

# Applying Optimization: Three 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  $[x_1, x_2, x_3, \dots, x_n]$  – 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.,

$$3x_1 - 4x_3 + x_7 \leq 30.$$

- A total of  $I+E$  constraint equations for  $n$  parameters. Obviously,  $I+E \geq n$  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, and gives 61MJ/kg heat energy
- Coal has 2.80% sulfur, costs \$28/kg, 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

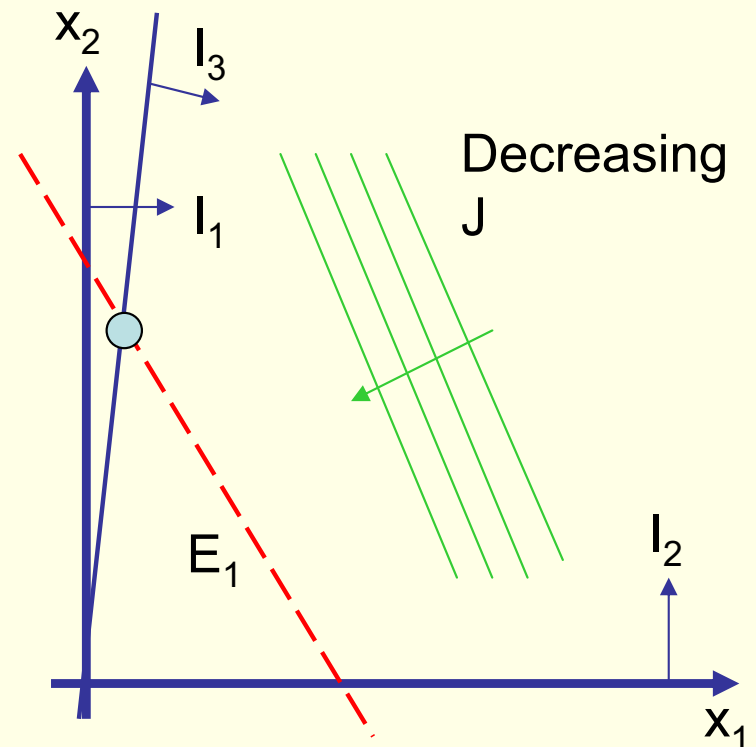
$x_2$  = kg of coal to burn per second

$$J = 55x_1 + 28x_2 \quad (\text{cost})$$

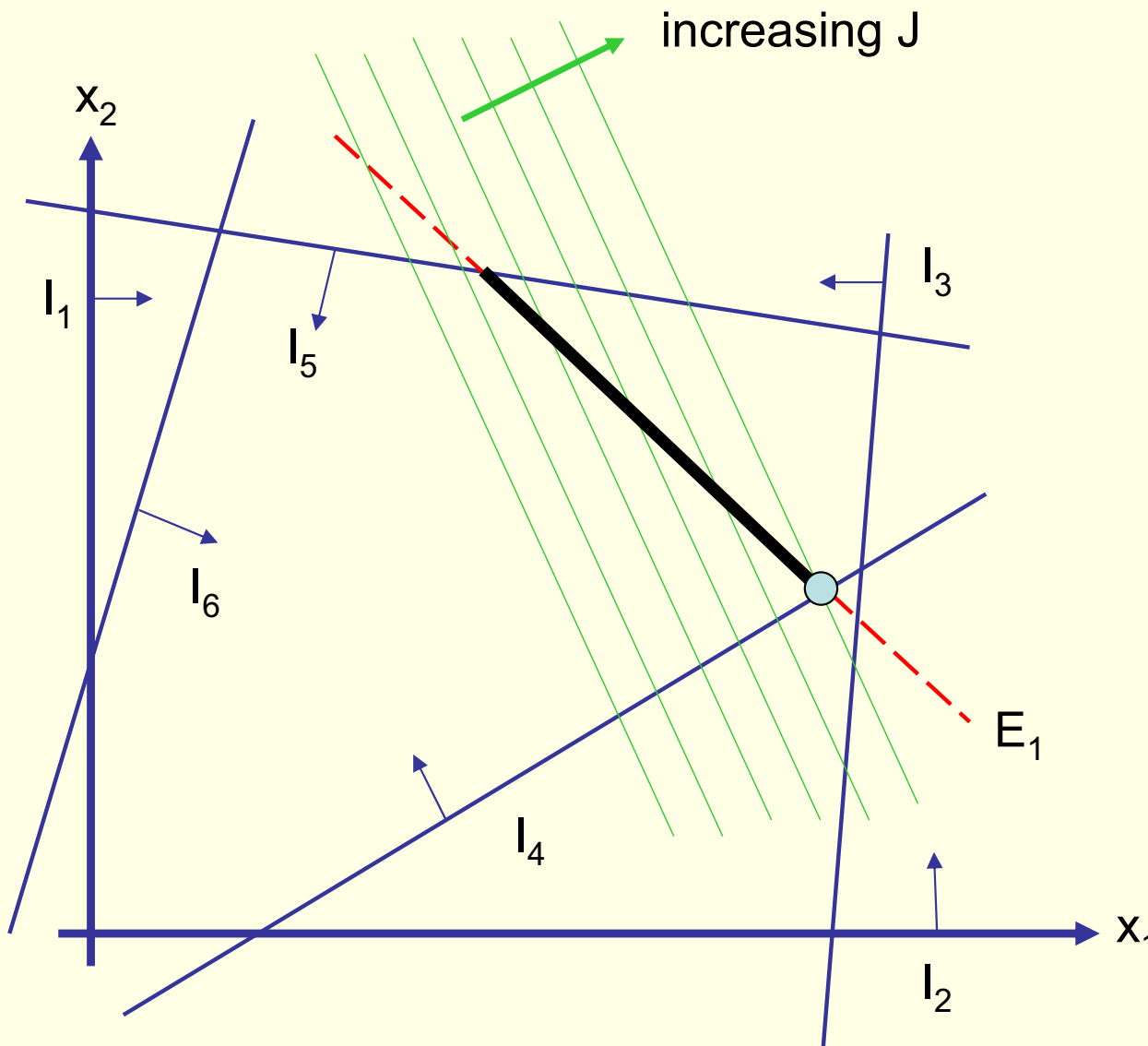
$$61x_1 + 38x_2 = 4 \quad (E_1)$$

$$0.12x_1 + 2.8x_2 \leq 2.5(x_1 + x_2) \\ \rightarrow x_2 \leq 8x_1 \quad (I_3)$$

**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.

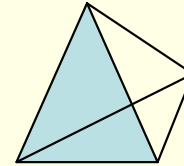
How many vertices are there to consider?

$N =$  “Combinations of  $n$  items from a collection of  $I+E$  items”  $\rightarrow$

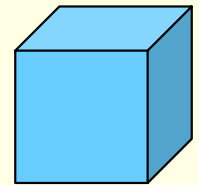
$$N = (I+E)! / n! (I+E-n)!$$

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 ( $n=5$ );

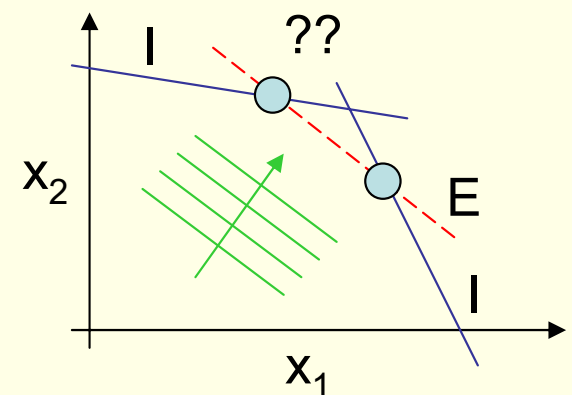
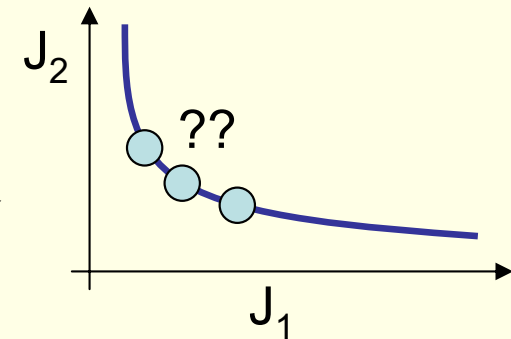
If  $I+E = 10$ ,  $N \sim 250$  (still quite reasonable for calculations!)

1. Step through *all sets 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.



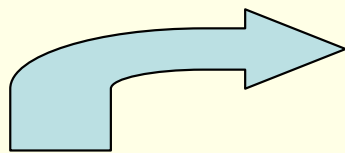
We're going to hire a teacher... three were interviewed and scored...

	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.,



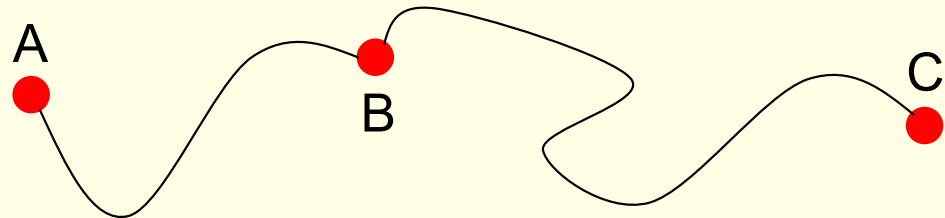
<b>0/5</b>	<b>6/8</b>	<b>1/2</b>
5/5	0/8	2/2
5/5	8/8	0/2

**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

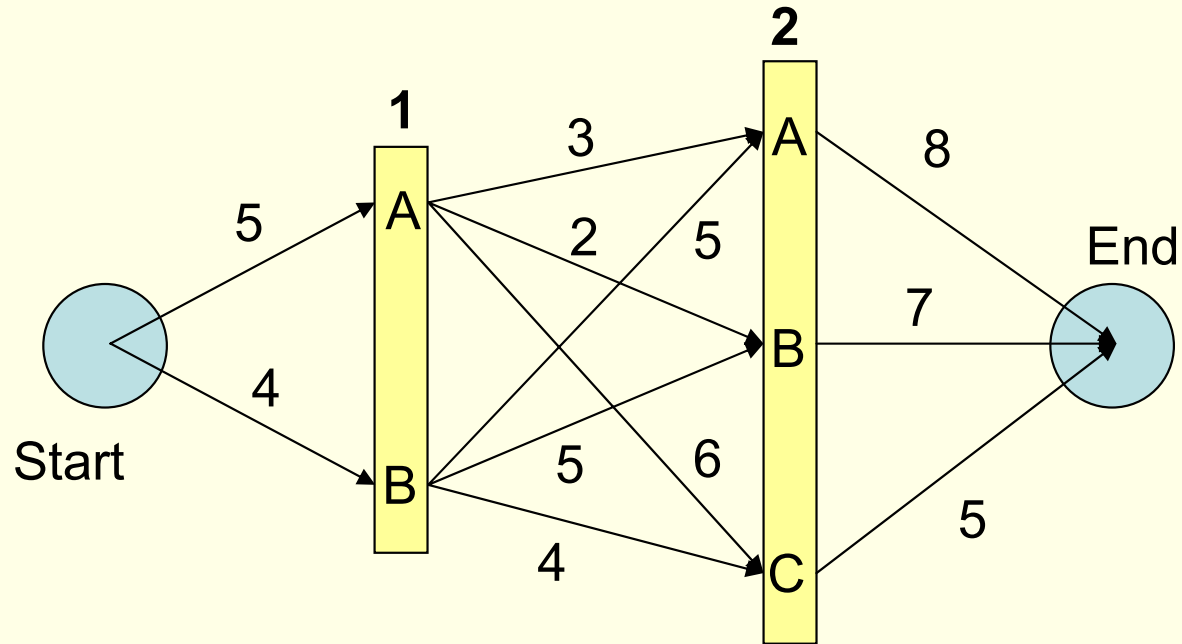
- Optimal sequences or trajectories, e.g.,
  - minimize a scalar cost  $J(x(t), u(t), t)$ , subject to  $dx(t)/dt = f(x(t), u(t), t)$ .
  - minimize the driving distance through the American highway system from Boston to Los Angeles
  - minimize travel time of a packet on the internet
  - etc...
- Dynamic programming is at the heart of nearly all modern path optimization tools
- Key ingredient: 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.



Seems trivial?

# Numerical Example

Brute force:  
12 additions



1. Evaluate optima at Stage 1:

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

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

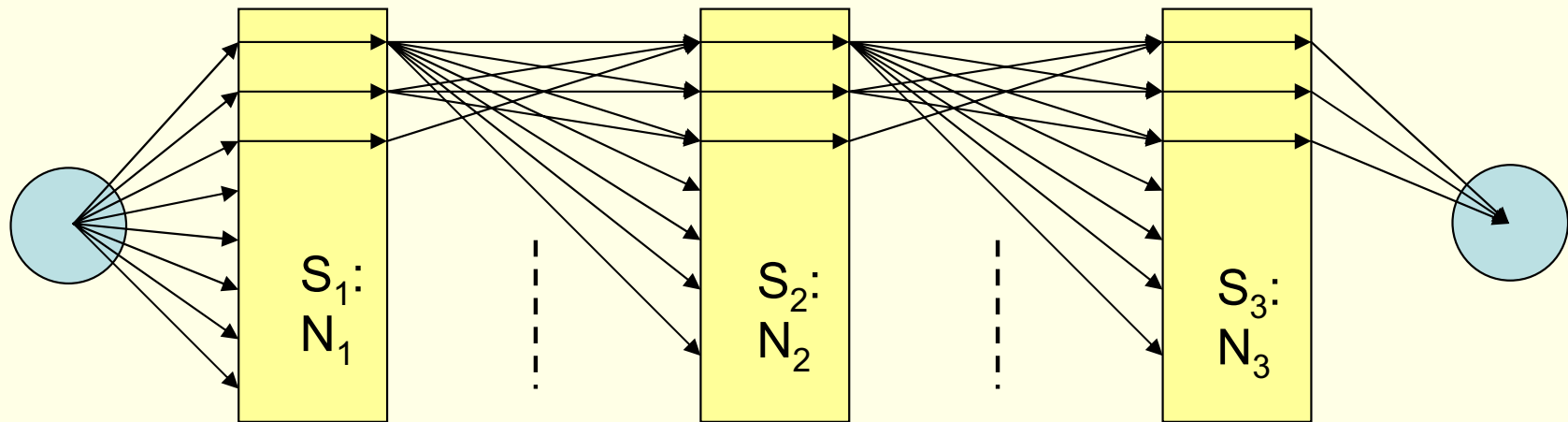
2. Evaluate optima from start:

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

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  $N_2(N_1+N_3)+N_1$  additions vs.  $3 N_1N_2N_3$  triple additions.

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