Lecture 2: Classical Optimization

- Single variable numerical optimization
  - Direct search methods (Bounding phase and Golden section search)
  - Gradient search methods (Newton's search)
  - Point estimation method
- Multi-variable numerical optimization
  - Direct search methods (simplex method, Hooke-Jeeves Pattern search)
  - Gradient search methods (Steepest descent, Fletcher-Reeves and DFP method)

Constrained Optimization by Classical Means

- Constrained optimization
  - Penalty function approach
  - Linearized methods using LP solvers
  - Specific equality constraint handling methods
- Structured optimization
  - Geometric Programming
  - Dynamic programming
  - Integer programming

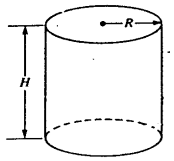
Nonlinear Programming (NLP) Problem

- $\boldsymbol{x}$ is a n-dimensional vector
- Solve the following problem:

$$\begin{aligned} \text{Minimize} \quad & f(\boldsymbol{x}) \\ \text{Subject to} \quad & g_j(\boldsymbol{x}) \geq 0 \qquad j = 1,2,\ldots,J \\ & h_k(\boldsymbol{x}) = 0 \qquad k = 1,2,\ldots,K \\ & x_i^L \leq x_i \leq x_i^U \qquad i = 1,2,\ldots,n \end{aligned}$$

An Example

- Design of a Beer Can
- Minimize cost of manufacturing
- (directly related to surface area)
- Can must hold at least a
- specified amount (say, 400 ml)
- Handling, shipping, aesthetic, and fabrication limitation
- Diameter of can not be more than 8 cm and less than 3.5 cm
- Height of can not be more than 18 cm and less than 8 cm

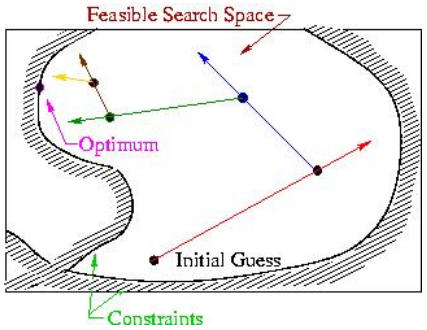


Beer Can Example (cont.)

- Problem parameters: Can thickness, can diameter, can height, can shape (circular or square), can material, fabrication process, refrigerated temperature, reuse, etc.
- Choose variables: diameter (D) and height (H)
- Circular from an available AL sheet
- Constraint: g(D,H): $\pi D^2H/4 \geq 300$ (ml)
- Objective function: f (D,H) = $\pi DH + \pi D^2/2$ (cm^2)
- Variable bound: $3.5 \leq D \leq 8$, $8 \leq H \leq 18$ (cm)
- Optimization problem has 2 variables and 1 inequality constraint

Classical Optimization Methods

- grad f(x) = 0
- Direct methods
  - No use of any gradient or other information
- Gradient-based methods
  - Explicit use of first and second-order gradients



Uni-directional Search

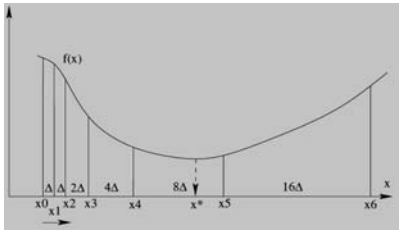
- A multi-variable problem, but the search is restricted along a direction
  - Single-variable search
  - $\mathbf{x}(a) = \mathbf{x}^{(t)} + a^*\mathbf{s}^{(t)}$, a is the only variable
- Repeatedly used in multi-variable optimization
- Need a quick single-variable search method

Single-Variable Optimization

- Only one variable
- Two-step method for direct search
  - Rough bounding: Bounding phase method
  - Accurate bounded search: Golden section search
- Assumption: Existence of only one optimum

Bounding Phase Method:
Coarse Bounding of Optimum

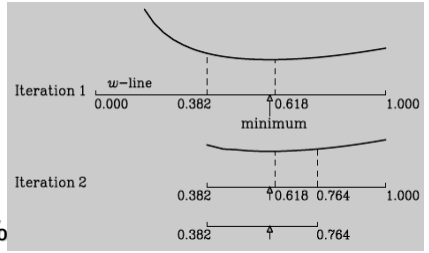
- Start with $x^{(0)}$ and Δ
- Determine sign of Δ, k=1
- $x^{(k)} = x^{(k-1)}+2^k \Delta$
- Compare $f(x^{(k)})$ and $f(x^{(k-1)})$
- Continue till no more improvement
- Exponentially approach minimum



Bound: $(x^{(k-2)}, x^{(k)})$

Golden Section Search:
Finding Minimum Prcisely

- Search within bounds obtained by rough bounding
- Golden number (0.618)
- Every iteration eliminates 38.2% of search space
- Most efficient on a serial processor



What strategy is best for a parallel computer?

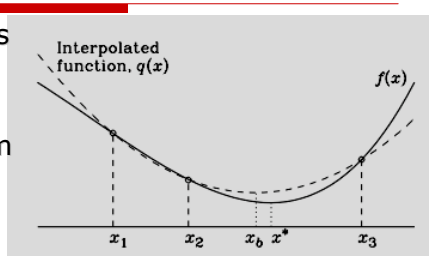
Gradient Based Method:
Single-variable search

- Each gradient takes two evaluations
- Central difference method:

  $df/dx \approx (f(x+\delta x)- f(x-\delta x))/(2*\delta x)$
- $\delta x=0.01|x|$, if $|x|<0.01$, else 0.0001
- $(\delta x)^2$ accuracy
- Newton's search requires the second derivative -> 3 function evaluations
  - Better accuracy with more evaluations

Point Estimation Method

- Take three points and compute an approximate model
- Find the optimum of the model
- Develop an algorithm
- A primitive version of meta-modeling technique



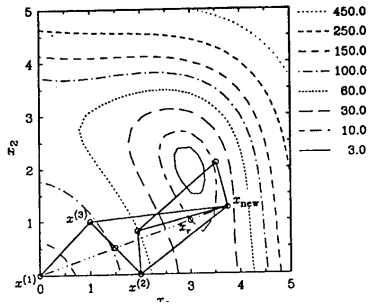
Extension to multiple variables computationally expensive!

Multi-Variable Optimization

- Direct search methods do not use any gradients
  - Nelder and Meade's simplex search method
  - Hooke-Jeeves Pattern search methods
  - Powell's conjugate direction method
- Gradient based methods
  - Steepest descent, Newton's, Quasi-Newton's methods

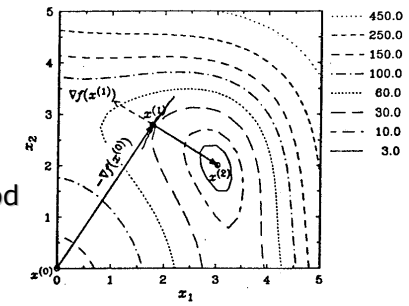
Simplex Search Method

- Simplex search method
- Conjugate direction method
- Randomized methods
- Hill-climbing methods



Gradient-Based Methods

- Steepest-descent method
  - Reduces the function value the maximum
  - For convex problems, probably best
- Conjugate gradient method
- Quasi-Newton method
  - BFGS, DFP



Gradient-Based Methods (cont.)

- Gradient computation is expensive
- Gradient vector takes 2n soln. evals.
- Hessian matrix takes $2n^2$ +1 soln. evals.
- Quasi-Newton's methods are good compromise for time and accuracy
  - Attempt to simulate Hessian matrix using gradient vectors at successive iterations
  - DFP, BFGS adopted by MATLAB etc.

Constrained Optimization

- Decision variables: $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$
- Constraints restrict some solutions to be feasible

Min. $f(\boldsymbol{x})$
s.t. $g_j(\boldsymbol{x}) \geq 0 \quad j = 1,2,\ldots,J$
$h_k(\boldsymbol{x})=0 \quad k = 1,2,\ldots,K$
$x_i^L \leq x_i \leq x_i^U \; i = 1,2,\ldots,n$

- Equality and inequality constraints
- Minimum of $f(\boldsymbol{x})$ need not be constrained minimum
- Constraints can be non-linear

Constraint Handling

- Theory: KKT points are likely to be optimal

$$\nabla f(x) - \sum_{j=1}^{J} u_j \nabla g_j(x) - \sum_{k=1}^{K} v_k \nabla h_k(x) = 0$$

$g_j(\boldsymbol{x}) \geq 0, \quad j = 1,2,\ldots,J$
$h_k(\boldsymbol{x})=0, \quad k = 1,2,\ldots,K$
$u_j g_j(\boldsymbol{x})=0, \quad j = 1,2,\ldots,J$
$u_j \geq 0, \quad j = 1,2,\ldots,J$

$\Delta f^*/\Delta b_j$ (shadow price)

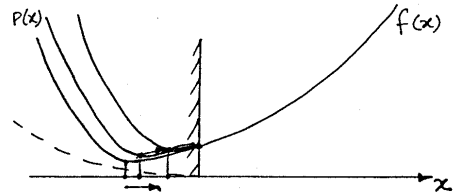
- Find $(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v})$ satisfying all above
- $\boldsymbol{u},\boldsymbol{v}$ are Lagrange multiplier (important to know)

Constraint Handling Algorithms

- Penalty function method

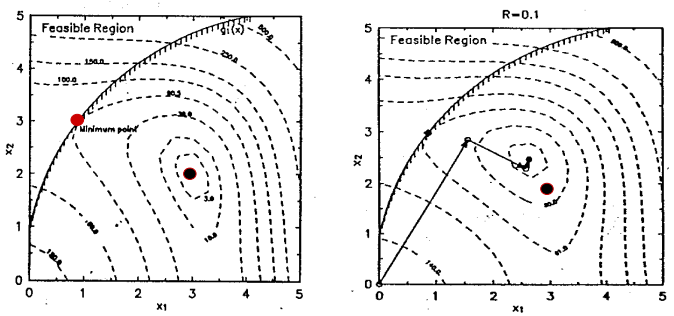
$$P(x, R) = f(x) + \Omega(R, g(x), h(x))$$

- Start with a small R, then increase
- Solve each unconstrained problem

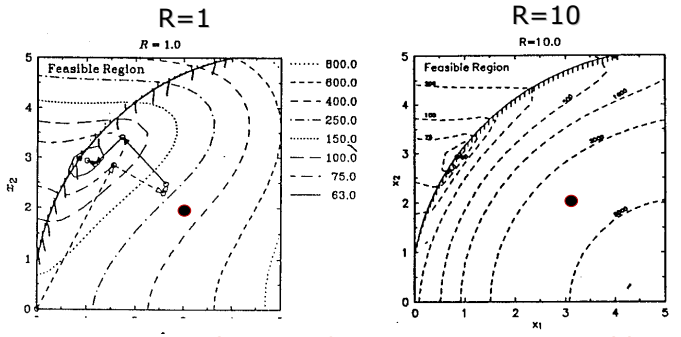


An Example Problem

R=0 R=0.1



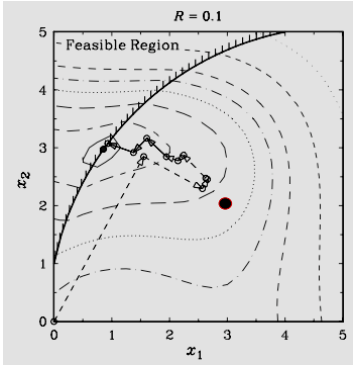
Penalty Function Approach (cont.)



Distortion in objective function is a major problem

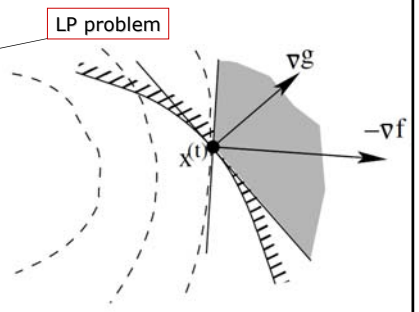
Augmented Lagrangian Method

- Penalty update is different
  - Expressions are complex
- Not much distortion in the penalized function
- In addition, the resulting solution is a KKT point
- Lagrange multiplier are by-products



Other Specialized Algorithms

- Feasible Direction Method
  - Direction both feasible and descent
- Cutting plane method
- Generalized reduced gradient method
  - Move along constraint boundaries

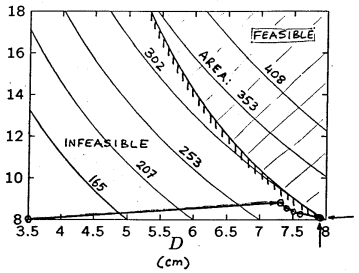


Beer Can Example

- Problem parameters: Can thickness, can diameter, can height, can shape (circular or square), can material, fabrication process, refrigerated temperature, reuse, etc.
- Choose variables: diameter (D) and height (H)
- Circular from an available AL sheet
- Constraint: $g(D,H)$: $\pi D^2H/4 \geq 400$
- Objective function: $f(D,H) = \pi DH + 2\pi D^2/4$
- Variable bound: $3.5 \leq D \leq 8$, $8 \leq H \leq 18$
- Optimization problem has 2 variables and 1 inequality constraint

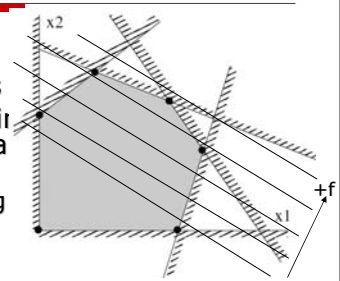
Beer Can Optimization

- Final solution is
D = 7.956 and
H = 8.048 cm,
- Volume 400 ml
- Surface area
300.6 cm^2
- Can we fabricate it?
- Use discrete values
D: (3.5, 4, 4.5, 5,
5.5, 6, 6.5, 7, 7.5, 8)



Structured Algorithms: *Linear Programming*

- Optimization problems having linear objective and constraint functions
- Optimum is a corner poi or multiple point along a edge
  - Search restricted among corner points (Simplex Search)
  - Search inside feasible region (Interior point method)



For small sized problems, use 'simplex search'

Integer Programming Methods

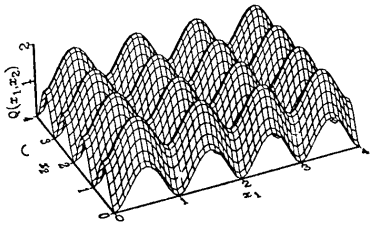
- Many design variables take integer values only
- Example: Number of teeth in a gear, pipe diameter may come in discrete dimensions, etc.
- Search space becomes discrete and discontinuous
- Gradient methods can not be applied directly (Example: Material)
- Direct search methods also fail
  - Points dictated by the algorithm may not be available

Integer Programming (cont.)

- One simple-minded strategy:
  - Assume design variables can take any real value
  - Round-off to nearest integer solutions
- Example: Two integer variables (23.76,4.12)
- Compare (23,4), (23,5), (24,4), and (24,5)
- For 10 integer variables compare 210 or 1,024 designs

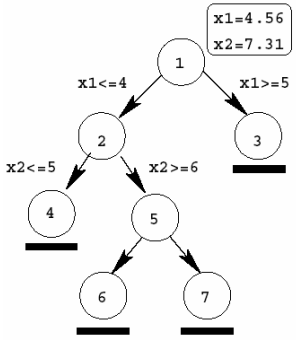
IP Algorithms

- Penalty function method
- Introduces multi-modalities
- Not very popular



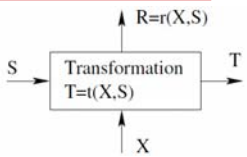
Branch-and-Bound Method

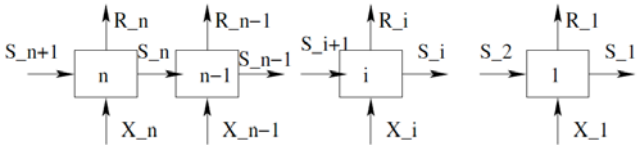
- Treat problem without integer restrictions
- Branch into two problems with an additional constraint each
- Bound:
  - Infeasible solution
  - Integer solution
  - Best function value is worse than an integer solution



Dynamic Programming

- Applicable to a multi-stage decision process
- Optimize each stage with a parameter S
- Min Σ R_i(S_i+1,X_i)
- Constraints: S_i=t_i(X_i,S_i+1)





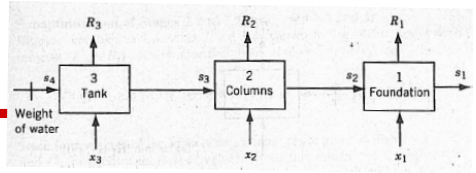
Initial Value Problem

- S_n+1 is given, find all X_i
- Optimization of subsystems
  - start from subsystem 1 (parametric optimization with S_2)
    - know x_1*, R_1* (=f_1*), S_1* as a function of S_2
  - Move to subsystem 2 (parametric with S_3): Min R_2(X_2,S_3)+f_1*(s_2)
    - Know X_2*,R_2*,S_2* as a function of S_3
  - Carry on till the end (i=n)
- Here, S_n+1 is given, make a forward pass to determine all S_i*, X_i* and R_i*

Final Value and Boundary Value Problems

- *Final value problem*: S_1 is given
  - If inverse t' is known, possible to do
- *Boundary value problem*: S_n+1 and S_1 are given
  - Solution may not exist, if exists it is a complex task
- *Curse of Dimensionality* comes against the use of the dynamic programming method

Tank Design Problem (3 Stages)



- Tank Data

Taken from SS Rao's book

TABLE 9.1 Component 3 (Tank)

Type of Tank	Load Acting on the Tank, s_4 (kg_f)	R_3 Cost ($)	Self-Weight of the Component (kg_f)	$s_3 = s_4$ + Self-Weight (kg_f)
(a) Cylindrical RCC tank	100,000	5,000	45,000	145,000
(b) Spherical RCC tank	100,000	8,000	30,000	130,000
(c) Rectangular RCC tank	100,000	6,000	25,000	125,000
(d) Cylindrical steel tank	100,000	9,000	15,000	115,000
(e) Spherical steel tank	100,000	15,000	5,000	105,000
(f) Rectangular steel tank	100,000	12,000	10,000	110,000
(g) Cylindrical RCC tank with hemispherical RCC dome	100,000	10,000	15,000	115,000

Column Data

TABLE 9.2 Component 2 (Columns)

Type of Columns	s_3 (kg_f)	R_2 Cost ($)	Self-Weight (kg_f)	$s_2 = s_3$ + Self-Weight (kg_f)
(a) RCC columns	150,000	6,000	70,000	220,000
	130,000	5,000	50,000	180,000
	110,000	4,000	40,000	150,000
	100,000	3,000	40,000	140,000
(b) Concrete columns	150,000	8,000	60,000	210,000
	130,000	6,000	50,000	180,000
	110,000	4,000	30,000	140,000
	100,000	3,000	15,000	115,000
(c) Steel columns	150,000	15,000	30,000	180,000
	130,000	10,000	20,000	150,000
	110,000	9,000	15,000	125,000
	100,000	8,000	10,000	110,000

Foundation Data

TABLE 9.3 Component 1 (Foundation)

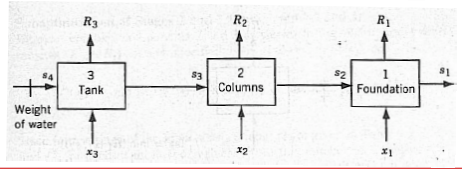
Type of Foundation	s_2 (kg_f)	R_1 Cost ($)	Self-Weight (kg_f)	$s_1 = s_2$ + Self-Weight (kg_f)
(a) Mat foundation	220,000	5,000	60,000	280,000
	200,000	4,000	45,000	245,000
	180,000	3,000	35,000	215,000
	140,000	2,500	25,000	165,000
	100,000	500	20,000	120,000
(b) Concrete pile foundation	220,000	3,500	55,000	275,000
	200,000	3,000	40,000	240,000
	180,000	2,500	30,000	210,000
	140,000	1,500	20,000	160,000
	100,000	1,000	15,000	115,000
(c) Steel pile foundation	220,000	3,000	10,000	230,000
	200,000	2,500	9,000	209,000
	180,000	2,000	8,000	188,000
	140,000	2,000	6,000	146,000
	100,000	1,500	5,000	105,000

Dynamic Optimization

- Stage 1 (Foundation) optimization

Specific Value of s_2 (kg_f)	x_1^* (Type of Foundation for Minimum Cost)	f_1^* ($)	Corresponding Value of s_1 (kg_f)
220,000	(c)	3,000	230,000
200,000	(c)	2,500	209,000
180,000	(c)	2,000	188,000
140,000	(b)	1,500	160,000
100,000	(a)	500	120,000

No idea which s2 Will be optimum at this stage



Stage 2 (column) optimization

No idea which s3 will be optimum at this stage

Specific Value of s_3 (kg_f)	Value of x_2 (Type of Columns)	Cost of Columns, R_2 ($)	Value of the output state variable s_2 (kg_f)	x_1^* (Type of Foundation)	f_1^* ($)	$R_2 + f_1^*$ ($)
150,000	(a)	6,000	220,000	(c)	3,000	9,000
	(b)	8,000	210,000	(c)	2,750**	10,750
	(c)	15,000	180,000	(c)	2,000	17,000
130,000	(a)	5,000	180,000	(c)	2,000	7,000
	(b)	6,000	180,000	(c)	2,000	8,000
	(c)	10,000	150,000	(b)	1,625**	11,625
110,000	(a)	4,000	150,000	(b)	1,625**	5,625
	(b)	4,000	140,000	(b)	1,500	5,500
	(c)	9,000	125,000	(b)	1,125**	10,125
100,000	(a)	3,000	140,000	(b)	1,500	4,500
	(b)	3,000	115,000	(a)	875**	3,875
	(c)	8,000	110,000	(a)	750**	8,750

Summary:

Specified Value of s_3 (kg_f)	Type of Columns Corresponding to Minimum Cost of Stages 2 and 1, (x_2^*)	Minimum Cost of Stages 2 and 1, f_2^* ($)	Value of the Corresponding State Variable, s_2 (kg_f)
150,000	(a)	9,000	220,000
130,000	(a)	7,000	180,000
110,000	(b)	5,500	140,000
100,000	(b)	3,875	115,000

** indicates interpolation

Stage 3 (tank) optimization

Specific Value of s_4 (kg_f)	Type of Tank (x_3)	Cost of Tank R_3 ($)	Corresponding Output State, s_3 (kg_f)	x_2^* (Type of Columns for Minimum Cost)	f_2^* ($)	$R_3 + f_2^*$ ($)
Supplied-> 100,000	(a)	5,000	145,000	(a)	8,500**	13,500
	(b)	8,000	130,000	(a)	7,000	15,000
	(c)	6,000	125,000	(a)	6,625**	12,625
	(d)	9,000	115,000	(b)	5,875**	14,875
	(e)	15,000	105,000	(b)	$4,687\frac{1}{2}$**	$19,687\frac{1}{2}$
	(f)	12,000	110,000	(b)	5,500	17,500
	(g)	10,000	115,000	(b)	5,875**	15,875

Now, go back with s3=125K

- Final solution:

  x3*=type (c) tank, x2*=type (a) column, x1*=type (c) foundation, Cost=$12,625, s3=125,000, s2=170,000, s1=181,000 kgf, R3=$6,000, R2=$4,750, R1=$1,875

Duality Theory in Optimization

- By using constraints, a primal problem has an equivalent dual problem
- Dual problem is always concave
  - Comparatively easier to solve
- Theoretical results:
  - Convex problems and in some special cases:
    - Optimal primal and dual function values are same
  - Generic cases:
    - Optimal dual function value underestimates optimal primal function value
- Beneficial to use duality theory in many problems

Summary

- Foundation based on mathematics
  - Must-know for everyone interested in optimization
- Structured algorithms
  - If applicable, no competitor, use them
- Unfortunately, practice is full of exceptions
- One needs to be open-minded, but first application must be a classical one
- Many theories of optimization can be utilized in more flexible methods

