Applying Optimization: Some Samples

Reference
Linear problems example: *A.D. Belegundu and T.R. Chandrupatla (1999). Optimization Concepts and Applications in Engineering. Upper Saddle River, New Jersey.*

1. Linear Optimization

- Idea: many problems of optimization are linear, but of high dimension.
- Parameter space is $[x_1, x_2, x_3, \ldots, x_n]$ – this is what we are trying to find the best values of
- Best parameter set *minimizes or maximizes a linear cost*, e.g.,

$$J = 14x_1 + 9x_3 + 42x_4$$

- but the parameter space is confined by some *equalities E*, e.g.,

$$x_1 + 3x_2 + 7x_4 = 16,$$

- and some *inequalities I*, e.g.,

$$3x_1 - 4x_3 + x_7 <= 30.$$

- A total of I+E constraint equations for n parameters. Obviously, I+E >= n for a solution to exist

Example of Fuel Selection

A case where n = I+E ; unique solution

The *problem statement:*

- Natural gas has 0.12% sulfur, costs \$55/(kg/s), and gives 61MJ/kg heat energy
- Coal has 2.80% sulfur, costs \$28/(kg/s), and gives 38MJ/kg heat energy
- We have a steady 4MW load requirement.
- The sulfur emissions by weight have to be equal to or less than 2.5%.
- Minimize the money cost.

In *mathematical form:*

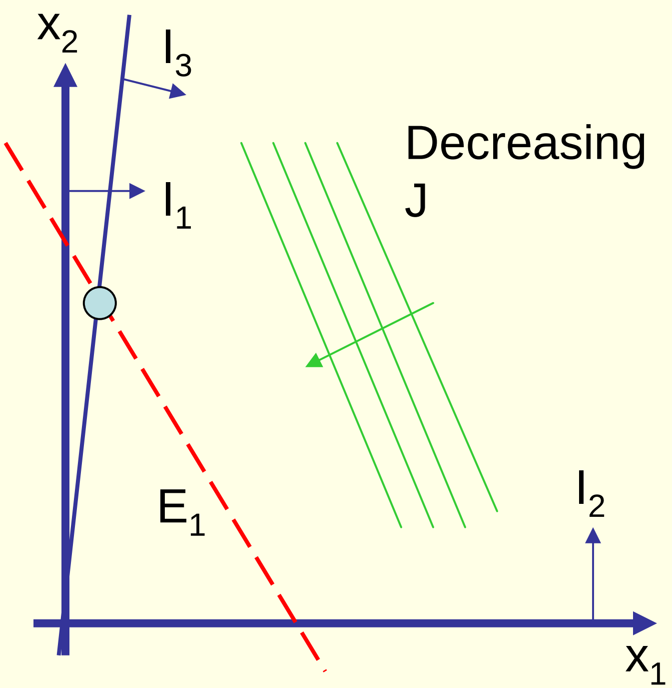
x_1 = kg of natural gas to burn per second
x_2 = kg of coal to burn per second

$J = 55x_1 + 28x_2$ (cost)

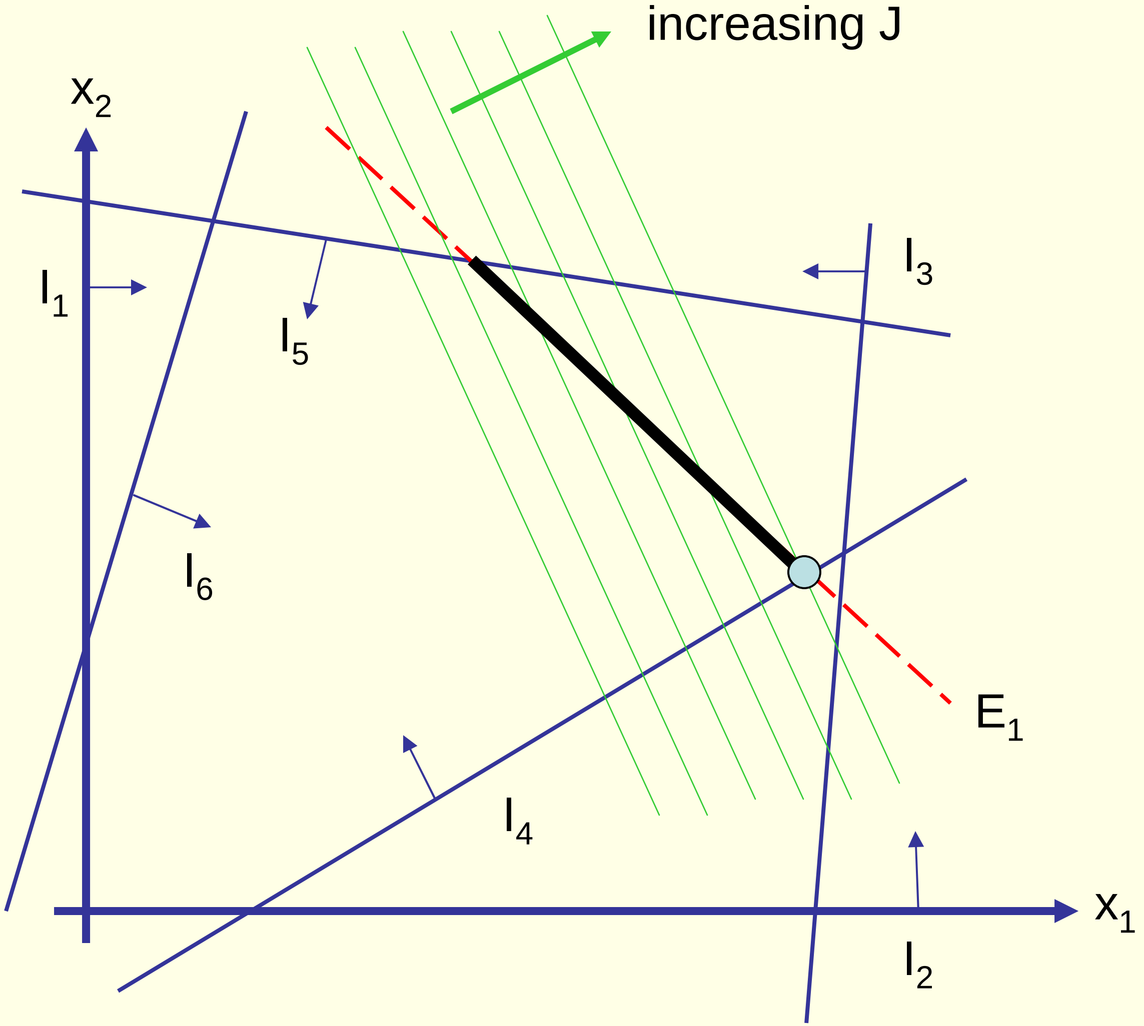
$61x_1 + 38x_2 = 4$ (E_1)

$0.12x_1 + 2.8x_2 <= 2.5(x_1 + x_2)$
→ $x_2 <= 8x_1$ (I_3)

Optimum: $x_1 = 0.011$, $x_2 = 0.087$ kg/s



More complex cases: the 2D case tells all!



Solution always falls within admissible regions defined by inequalities, AND along equality lines

OR

Solution always falls *on a vertex of n constraint equations, either I or E*.

Leads to a *simple systematic* procedure for small (e.g., $n < 5$, $I+E < 10$) problems

Idea: Calculate J at *all* existent vertices, and pick the best one.

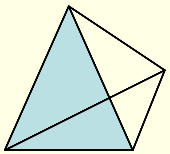
How many vertices are there to consider?

N = "Combinations of n items from a collection of I+E items" →

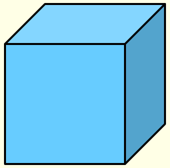
N = (I+E)! / n! (I+E-n)!

Consider 3-space (n=3);

If I+E = 4, N = 4 "TETRAHEDRON"



If I+E = 6, N = 20 "CUBE" *Not all 20 vertices may exist!*



Consider 5-space (n=5);

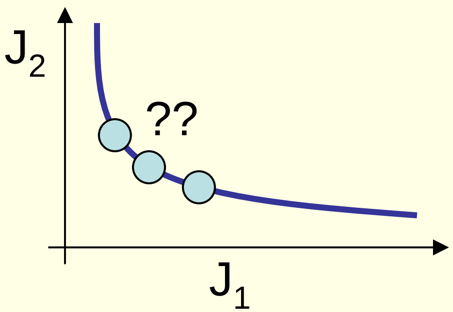
If I+E = 10, N ~ 250 (still quite reasonable for calculations)

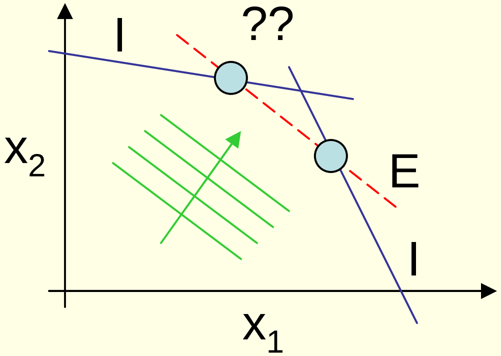
1. Step through *all combinations of n equations* from the I+E available, solving an n-dimensional linear problem for each; Ax = b, when A is non-singular. If A is singular, no vertex exists for the set.
2. For a calculated vertex location, check that it meets all of the *other* I+E-n constraints. If it does not, then throw it out.
3. Evaluate J at all the admissible vertices.
4. Pick the best one!

More general case is Linear Programming; very powerful and specialized tools are available!

2. Min-Max Optimization

- Difficulties with the linear and nonlinear continuous problems
  - Multiple objectives or costs
  - The real world sometimes offers only finite choices, with no clear "best candidate." Tradeoffs must be made somehow!
  - Sensitivity of solutions depending on poorly defined weights or costs
- Min-max: Select the candidate with the *smallest maximum deviation from the optimum value,* obtained over all candidates.





We're going to hire a teacher… three were interviewed and scored…

	Modeling	Experiments	Writing
Alice	9	2	7
Barbara	4	8	6
Cameron	4	0	8

For each attribute and candidate, compute peak value and range, e.g.,

Range	5	8	2
Peak	9	8	8

Calculate deviation from peak value, normalize by range for given attribute, e.g.,

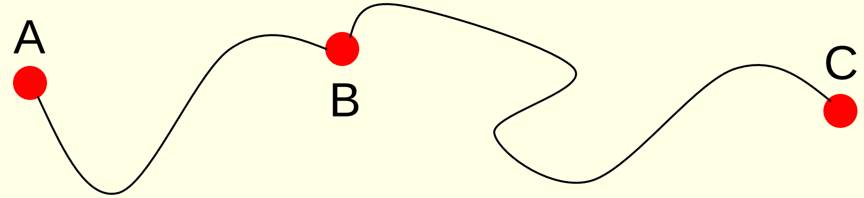
0/5	**6/8**	**1/2**
5/5	0/8	2/2
5/5	8/8	0/2

Alice has smallest normalized maximum deviation from peak values (6/8=0.75)

Alice wins by ranking first, second, and second; is it fair?

3. Dynamic Programming

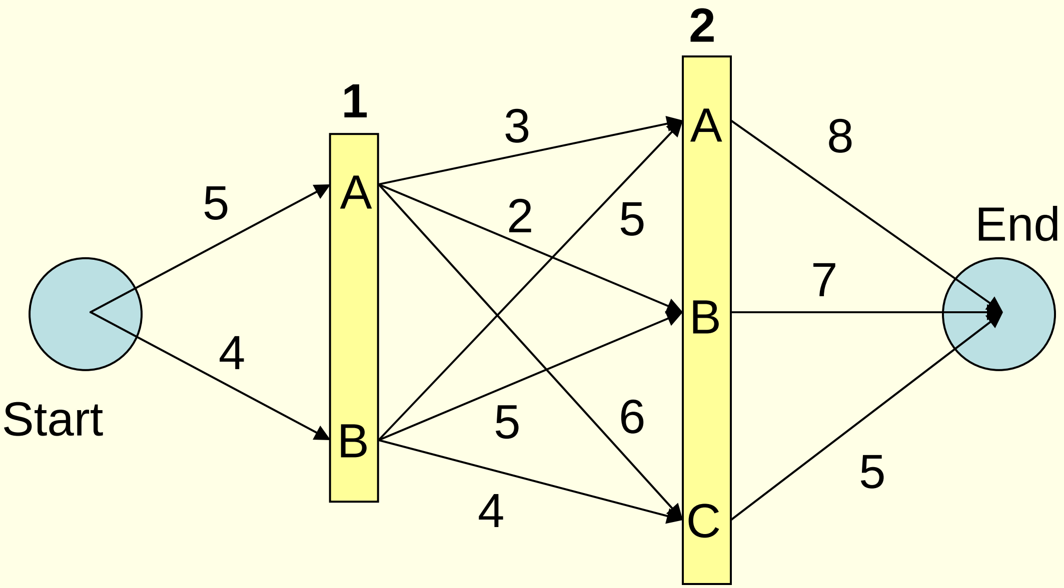
- Optimal *sequences* or *trajectories*, e.g.,
  - *minimize a scalar cost J(x(t),u(t),t), subject to*
    dx(t)/dt = f(x(t),u(t),t).
  - *minimize the driving distance through the American highway system from Boston to Los Angeles*
  - *minimize travel time of a packet on the internet*
  - *etc…*
- Dynamic programming is at the heart of nearly all modern path optimization tools
- Key ingredient: <u>Suppose the path from A to C is optimal, and B is an intermediate point. Then the path from B to C is optimal also.</u>



Seems trivial?

Numerical Example

Brute force:
12 additions



1. Evaluate optima at Stage 1:

$[A,End]_{opt}$ = min(3 + 8 , 2 + 7 , 6 + 5) = 9, path [A,B,End]

$[B,End]_{opt}$ = min(5 + 8 , 5 + 7 , 4 + 5) = 9, path [B,C,End]

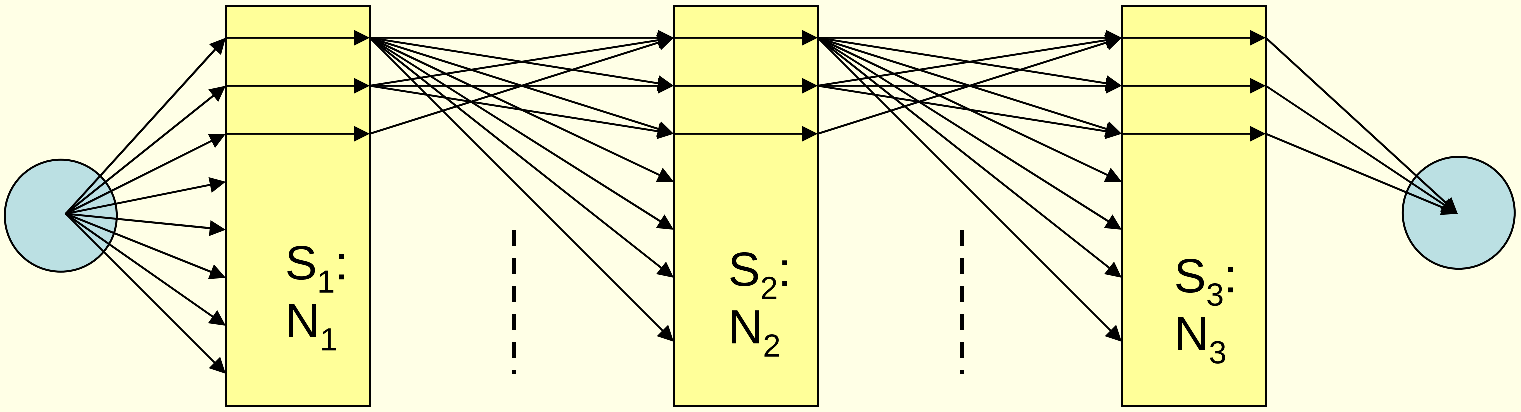
2. Evaluate optima from start:

$[Start,End]_{opt}$ = min(5 + **9** , 4 + **9**) = 13, path [Start,**B,C,End**]

Inherited values from prior optimization

→ *Total cost is 8 additions*

Power of Dynamic Programming grows dramatically with number of stages, and number of nodes per stage.



Consider three decision stages, with N_1, N_2, and N_3 choices respectively.
Total paths possible is $N_1 \times N_2 \times N_3$. To evaluate them all costs $3N_1N_2N_3$ additions.

Dynamic programming solution:

At stage 2, evaluate the best solution from each node in S_2 through S_3 to the end:
$N_2\ N_3$ additions. Store the best path from each node of S_2.

At stage 1, evaluate the best solution from each node in S_1 through S_2 to the end;
$N_1\ N_2$ additions. Store the best path from each node of S_1.

At start, evaluate best solution from start through N_1 to the end;
N_1 additions. Pick the best path!

Total burden is $\mathbf{N_2(N_1+N_3)+N_1}$ additions vs. $\mathbf{3\ N_1N_2N_3}$ additions.

GENERAL CASE: $N^2(S-1)+N$ vs. SN^S for S stages of N nodes each

4. Lagrange Multipliers

Let $\underline{x}$ be a n-dimensional vector – the parameter space

Let $\underline{f}(\underline{x})$ be a vector of m functions that are functions of $\underline{x}$ - constraints

SOLVE: min $C(\underline{x})$ subject to constraints $\underline{f}(\underline{x}) = \underline{0}$

Without the constraints, we can solve the n equations $\delta C / \delta x_i = 0$, because at the optimum point $\underline{x}^*$, $C(\underline{x}^*)$ is *flat*.

But in the presence of the constraints, we know only that

$\delta C(\underline{x}^*) = 0$ and $\delta f_k(\underline{x}^*) = 0$ or:

$\Sigma_i \, [\delta C / \delta x_i] \, dx_i = 0$ and $\Sigma_i \, [\delta f_k / \delta x_i] \, \delta x_i = 0$ *(m+1 equations)*

Lagrange Multipliers cont.

Use m *Lagrange multipliers* λ to augment the cost function:

$C'(x) = C(\underline{x}) + \Sigma_k \lambda_k f_k(\underline{x})$

NOTE $\underline{\lambda}$ CAN TAKE ARBITRARY VALUES BY DESIGN

$\delta C' = \delta C + \Sigma_k \lambda_k \delta f_k = \Sigma_i [\delta C/\delta x_i + \Sigma_k \lambda_k \delta f_k/\delta x_i] \delta x_i$

At optimum $\underline{x}$*, we have $\delta C' = 0$; Each [ ] has to be zero, so we get n equations: $\delta C/\delta x_i + \Sigma_k \lambda_k \delta f_k/\delta x_i = 0, \quad i = 1,\ldots,n$

We already had m equations: $f_k(\underline{x}^*) = 0, \quad k = 1, \ldots,m$

Solve the (m+n) equations for the n elements of $\underline{x}$* and the m values of $\underline{\lambda}$

2.017J Design of Electromechanical Robotic Systems
Fall 2009