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
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

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
“Matrix notation”:

Minimize: \( c^T x \)
subject to: \( Ax \leq b \)

where

\[
\begin{align*}
x &= [x_1, x_2, \ldots, x_N]^T \\
c &= [c_1, c_2, \ldots, c_N]^T \\
[A]_{i,j} &= a_{i,j}, \quad A \in \mathbb{R}^{M \times N} \\
b &= [b_1, b_2, \ldots, b_M]^T
\end{align*}
\]
Ex 1: Transportation Problem - General LP Formulation

\[
\begin{align*}
\text{min:} & \quad \sum_{i=1}^{M} \sum_{j=1}^{N} c_{ij} x_{ij} \\
\text{s. to} & \quad \sum_{i=1}^{M} x_{ij} = d_j, \quad j = 1, \ldots, N \\
& \quad \sum_{j=1}^{N} x_{ij} = s_i, \quad i = 1, \ldots, M \\
& \quad x_{ij} \geq 0, \quad \forall i, j
\end{align*}
\]
Example 2: Shortest Path

Minimize: \[ J = \sum_{j \in N_A} C_{Aj} X_{Aj} + \sum_{i=1}^{10} \sum_{j \in N_i} C_{ij} X_{ij} + \sum_{j \in N_B} C_{jB} X_{jB} \]

subject to:

\[ \sum_{j \in N_i} x_{ji} = \sum_{j \in N_i} x_{ij}, \quad i = 1, \ldots, 10 \]

\[ \sum_{j \in N_A} x_{Aj} = 1 \]

\[ \sum_{j \in N_B} x_{jB} = 1 \]

\[ x_{ij} \geq 0, \quad x_{Aj} \geq 0, \quad x_{jB} \geq 0 \]

\( N_i \): Set of nodes \( j \) with direct connections to node \( i \)
Graphical Solns to LP

\[ Z = 140x_1 + 160x_2 \]

\( Z^* = 1480 \)
Conditions for Optimality

Consider an unconstrained QP

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

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

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

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

$$
\frac{\partial^2}{\partial x^2} f(x^\dagger) \quad \text{positive definite}
\Rightarrow Q \quad \text{positive definite}$$
<table>
<thead>
<tr>
<th>Hessian matrix</th>
<th>Quadratic form</th>
<th>Nature of $x^\dagger$</th>
</tr>
</thead>
<tbody>
<tr>
<td>positive definite</td>
<td>$x^T Q x &gt; 0$</td>
<td>local minimum</td>
</tr>
<tr>
<td>negative definite</td>
<td>$x^T Q x &lt; 0$</td>
<td>local maximum</td>
</tr>
<tr>
<td>positive semi-definite</td>
<td>$x^T Q x \geq 0$</td>
<td>valley</td>
</tr>
<tr>
<td>negative semi-definite</td>
<td>$x^T Q x \leq 0$</td>
<td>ridge</td>
</tr>
<tr>
<td>indefinite</td>
<td>$x^T Q x$ any sign</td>
<td>saddle point</td>
</tr>
</tbody>
</table>
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
Fractional solution

What should one do?

9 drivers  2 trucks
8 drivers  3 trucks
8.9

9 drivers  3 trucks
8 drivers  2 trucks
2.2
Fractional solution

What should one do?

Feasible candidate solution 1

Feasible candidate solution 2
Result: Shortest path and distance from A

![Graph and Table]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) A</td>
<td>20</td>
<td>∞</td>
<td>80</td>
<td>∞</td>
<td>∞</td>
<td>90</td>
<td>∞</td>
</tr>
<tr>
<td>(2) B</td>
<td>20</td>
<td>∞</td>
<td>80</td>
<td>∞</td>
<td>30</td>
<td>90</td>
<td>∞</td>
</tr>
<tr>
<td>(3) F</td>
<td>20</td>
<td>40</td>
<td>70</td>
<td>∞</td>
<td>30</td>
<td>90</td>
<td>∞</td>
</tr>
<tr>
<td>(4) C</td>
<td>20</td>
<td>40</td>
<td>50</td>
<td>∞</td>
<td>30</td>
<td>90</td>
<td>60</td>
</tr>
<tr>
<td>(5) D</td>
<td>20</td>
<td>40</td>
<td>50</td>
<td>∞</td>
<td>30</td>
<td>70</td>
<td>60</td>
</tr>
<tr>
<td>(6) H</td>
<td>20</td>
<td>40</td>
<td>50</td>
<td>∞</td>
<td>30</td>
<td>70</td>
<td>60</td>
</tr>
<tr>
<td>(7) G</td>
<td>20</td>
<td>40</td>
<td>50</td>
<td>∞</td>
<td>30</td>
<td>70</td>
<td>60</td>
</tr>
<tr>
<td>(8) E</td>
<td>20</td>
<td>40</td>
<td>50</td>
<td>∞</td>
<td>30</td>
<td>70</td>
<td>60</td>
</tr>
</tbody>
</table>
\[ \text{min} \quad x_1 - 2x_2 \]
\[ \text{s. to} \quad -4x_1 + 6x_2 \leq 9 \]
\[ x_1 + x_2 \leq 4 \]
\[ x_1 \geq 0 \]
\[ x_2 \geq 0 \]
\[ x_1, x_2 \in \mathbb{Z} \]
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:
\[
\text{OR} \begin{cases} 
  t_1 - t_2 \geq \Delta & \text{if } t_1 \geq t_2 \\
  t_2 - t_1 \geq \Delta & \text{o.w.}
\end{cases}
\]

Statement 2:
\[
\text{AND} \begin{cases} 
  t_1 - t_2 \geq \Delta - Md \\
  t_1 - t_2 \leq -\Delta + M(1 - d)
\end{cases}
\]

Transform an OR condition to an AND condition, at the expense of an added binary variable $d$.
Variable $d$ encodes the order.
\[
\begin{align*}
  d = 0 & \rightarrow \text{Order: } t_2, t_1. \\
  d = 1 & \rightarrow \text{Order: } t_1, t_2.
\end{align*}
\]
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
Let $D = \{x \in \mathbb{R} \mid a \leq x \leq b\}$.

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

$$f(x) = f(\lambda a + (1 - \lambda)b) \leq \lambda f(a) + (1 - \lambda)f(b)$$
Definition:
Convex set: for all $A$ and $B$ in the set, if $A$ and $B$ are in the set, $\lambda A + (1 - \lambda)B$ is also in this set, for $0 \leq \lambda \leq 1$

This set is convex
Definitions of minimizers

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

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

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

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

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

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:
\[
b(x, \varepsilon) = -\varepsilon \log ((x - a)(b - x))
\]
as \( \varepsilon \to 0 \).

Modified optimization:
\[
\min f(x) + \varepsilon b(x, \varepsilon)
\]

Pick \( \varepsilon \) small, solve.
Set \( \varepsilon = \varepsilon / 2 \). Solve again.
Repeat
Method of Lagrange Multipliers

Equality Constrained Optimization Problem

\[
\begin{align*}
\text{min} & \quad f(x) \\
\text{s. to} & \quad h_j(x) = 0, \quad j = 1, \ldots, l
\end{align*}
\]

Lagrangian

Introduce the so-called “Lagrange multipliers” \( \lambda_j, j = 1, \cdots, l \). The Lagrangian is

\[
L(x) = f(x) + \sum_{j=1}^{l} \lambda_j h_j(x)
\]

\[
= f(x) + \lambda^T h(x)
\]

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

\[
\begin{align*}
\min & \quad f(x) \\
\text{s. to} & \quad g_i(x) \leq 0, \quad i = 1, \ldots, m \\
& \quad h_j(x) = 0, \quad j = 1, \ldots, l
\end{align*}
\]

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

\[
\nabla f(x^*) + \mu^T \nabla g(x^*) + \lambda^T \nabla h(x^*) = 0, \quad \text{Stationarity} \quad (1)
\]

\[
\begin{align*}
\quad g(x^*) & \leq 0, \quad \text{Feasibility} \quad (2) \\
\quad h(x^*) & = 0, \quad \text{Feasibility} \quad (3) \\
\quad \mu & \geq 0, \quad \text{Non-negativity} \quad (4) \\
\quad \mu^T g(x^*) & = 0, \quad \text{Complementary slackness} \quad (5)
\end{align*}
\]
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
**Discrete-time system**

\[ x_{k+1} = f(x_k, u_k), \quad k = 0, 1, \ldots, N - 1 \]

- \( k \): 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 \( N \)
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:

\[
V_k(x_k) = \min_{u_k} \left\{ c_k(x_k, u_k) + V_{k+1}(x_{k+1}) \right\}
\]

with the boundary condition

\[
V_N(x_N) = c_N(x_N)
\]

Admittedly awkward aspects:

- You solve the problem \textit{backward}!
- You solve the problem \textit{recursively}!
DP Application Examples

- Shortest Path in Networks
- Knapsack Problem
- Smart Appliances
- Resource Economics
- Cal Band formations
Why take CE 191?

Learn to abstract mathematical programs from physical systems to "optimally" design a civil engineered system.
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!