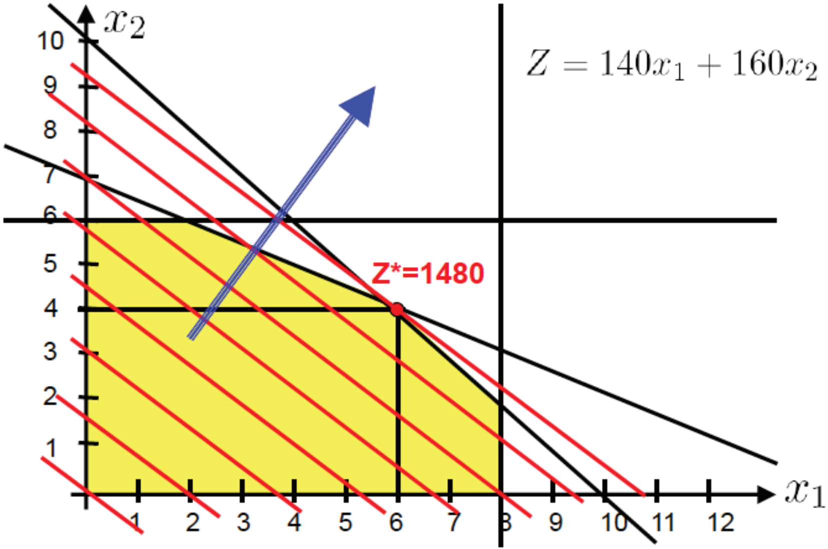
Example 2: Shortest Path

$$\text{Minimize:} \qquad J = \sum_{j \in N_A} c_{Aj} x_{Aj} + \sum_{i=1}^{10} \sum_{j \in N_i} c_{ij} x_{ij} + \sum_{j \in N_B} c_{jB} x_{jB}$$

$$\text{subject to:} \qquad \begin{aligned} \sum_{j \in N_i} x_{ji} &= \sum_{j \in N_i} x_{ij}, \quad i = 1, \cdots, 10 \\ \sum_{j \in N_A} x_{Aj} &= 1 \\ \sum_{j \in N_B} x_{jB} &= 1 \\ x_{ij} &\geq 0, \;\; x_{Aj} \geq 0, \;\; x_{jB} \geq 0 \end{aligned}$$

N_i : Set of nodes j with direct connections to node i

Graphical Solns to LP



Outline

1. Unit 1: Linear Programming
2. Unit 2: Quadratic Programming
3. Unit 3: Integer Programming
4. Unit 4: Nonlinear Programming
5. Unit 5: Dynamic Programming

Conditions for Optimality

Consider an unconstrained QP

$$\min \qquad f(x) = x^T Q x + R x$$

Recall from calculus (e.g. Math 1A) the first order necessary condition (FONC) for optimality: If x^* is an optimum, then it must satisfy

$$\begin{aligned} \frac{d}{dx} f(x^*) &= 0 \\ &= 2Qx^* + R = 0 \\ &\Rightarrow \boxed{x^* = -\frac{1}{2} Q^{-1} R} \end{aligned}$$

Also recall the second order sufficiency condition (SOSC): If $x^\dagger$ is a stationary point (i.e. it satisfies the FONC), then it is also a minimum if

$$\begin{aligned} \frac{\partial^2}{\partial x^2} f(x^\dagger) & \qquad \text{positive definite} \\ \Rightarrow Q & \qquad \text{positive definite} \end{aligned}$$

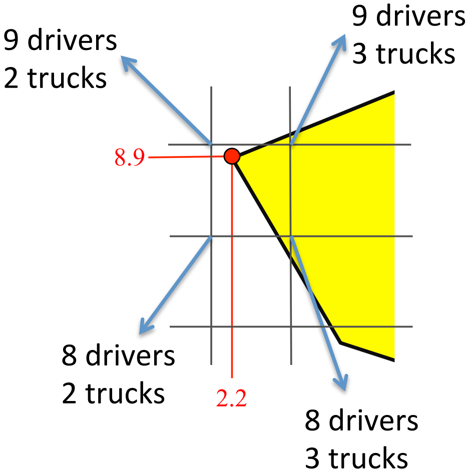
Nature of stationary point based on SOSC

Hessian matrix	Quadratic form	Nature of $x^\dagger$
positive definite	$x^T Q x > 0$	local minimum
negative definite	$x^T Q x < 0$	local maximum
positive semi-definite	$x^T Q x \geq 0$	valley
negative semi-definite	$x^T Q x \leq 0$	ridge
indefinite	$x^T Q x$ any sign	saddle point

Outline

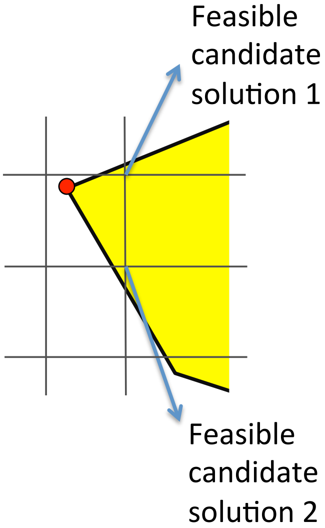
Fractional solution

What should one do?



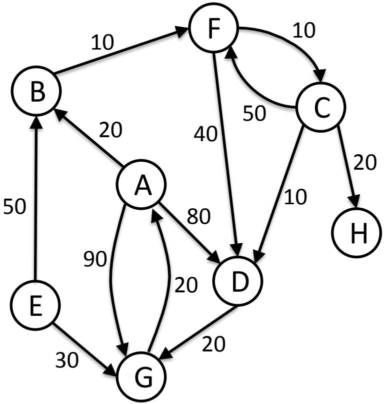
Fractional solution

What should one do?



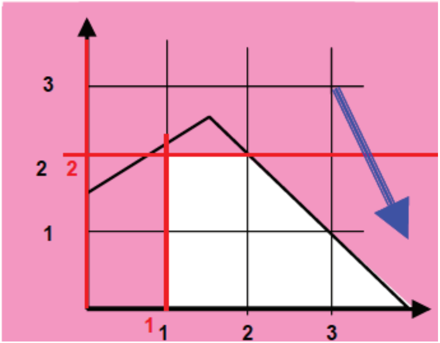
Dijkstra's Algorithm Example - Final Result

Result: Shortest path and distance from A

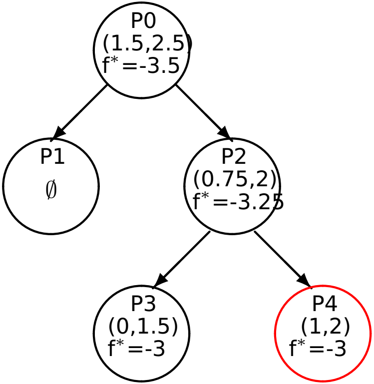


A $\rightarrow$	B	C	D	E	F	G	H
(1) A	20	∞	80	∞	∞	90	∞
(2) B	20	∞	80	∞	30	90	∞
(3) F	20	40	70	∞	30	90	∞
(4) C	20	40	50	∞	30	90	60
(5) D	20	40	50	∞	30	70	60
(6) H	20	40	50	∞	30	70	60
(7) G	20	40	50	∞	30	70	60
(8) E	20	40	50	∞	30	70	60

Branch and bound: summary



$$
\begin{aligned}
\min \quad & x_1 - 2x_2 \\
\text{s. to} \quad & -4x_1 + 6x_2 \leq 9 \\
& x_1 + x_2 \leq 4 \\
& x_1 \geq 0 \\
& x_2 \geq 0 \\
& x_1, x_2 \in \mathbb{Z}
\end{aligned}
$$



Transformation of **OR** into an **AND**

Pick a very large number M.
Also consider a decision variable $d \in \{0, 1\}$.

For sufficiently large M, the following two statements are equivalent:

Statement 1:

$$\textbf{OR} \qquad \begin{cases} t_1 - t_2 \geq \Delta & \text{if } t_1 \geq t_2 \\ t_2 - t_1 \geq \Delta & \text{o.w.} \end{cases}$$

Statement 2:

$$\textbf{AND} \qquad \begin{cases} t_1 - t_2 \geq \Delta - Md \\ t_1 - t_2 \leq -\Delta + M(1 - d) \end{cases}$$

Transform an **OR** condition to an **AND** condition,
at the expense of an added binary variable d.
Variable d encodes the **order**.

$d = 0 \rightarrow$ Order : t_2, t_1.
$d = 1 \rightarrow$ Order : t_1, t_2.

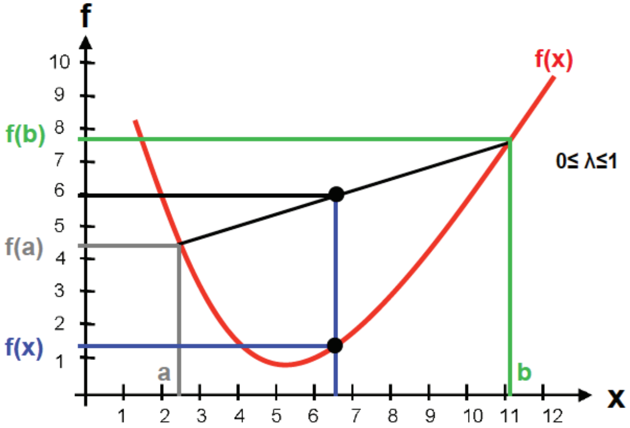
Outline

Convex Functions

Let $D = \{x \in \mathbb{R} \mid a \le x \le b\}$.

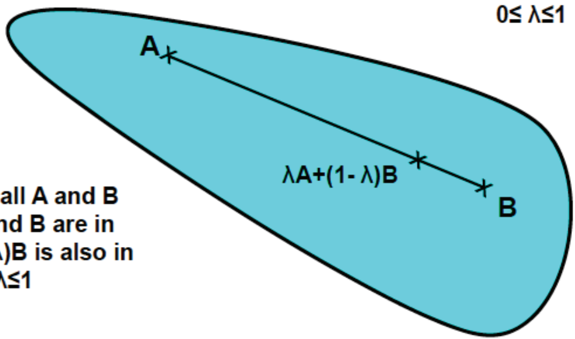
Def'n (Convex function) : The function $f(x)$ is convex on D if and only if

$$f(x) = f(\lambda a + (1-\lambda)b) \le \lambda f(a) + (1-\lambda) f(b)$$



Convex Sets

Definition:
Convex set: for all A and B in the set, if A and B are in the set, λA+(1- λ)B is also in this set, for 0≤ λ≤1



This set is convex

Definitions of minimizers

Def'n (Global minimizer) : $x^* \in D$ is a global minimizer of f on D if

$$f(x^*) \leq f(x) \qquad \forall x \in D$$

in English: x^* minimizes f everywhere in D.

Def'n (Local minimizer) : $x^* \in D$ is a local minimizer of f on D if

$$\exists \epsilon > 0 \quad \text{s.t.} \quad f(x^*) \leq f(x) \qquad \forall x \in D \cap \{x \in \mathbb{R} \mid \|x - x^*\| < \epsilon\}$$

in English: x^* minimizes f locally in D.

Gradient Descent Algorithm

Start with an initial guess

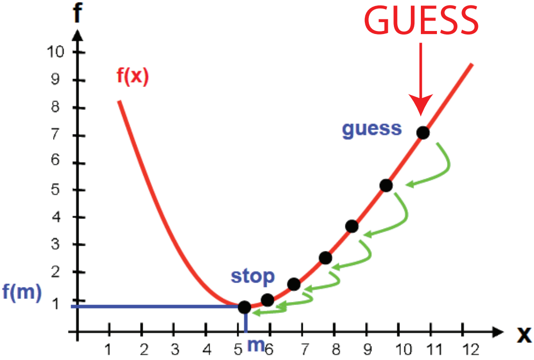
Repeat

– Determine descent direction

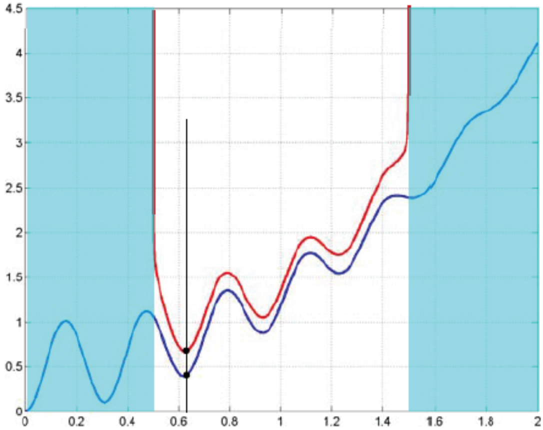
– Choose a step size

– Update

Until stopping criterion is satisfied



Log Barrier Functions



Consider: $\min f(x)$
s. to: $a \leq x \leq b.$

Convert "hard" constraints to "soft" constraints.

Consider barrier function:

$$b(x, \varepsilon) = -\varepsilon \log\left((x-a)(b-x)\right)$$

as $\varepsilon \to 0$.

Modified optimization:

$$\min f(x) + \varepsilon b(x, \varepsilon)$$

Pick ε small, solve.
Set $\varepsilon = \varepsilon/2$. Solve again.
Repeat

Method of Lagrange Multipliers

Equality Constrained Optimization Problem

$$\begin{aligned} \min \quad & f(x) \\ \text{s. to} \quad & h_j(x) = 0, \qquad j = 1, \cdots, l \end{aligned}$$

Lagrangian

Introduce the so-called "Lagrange multipliers" $\lambda_j, j = 1, \cdots, l$. The Lagrangian is

$$\begin{aligned} L(x) &= f(x) + \sum_{j=1}^{l} \lambda_j h_j(x) \\ &= f(x) + \lambda^T h(x) \end{aligned}$$

First order Necessary Condition (FONC)

If a local minimum x^* exists, then it satisfies

$$\nabla L(x^*) = \nabla f(x^*) + \lambda^T \nabla h(x^*) = 0$$

Karush-Kuhn-Tucker (KKT) Conditions

General Constrained Optimization Problem

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

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

$$\nabla f(x^*) + \mu^T \nabla g(x^*) + \lambda^T \nabla h(x^*) = 0, \qquad \text{Stationarity} \tag{1}$$

$$
\begin{aligned}
g(x^*) &\leq 0, && \text{Feasibility} && (2) \\
h(x^*) &= 0, && \text{Feasibility} && (3) \\
\mu &\geq 0, && \text{Non-negativity} && (4) \\
\mu^T g(x^*) &= 0, && \text{Complementary slackness} && (5)
\end{aligned}
$$

Outline

Formulation

Discrete-time system

$$x_{k+1} = f(x_k, u_k), \qquad k = 0, 1, \cdots, N-1$$

k : discrete time index
x_k : state - summarizes current configuration of system at time k
u_k : control - decision applied at time k
N : time horizon - number of times control is applied

Additive Cost

$$J = \sum_{k=0}^{N-1} c_k(x_k, u_k) + c_N(x_N)$$

c_k : instantaneous cost - instantaneous cost incurred at time k
c_N : final cost - incurred at time N

Principle of Optimality (in math)

Define $V_k(x_k)$ as the optimal "cost-to-go" from time step k to the end of the time horizon N, given the current state is x_k.

Then the principle of optimality can be written in recursive form as:

$$V_k(x_k) = \min_{u_k} \{c_k(x_k, u_k) + V_{k+1}(x_{k+1})\}$$

with the boundary condition

$$V_N(x_N) = c_N(x_N)$$

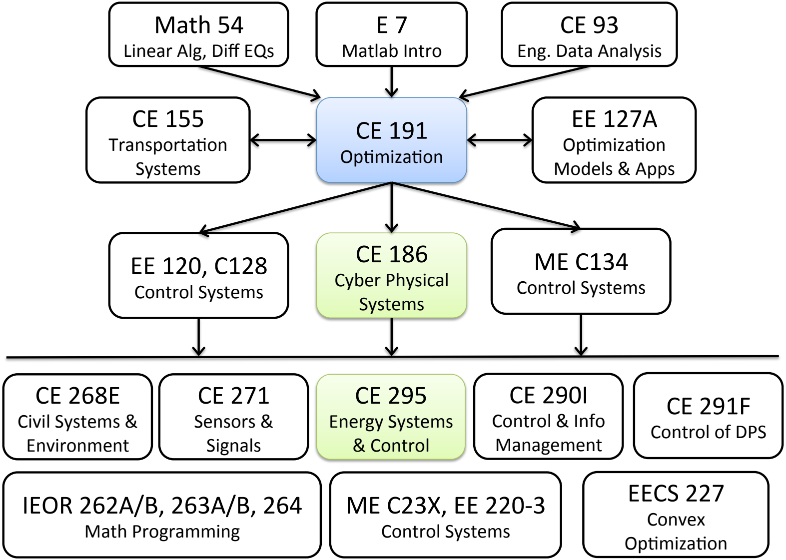
Admittedly awkward aspects:

- You solve the problem backward!
- You solve the problem recursively!

DP Application Examples

- Shortest Path in Networks
- Knapsack Problem
- Smart Appliances
- Resource Economics
- Cal Band formations

Flowchart of Methods-based Courses



Why take CE 191?

Why take CE 191?

Learn to abstract mathematical programs

from physical systems to “optimally” design

a civil engineered system.

Thank you for a fantastic semester!