Evolutionary Practical Optimization

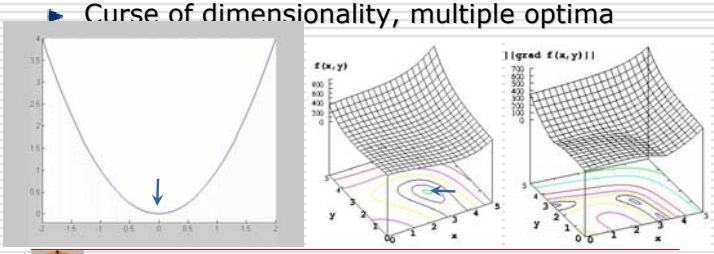
Kalyanmoy Deb
Professor of Mechanical Engineering
Indian Institute of Technology Kanpur
Kanpur, PIN 208016, India
Email: deb@iitk.ac.in
http://www.iitk.ac.in/kangal/deb.htm

Outline of Tutorial

- Optimization fundamentals
- Scope of *optimization* in practice
- Classical *point-by-point* approaches
- Advantages of evolutionary *population-based* approaches
- Scope of evolutionary approaches in different problem solving tasks
  - Having one algorithm for various practical optimizations is difficult
- Final words

Fundamentals of Optimization

- A generic name for minimization and maximization of a function *f(**x**)*
- Everyone knows: *df/dx=0* or $\nabla f(x) = 0$
- Curse of dimensionality, multiple optima

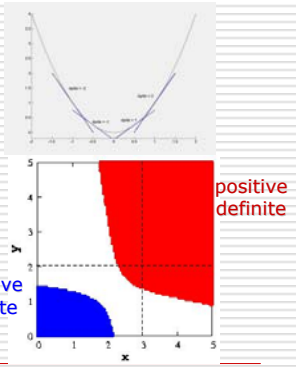


Fundamentals (cont.)

- Concept relates to mathematics
  - Second and higher-order derivatives
    d²f/dx²>0, minimum
    d²f/dx²<0, maximum

if $\nabla^2 f$ is positive definite at x*, it is a minimum

- Convex: One optimum



Constrained Optimization Basics

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

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

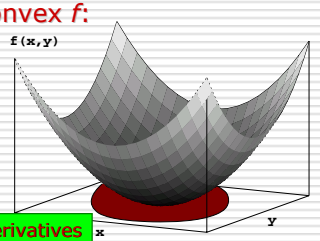
- Equality and inequality constraints
- Minimum of $f(\boldsymbol{x})$ need not be constrained minimum
- Constraints can be non-linear

Constrained Optimization Basics (cont.)

- Kuhn-Tucker (K-T) conditions for optimality
  - First-order necessary conditions
  - Convex search space, convex f: K-T point is minimum

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

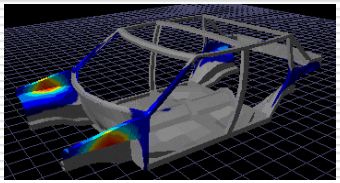
$g_j(\boldsymbol{x}) \geq 0 \quad j = 1,2,\ldots,J$
$h_k(\boldsymbol{x})=0 \quad k = 1,2,\ldots,K$
$u_j g_j(\boldsymbol{x})=0 \quad j = 1,2,\ldots,J$
$u_j \geq 0 \quad j = 1,2,\ldots,J$

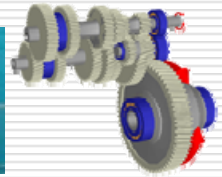


Involve Derivatives and solve for roots

Scope of Optimization in Practice

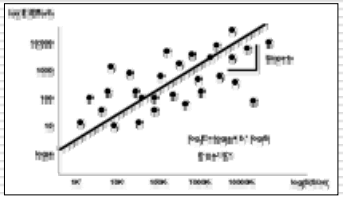
- *Optimal design & manufacturing* for desired goals
- Major application in engineering



Scope of Optimization (cont.)

- *Parameter optimization* for optimal performance

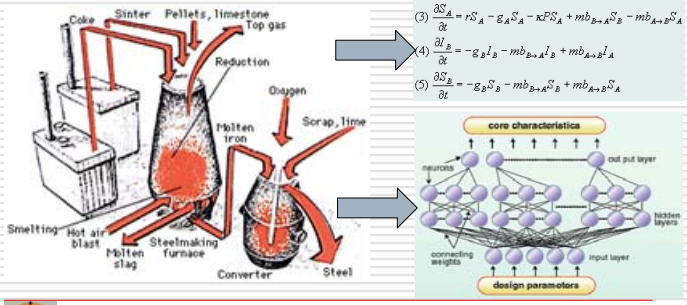


- Scientific experiments, computer experiments


Scope of Optimization (cont.)

- *Modeling* (system, process)

$$(1)\ \frac{\partial P}{\partial t} = -\gamma P + \beta U_A - \kappa P S_A$$
$$(2)\ \frac{\partial l_A}{\partial t} = r l_A - g_A l_A + \kappa P S_A - \lambda l_A + m b_{B\to A} l_B - m b_{A\to B} l_A$$
$$(3)\ \frac{\partial S_A}{\partial t} = r S_A - g_A S_A - \kappa P S_A + m b_{B\to A} S_B - m b_{A\to B} S_A$$
$$(4)\ \frac{\partial l_B}{\partial t} = -g_B l_B - m b_{B\to A} l_B + m b_{A\to B} l_A$$
$$(5)\ \frac{\partial S_B}{\partial t} = -g_B S_B - m b_{B\to A} S_B + m b_{A\to B} S_A$$



Scope of Optimization (cont.)

- *Scheduling and planning* (combinatorial optimization)

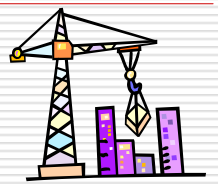
Timetabling

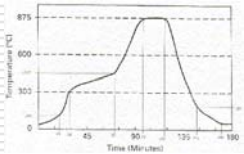
Routing & Scheduling

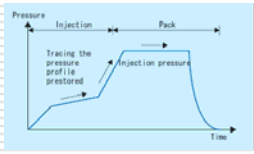
Traveling Salesperson problem

Scope of Optimization (cont.)

- *Optimal control*
- Time-variant profiles are to be found
  - How to lower load?
  - How to control temp, pressure?

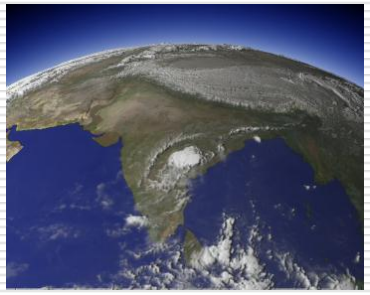






Scope of Optimization (cont.)

- *Forecasting and prediction*



Scope of Optimization (cont.)

- *Data mining* (classification, clustering, pattern recognition)

Scope of Optimization (cont.)

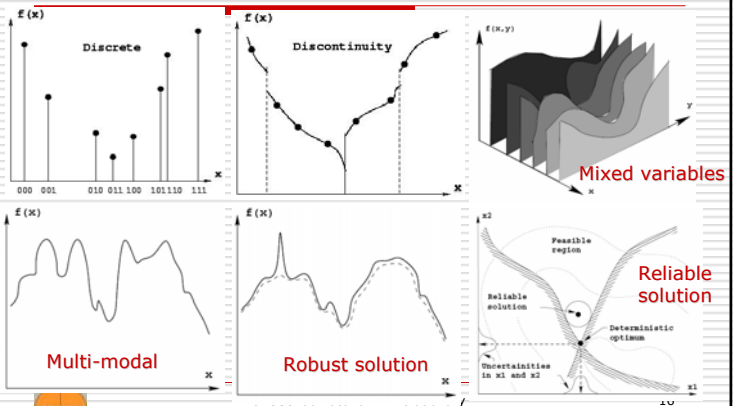
- *Machine learning*
  - Designing intelligent systems

Developed at KanGAL

Properties of Practical Optimization Problems

- Non-differentiable functions and constraints
- Discontinuous search space
- Discrete search space
- Mixed variables (discrete, continuous, permutation)
- Large dimension (variables, constraints, objectives)
- Non-linear constraints
- Multi-modalities
- Multi-objectivity
- Uncertainties in variables
- Computationally expensive problems
- Multi-disciplinary optimization

Different Problem Complexities



Classical Optimization Methods and Past

- Exact differentiation & root-finding method
  - Intractable and not sufficient for practical problems

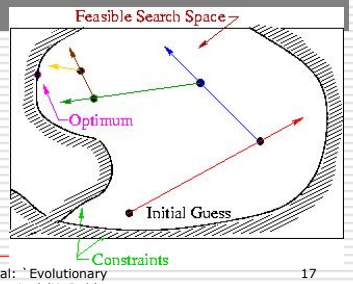
Linear regression:

y=mx+b

$$m = \frac{n\sum(xy) - \sum x \sum y}{n\sum(x^2) - (\sum x)^2}$$

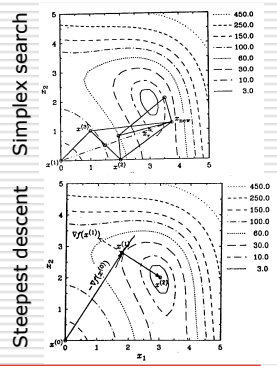
$$b = \frac{\sum y - m\sum x}{n}$$

- Numerical algorithms
  - Iterative and deterministic
  - Directions based on gradients (mostly)
- Point-by-point approaches



Classical Methods (cont.)

- Direct and gradient based methods
- Convexity assumption
  - No guarantee otherwise
- Local perspectives
- Discreteness cause problems
- Non-linear constraints
- Large-scale application time-consuming
- Serial in nature



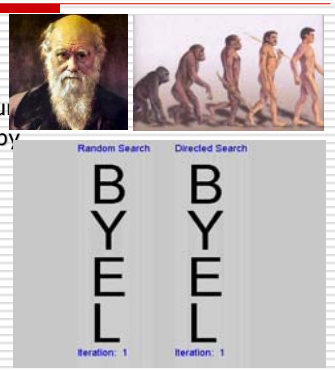
Practical Optimization

With a point in each iteration, scope is limited

A need for non-classical, population-based optimization methods exists

Evolutionary Optimization: A Motivation from Nature

- Natural evolution + genetics
  - Guided search procedu
  - Offspring are created by duplication, mutation, crossover etc.
  - Good solutions are retained and bad are deleted
  - Information is coded



Nature as an Optimizer

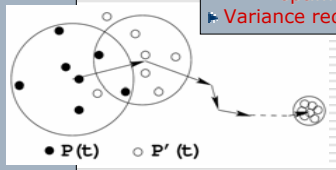
- Nature as structural engineer
  - Stem, Bamboo, insect trachea, bee-hive
- Nature as a CFD solver
  - Birds, fishes
- Nature as a drag reducer
  - Penguin body

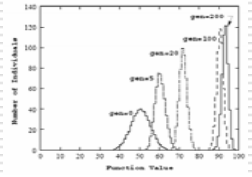



Evolutionary Algorithms as Optimizers

- Mean approaches optimum
- Variance reduces

begin
Solution Representation
t := 0; // generation counter
Initialization P(t);
Evaluation P(t);
while not Termination
do
  P'(t) := Selection (P(t));
  P''(t) := Variation (P'(t));
  Evaluation P''(t);
  P(t+1):= Survivor (P(t),P''(t));
  t := t+1;
od
end

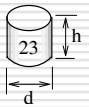




Traditional Evolutionary Approach:
Binary-Coded Genetic Algorithms

- Genetic Algorithms (John Holland, 1962)
- Design of a can for minimum cost having at most *V* volume
- Objective function: Cost(*d*,*h*): $f(d,h) = \pi dh + 2\pi d^2/4$
- Constraint: Volume (*d*,*h*): $\Pi d^2 h/4 \geq V$
- Representation in binary strings
- Fitness: objective value + penalty for constraint

Solution Representation:



(d, h) = (8, 10) cm
(Chromosome) = 0 1 0 0 0 0 1 0 1 0

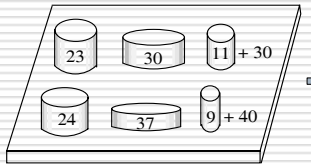
Binary-Coded Genetic Algorithm:
A Hand Simulation

Initialization and Evaluation:

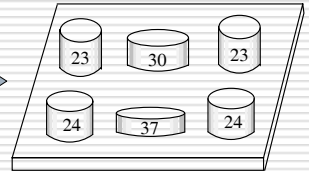
- Fitness = Cost + Penalty (proportional to constraint violation)

Selection: *Emphasizing better solutions*

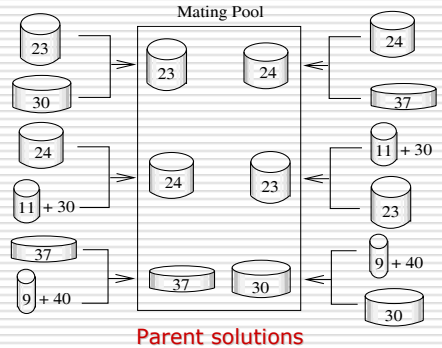
Random Initialization (N members)



Population after Selection



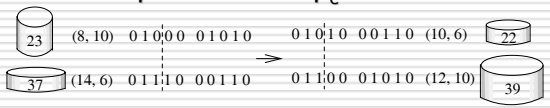
Tournament Selection Operator



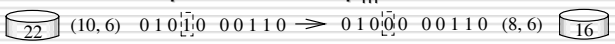
Parent solutions

Variation Operators:
Creating new offspring solutions

- Crossover operator with p_c:

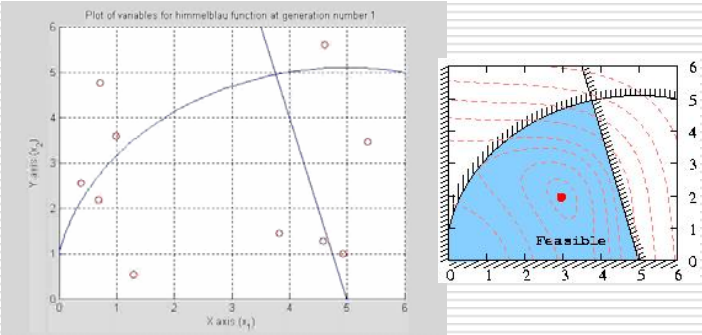


- Mutation operator with p_m:



- Good, partial information propagates leading to optimum
- Other and modified operators often used

Simulation of a GA



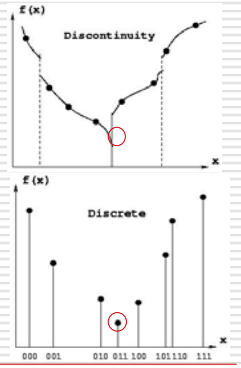
Differences with Classical Methods

- GAs work on a coding of variables
  - Mixed variable types can be handled
  - Permutation, continuous, discrete together
- GAs are population-based
  - Safety in numbers -> global perspective
  - *Implicit parallelism*
- GA operators are probabilistic
  - Less chance of getting stuck
- GAs do not use any gradient information
  - Black-box search -> wide applicability
- Ideal for parallel computation
  - Distribute evaluations among processors and more

Non-smooth Problems

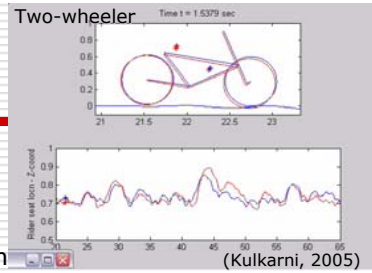
Handling discrete, discontinuous variables

- Non-differentiability, discontinuity, discreteness, non-linearity not a difficulty
- GAs work with a discrete search space anyway
  - So, GAs are natural choice for discrete problem solving

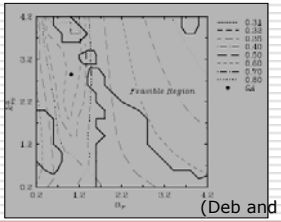


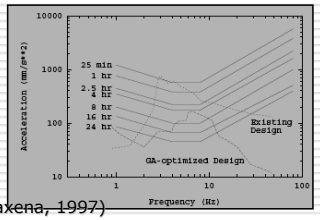
Car Suspension Design

- Practice is full of non-linearities
- MATLAB gets stuck
- An order of magnitude better than existing design



(Kulkarni, 2005)





(Deb and Saxena, 1997)

EAs for Continuous Variables

- Decision variables are coded directly, instead of using binary strings
- Recombination and mutation need structural changes
- Selection operator remains the same

Recombination

$(x_1 x_2 \ldots\ldots\ldots\ldots x_n)$

$(y_1 y_2 \ldots\ldots\ldots\ldots y_n)$

$\Rightarrow ?$

Mutation

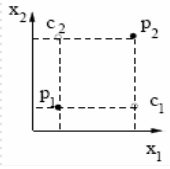
$(x_1 x_2 \ldots\ldots\ldots\ldots x_n) \Rightarrow ?$

- Simple exchanges are not adequate

Naive Recombination

- Crossing at boundaries do not constitute adequate search
  - Least significant digits are taken too seriously

15.345	3.569	$\rightarrow$	11.142	3.587
21.142	5.687		25.345	5.669



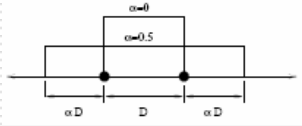
- Two Remedies:
  - Parent values (variable-wise) need to be blended to each other
  - Vector-wise recombination

Variable-wise Blending of Parents

- Use a probability distribution to create child
- Different implementations since 1991:
  - Blend crossover (BLX-α), 1991
  - Simulated binary crossover (SBX-β), 1995
  - Fuzzy recombination (FR-d), 1995
- Main feature: Difference between parents used to create children
  - Provides self-adaptive property

Blend Crossover (BLX-α)

- Uniform probability distribution within a bound controlled by α (Eshelman and Schaffer, 1991)



- Diversity in children proportional to that in parents
- Too wide a search, if parents are distant

Motivation for the SBX Operator

- Simulate processing in a binary crossover, say single-point crossover
- *Gedanken* experiment:
  - Two parent solutions in real number
  - Code them in l-bit strings
  - Use single-point crossover in all (l-1) places and find children strings
  - In each case map strings back to real numbers as children
  - Observe the relationship between parents and children
  - Use this relationship to directly recombine parents to form children

Properties of Binary Crossover

- To make crossovers independent of parents, define a spread factor, β

$$\beta = \left|\frac{c_1 - c_2}{p_1 - p_2}\right|$$

- Define probability distribution as a function of β
- Two observations:
  - Mean decoded value of parents is same as that of children
  - Child strings crossed at the same place produce the same parent strings

B_1				A_1			DV
1	0	1	0	1	0	1	85
0	1	1	0	0	1	1	51
B_2				A_2			68
B_1				A_2			DV
1	0	1	0	0	1	1	83
0	1	1	0	1	0	1	53
B_2				A_1			68

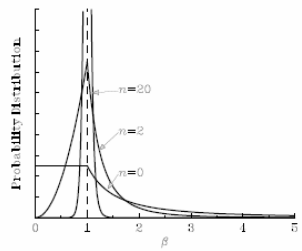
$\Longrightarrow$

Simulated Binary Crossover (SBX-β)

- Probability is large near the parents
- Extent is controlled by β
- Diversity in children proportional to that in parents
- Obeys *interval-schema* processing

$$\mathcal{C}(\beta) = 0.5(n+1)\beta^{\eta}$$

$$\mathcal{E}(\beta) = 0.5(n+1)\frac{1}{\beta^{n+2}}$$



SBX Procedure

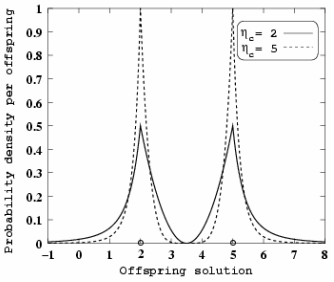
- Step 1: Choose a random number $u \in [0,1]$.
- Step 2: Calculate β_q:

$$\beta_q = \begin{cases} (2u)^{\frac{1}{\eta_c+1}}, & \text{if } u \le 0.5 \\ \left(\frac{1}{2(1-u)}\right)^{\frac{1}{\eta_c+1}}, & \text{otherwise} \end{cases}$$

- Step 3: Compute two offspring:

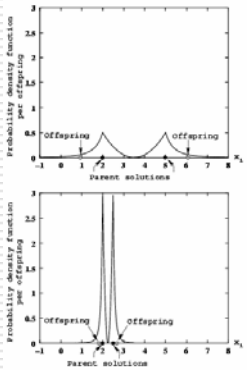
$$c_1 = 0.5\left(\left(1+\beta_q\right)p_1 + \left(1-\beta_q\right)p_2\right)$$

$$c_2 = 0.5\left(\left(1-\beta_q\right)p_1 + \left(1+\beta_q\right)p_2\right)$$



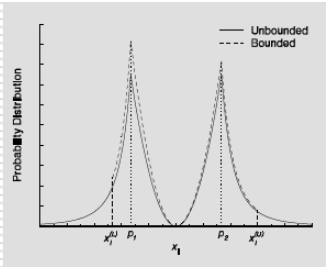
Properties of SBX Operator

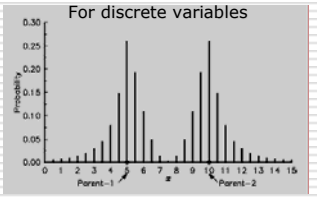
- If parents are distant, distant offspring are likely
- If parents are close, offspring are close to parents
- Self-adaptive property



Variations of SBX

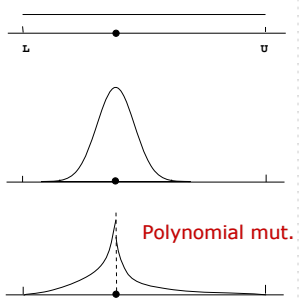
- For bounded and discrete variables





Real-Parameter Mutation Operators

- Idea: Create a neighboring solution
  - Random mutation
  - Normally distributed mutation
  - Non-uniform mutation
  - Extensions to bounded and discrete cases exist

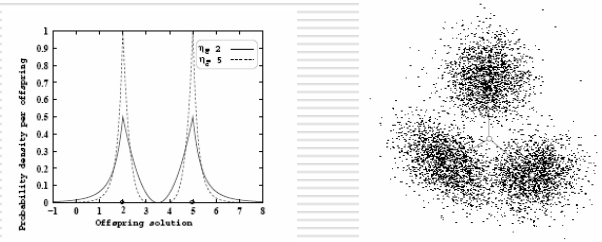


Vector-Wise Recombination Operators

- Variable-wise recombination cannot capture nonlinear interactions
- Recombine parents as vectors
  - Parent-centric recombination (PCX)
  - Unimodal normally-distributed crossover (UNDX)
  - Simplex crossover (SPX)
- Difference in parents is used to create offspring solutions
- DE, PSO, CMA-ES

Parent-Centric Recombination

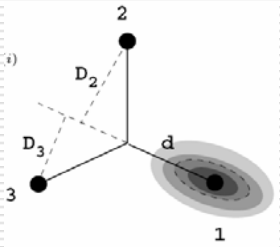
- SBX (Deb and Agrawal, 1995): variable-wise
- Vector-wise parent-centric recombination (PCX): O(μ) per offspring



PCX Operator

- μ parents create a offspring
- Each parent has its turn
- $e^{(i)}$ orthonormal bases spanning the subspace perpendicular to $d^{(p)}$
- w_ς and w_η are user-defined parameters, controlling extent of search

$$\vec{y} = \vec{x}_p + w_\varsigma |\vec{d}^{(p)}| + \sum_{i=1,\ i\neq p}^{\mu} w_\eta \bar{D} \vec{e}^{(i)}$$



Generalized Generation Gap (G3) Model

1. Select the best parent and μ-1 other parents randomly
2. Generate λ offspring using a recombination scheme
3. Choose two parents at random from the population
4. Form a combination of two parents and λ offspring, choose best two solutions and replace the chosen two parents

- Parametric studies with λ and N
- Desired accuracy in F is 10^{-20}

Quasi-Newton Method

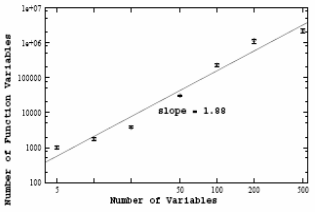
- Accuracy obtained by G3+PCX is 10^{-20}

Func.	FE	Best	Median	Worst
F_{elp}	6,000	$8.819(10^{-24})$	$9.718(10^{-24})$	$2.226(10^{-23})$
F_{sch}	15,000	$4.118(10^{-12})$	$1.021(10^{-10})$	$7.422(10^{-9})$
F_{ros}	15,000	$6.077(10^{-17})$	$4.046(10^{-10})$	3.987
F_{elp}	8,000	$5.994(10^{-24})$	$1.038(10^{-23})$	$2.226(10^{-23})$
F_{sch}	18,000	$4.118(10^{-12})$	$4.132(10^{-11})$	$7.422(10^{-9})$
F_{ros}	26,000	$6.077(10^{-17})$	$4.046(10^{-10})$	3.987

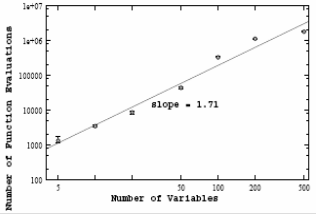
Scalability Study

- Accuracy 10^{-10} is set

Ellipsoidal

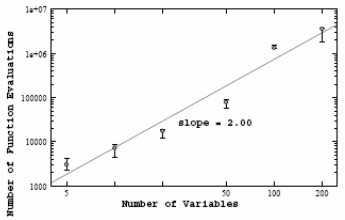


Schwefel



Scalability Study (cont.)

Rosenbrock



- All polynomial complexity $O(n^{1.7})$ to $O(n^2)$ similar to those reported by CMA-ES approach (Hansen and Ostermeier, 2001)

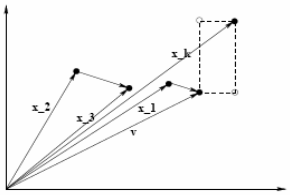
Differential Evolution (DE)

1. Start with a pool of random solutions
2. Create a child v
3. x_k and v are recombined with *p*

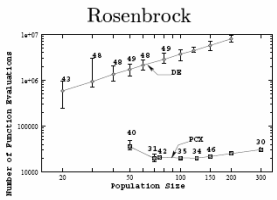
- Difference of parents in creating a child is important
- A number of modifications exist

$$\mathbf{v} = \mathbf{x}^{(1)} + \lambda(\mathbf{x}^{(2)} - \mathbf{x}^{(3)})$$

$$y_i = \begin{cases} v_i, & \text{with a prob. } p \\ \mathbf{x}_i^{(k)}, & \text{else} \end{cases}$$



DE Results



	F_{elp}		
	Best	Median	Worst
DE	9,660	12,033	20,881
G3	5,826	6,800	7,728
	F_{sch}		
DE	102,000	119,170	185,590
G3	13,988	15,602	17,188
	F_{ros}		
DE	243,800	587,920	942,040
G3	16,508	21,452	25,520

Particle Swarm Optimization (PSO)

- Kennedy and Eberhart, 1995
- Particles fly through the search space
- Velocity dynamically adjusted
- $x_i = x_i + v_i$
- $v_i = v_i + c_1 rnd()(p_{i,best} - x_i) + c_2 rnd()(p_g - x_i)$
- p_i: best position of i-th particle
- p_g: position of best particle so far
  - 1st term: momentum part (history)
  - 2nd term: cognitive part (private thinking)
  - 3rd term: social part (collaboration)

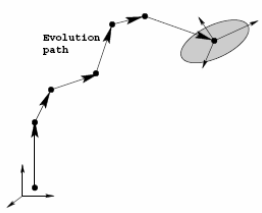
PSO Algorithm

1. Initialize a random population
2. For each particle
   - Evaluate and update p_{best}
3. Identify p_g
4. Compute velocity and position for each particle
5. Loop till termination

- Modifications exist
- Constrained PSO, Hybrid PSO, multi-objective PSO, etc.

CMA-ES (Hansen & Ostermeier, 1996)

- Selecto-mutation ES is run for n iterations
- Sucessful steps are recorded
- They are analyzed to find uncorrelated basis directions and strengths
- Required $O(n^3)$ computations to solve an eigenvalue problem
- Rotation invariant



CMA-ES On Three Test Problems

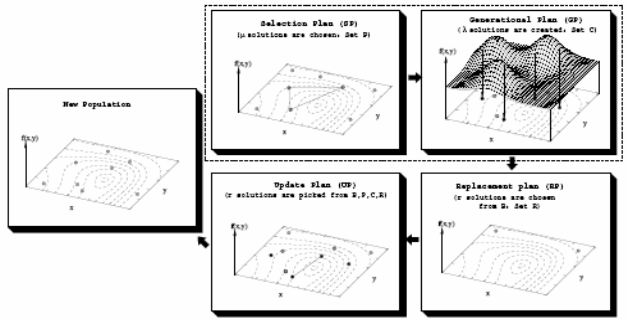
EA	F_{elp} Best	F_{elp} Median	F_{elp} Worst	F_{sch} Best	F_{sch} Median	F_{sch} Worst
CMA-ES	8,064	8,472	8,868	15,096	15,672	16,464
DE	9,660	12,033	20,881	102,000	119,170	185,590
G3+PCX	5,826	6,800	7,728	13,988	15,602	17,188
	F_{ros}					
CMA-ES	29,208	33,048	41,076			
DE	243,800	587,920	942,040			
G3+PCX	16,508	21,452	25,520			

Accuracy $1X10^{-20}$

Population-Based Optimization Algorithm-Generator

- Four functionally different Plans
  - Selection plan (SP): choose μ solutions from B to create P
  - Generation plan (GP): create λ solutions (C) using P
  - Replacement plan (RP): choose r solutions (R) from B
  - Update plan (UP): update B by replacing R (r solutions) from (P,C,R)

A Sketch of an Iteration



Designing an Optimization Procedure

1. Choose SP, GP, RP and UP
2. Perform a parametric study on a set of test problems
   - N, μ, λ, r

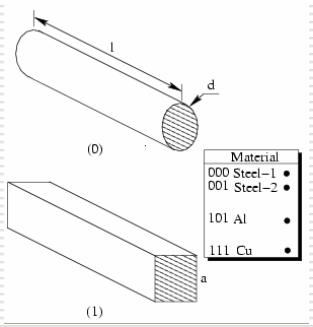
- Most classical and non-classical optimization algorithms explained
- Interestingly, PSO, DE, ES, EP can also be explained
- Possibility for a unified optimization approach

Mixed-Variable Optimization:
Handling mixed type of variables

- Treat type of cross-sections, materials, etc. as decision variables
- A mixed representation:

(1) 14 23.457 (101)

  - (1): circular or square cross-section
  - 14: diameter/side
  - 23.457: length
  - (101): material
- Permutation + real + cont.
- Complete optimization
- Deb and Goel, ASME-JMD, 1997



Constraint Handling:
Handling non-linear constraints

- Inequality constraints $(g_j(x) \geq 0)$ penalized for violation:

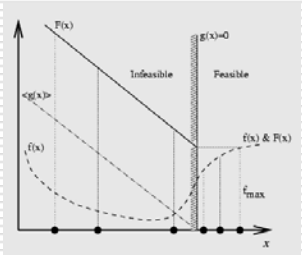
$$F(x) = f(x) + \sum_{j=1}^{J} R_j \langle g_j(x) \rangle^2$$

  - ‹a› = a if a is –ve, 0 otherwise
- Performance sensitive to penalty parameters

R	≤ 50%	Infeasible	Best	Median	Worst
10^0	12	13	2.41324	7.62465	483.50177
10^1	12	0	3.14206	4.33457	7.45453
10^3	1	0	3.38227	5.97060	10.65891
10^6	0	0	3.72929	5.87715	9.42353

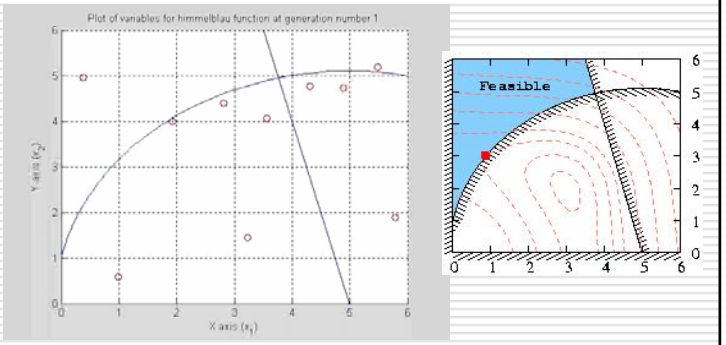
A Penalty-Parameter-less Population-based Approach

- Modify tournament sel.:
  - A feasible is better than an infeasible
  - For two feasibles, choose the one with better *f*
  - For two infeasibles, choose the one with smaller constraint violation $(\sum_j \langle g_j(x) \rangle)$
  - (Deb, 2000)



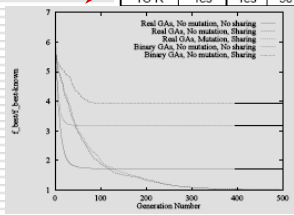
$$F(x) = \begin{cases} f(x), & \text{if } g_j(x) \geq 0, \forall j \in J \\ f_{\max} + \sum_{j=1}^{J} \langle g_j(x) \rangle, & \text{otherwise} \end{cases}$$

A Computer Simulation



Welded-Beam Design Problem

Method	Muta-tion	shar ing	ε ≤ 1%	≤ 2%	≤ 5%	≤ 10%	≤ 20%	≤ 50%	> 50%
Maximum generations = 500									
TS-B	No	No	0	0	0	0	0	0	50
TS-B	No	Yes	0	0	0	0	0	0	50
TS-R	No	No	0	0	1	4	8	16	34
TS-R	No	Yes	28	36	44	48	50	50	0
Maximum generations = 4,000									
TS-R	No	Yes	28	37	44	48	50	50	0
TS-R	Yes	Yes	50	50	50	50	50	50	0



Case	Best	Median	Worst
1	4.939027	7.183082	12.084255
2	3.820984	8.899963	14.298933
3	2.442714	3.834121	7.444246
4	2.381191	2.392892	2.645833
5	2.381187	2.392025	2.645833
6	2.381447	2.382634	2.383554

Large-Scale Optimization:
Handling large number of variables

- Large n, large pop-size, large computation
- Knowledge-augmented EAs
  - Representation
  - Operators
- EA's flexibility shows promise
- A case study involving millions of variables (Deb and Reddy, 2001)

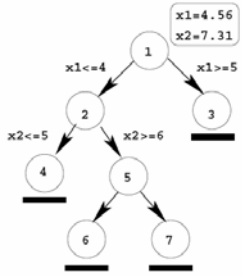
Casting Scheduling

Casting Scheduling Problem (cont.)

- Maximum metal utilization:

$$\frac{1}{H}\sum_{i=1}^{H}\frac{100\sum_{k=1}^{K} w_k x_{ki}}{W_i}$$

- Constraints:
  - Demand satisfaction: $\sum_{i=1}^{H} x_{ki} = r_k$ for $k = 1, \ldots, K$
  - Capacity constraint: $\sum_{k=1}^{K} w_k x_{ki} \le W_i$ for $i = 1, \ldots, H$
- An integer linear program (ILP)
- Branch-and-bound – exponential algorithm



Performance of LINGO

- Works up to n=500 on a Pentium IV (7

LINGO MILP Solver												
Heat No.	Order Number										Utilization/ Cruc. Size	Efficiency (%)
	1	2	3	4	5	6	7	8	9	10		
1	0	1	1	0	0	0	2	1	0	0	623/650	95.85
2	2	0	0	0	1	0	0	0	2	0	615/650	94.62
3	1	0	0	1	3	1	0	0	0	0	611/650	94.00
4	2	0	0	0	1	0	0	1	0	0	645/650	99.23
5	0	0	0	1	0	2	0	0	1	6	612/650	94.15
6	1	1	0	0	2	1	0	0	0	0	591/650	90.92
7	0	0	2	2	1	0	0	0	2	0	585/650	90.00
8	0	3	0	0	0	1	0	0	1	0	611/650	94.00
9	0	2	3	0	1	0	0	0	0	0	650/650	100.00
10	1	0	0	5	0	0	0	0	1	0	635/650	97.69
	7	7	6	9	9	5	2	2	7	6	Average	95.05

Off-The-Shelf GA Results

Number of Variables	Binary-coded GAs			Real-coded GAs		
	Population Size	Efficiency	Function Eval.	Population Size	Efficiency	Function Eval.
100	100	96.15	13,600	100	95.94	23,740
200	300	95.01	1,42,200	200	92.81	1,21,760
300	1,000	90.11	14,12,400	700	95.14	5,84,220

- Exponential function evaluations
- Random initialization, standard crossover and mutations are not enough
- Need a customized EA

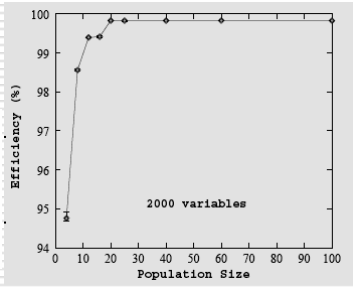
No Free Lunch (NFL) Theorem

- No one gives other a free lunch
- In the context of *optimization*
  - Wolpert and McCardy (1997)
  - Algorithms A1 and A2
  - All possible problems **F**
  - Performances P1 and P2 using A1 and A2 for a fixed number of evaluations
  - P1 = P2
- NFL breaks down for a narrow class of problems or algorithms
- Research effort: Find the best algorithm for a class of problems
  - Unimodal, multi-modal, quadratic etc.

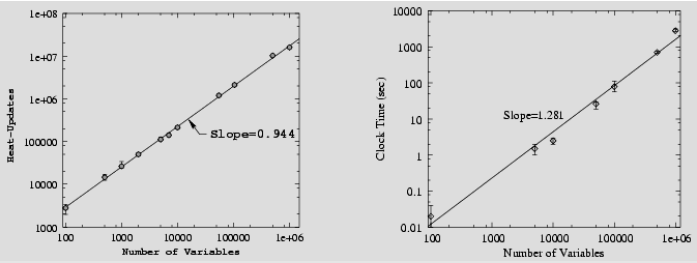
A Customized GA: Optimal Population Size

- N=2000 variables with max. gen.=1000/N
- A critical population size is needed

N:	4	8	12	16
# Heats:	211	204	201	201
N:	20	25	40	100
# Heats:	200	200	200	200



Scale-Up Results

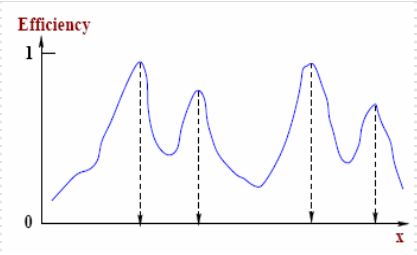


- Knowledge-augmented GA has sub-quadratic complexity and up to one million variables (Deb and Pal, 2003)
- Never before such a large problem was solved using EAs

Multi-Modal Optimization:

Handling multiple optimal solutions

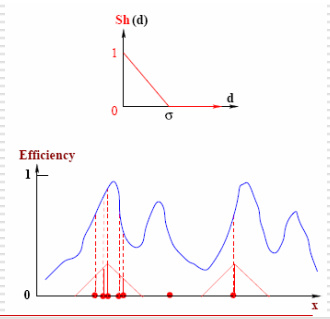
- To solve problems with multiple local/global optimum
- Classical methods can find only one optimum at a time
- EAs can, in principle, find multiple optima simultaneously



Niching Concept

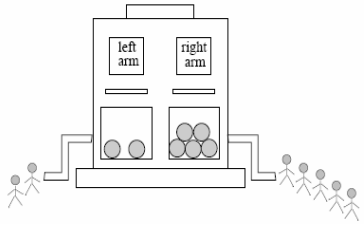
Multiple niches (human and animal) coexist in nature

- By sharing resources (land, food, etc.)
- How to mimmick the concept in EAs?
- Reduce selection pressure for crowded solutions
- Use a sharing function based on two-armed bandit problem



Two-Armed Bandit Problem

- If we know which arm pays more, all population members crowd there
  - Same as converging to one optima
- Introduce sharing payoff strategy
- All player share payoff with others interested in playing the same arm
- Steady-state distribution: Players in both arms

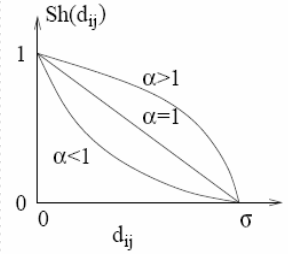


$f_L/m_L = f_R/m_R$

Sharing Function

- Goldberg and Richardson (1997)
- d is a distance measure between two solns.
  - Phenotypic distance: $d(x_i,x_j)$, x: variable
  - Genotypic distance: $d(s_i,s_j)$, s: string
- Calculate niche count, $nc_i=\Sigma_j Sh(d_{ij})$
- Shared fitness: $f_i'=f_i/nc_i$
- Use proportionate selection opeartor

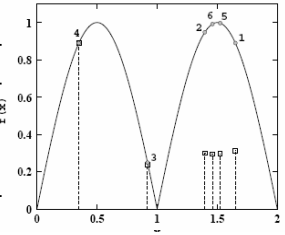
$$Sh(d_{ij}) = \begin{cases} 1-\left(\frac{d_{ij}}{\sigma}\right)^{\alpha} & \text{if } d_{ij} \le \alpha, \\ 0 & \text{otherwise.} \end{cases}$$



An Example: Phenotypic Sharing

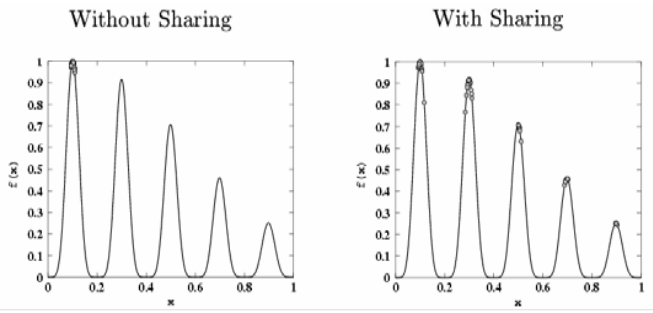
- Maximize $f(x) = |\sin(\pi x)|$, $x \in [0, 2]$
- $nc_i = \sum_{j=1}^{N} Sh(d_{ij})$

Sol. i	String	$x^{(i)}$	f_i	nc_i	f_i'
1	110100	1.651	0.890	2.856	0.312
2	101100	1.397	0.948	3.160	0.300
3	011101	0.921	0.246	1.048	0.235
4	001011	0.349	0.890	1.000	0.890
5	110000	1.524	0.997	3.364	0.296
6	101110	1.460	0.992	3.364	0.295

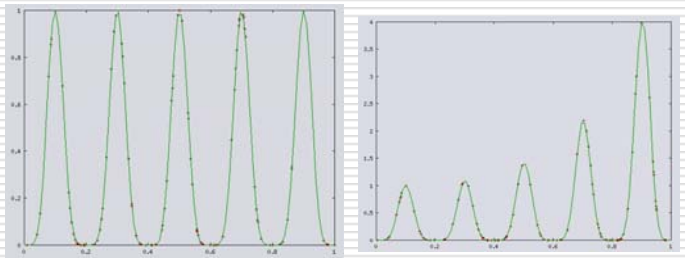


Simulation Results

- Inclusion of niche-formation strategy



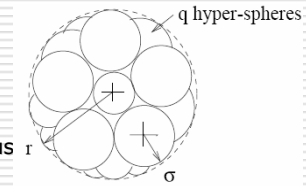
Simulation on Two Problems



Phenotypic σ_{share}

- Assume number of optima: q
- 1-D space:

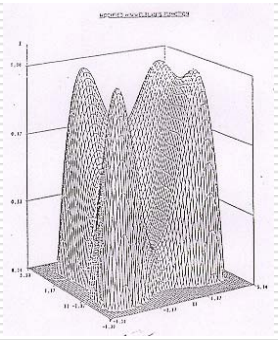
  $\sigma_{share} = (x_{max}-x_{min})/2q$
- p-D space:
  - r: Euclidean length of radius of entire p-D hyper-sphere
  - Volume: $V = cr^p$
  - Divide volume into q parts
  - σ_{share} is the radius of each part
  - $V = cr^p = q\, c\sigma_{share}^p$ yields



$$\sigma_{share} = r/q^{1/p}$$

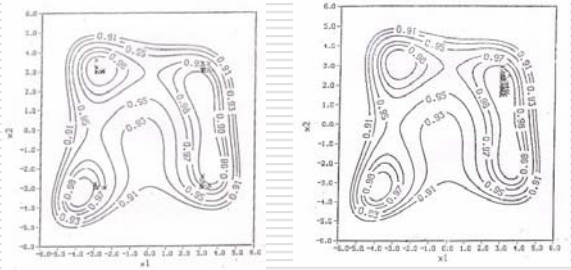
More Simulation Results

- A two-dimensional function
- Four optima
- Interested in finding all four optima



More Simulation Results (cont.)

- With and without sharing (phenotypic)



Genotopic Sharing

- d=Hamming distance (# of bit differences)
- q niches:

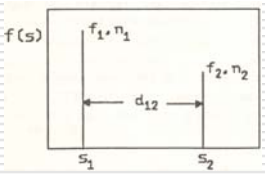
$$\frac{1}{2^l}\prod_{i=0}^{\sigma}\binom{l}{i}=\frac{1}{q}$$

- Approximate:

$$\sigma=\frac{1}{2}\left(l+z^*\sqrt{l}\right)$$

- z* found from cum. Normal distribution chart for 1/q
- Restriction on gen. sharing:
  - Need a minimum distance between optima ($\gamma=f_1/f_2$)

$$d \geq \left(1-\frac{1}{\gamma}\right)\sigma$$



Other Niching Approaches

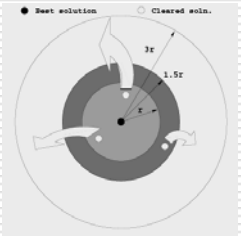
- Clearing approach
- Clustering approach
- Crowding approach
- Pre-selection approach
- Restrictive tournament selection approach
- All require at least one tunable parameter

Clearing Strategy

- Clear bad near good solutions
  - Sort population according to fitness
  - Keep the best and assign zero fitness to all others within *d* from best
  - Keep the next best and assign zero fitness to others within *d* from next best
  - Continue
  - Perform a selection with assigned fitness
- Petrowski (1996)

Modified Clearing Strategy

- Most members are wasted
- A better approach
  - All zero-fitness solutions are re-initialized outside 1.5*r* of best
  - Performs better
- Singh and Deb (2006)

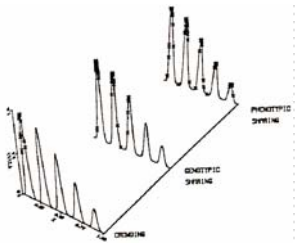


Clustering Approach

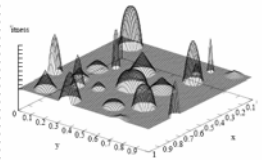
- Based on phenotypic space, a clustering is performed
  - With a predefined *d*
  - With a predefined number of clusters
  - Niching within each cluster
  - Mating restriction can also be performed
- Parameter sensitive
- Yin and Germay (1993)

Crowding Approach

- De Jong (1975)
  - Child solution replaces the most similar solution from a subpopulation
  - Subpopulation size
- Mahfoud (1993)
- Maintains diversity
- Not as good as sharing (Deb and Goldberg, 1989)



Comparison on a Scalable Problem



- Optima are placed randomly
- Gaussian shape
- A8 (Mod. Clearing) performs the best
  - A1: Clearing
  - A2: Clustering
  - A4: Crowding

No. of Peaks		A1	A2	A4	A8
20	SR	15.78	19.0	19.28	**20.0**
	SD	4.18	1.44	1.20	**0.0**
	TT	112.6	23.39	120.4	2156.7
	SD	54.69	4.52	69.15	1635.4
30	SR	25. 0	27.38	28.58	**30.0**
	SD	3.02	2.82	1.56	**0.0**
	TT	155.0	36.72	144.8	2526.8
	SD	219.6	3.83	245.6	1854.5
40	SR	35.14	36.24	38.12	**40.0**
	SD	4.50	2.80	2.23	**0.0**
	TT	121.6	120.7	125.0	2967.9
	SD	15.14	68.94	9.23	1797.6
50	SR	44.86	42.2	47.6	**50.0**
	SD	2.53	4.40	2.43	**0.0**
	TT	321.8	222.8	335.6	3326.0
	SD	64.14	60.92	138.9	2142.7

Optimization with Meta-Models:

Handling computationally expensive problems

- Evaluation of most real-world problems is computationally expensive
- Optimization algorithm run into days
- To save time, use approximate models of objective function and constraints
- Different techniques
  - A fixed model
  - Updating the model with iteration

Metamodeling Techniques

- Three tasks
  - Generate data using an experimental design method or a computer
  - Choose a model to represent data
  - Fit the model to the data
- Some common techniques
  - Response surface methodology (RSM)
  - Neural networks
  - Kriging
  - Inductive learning

Metamodeling Techniques (cont.)

Generate Data	Model Selection	Model Fitting	Tech.
Factorial	Polynomial	Least squares	RSM
D-Optimal	Splines	Weighted least squares	
G-Optimal	Stochastic process	Best linear un-biased predictor	Kriging
Orthogonal array	Kernel smoothing	Log-likelihood	
By hand	Neural networks	Backpropagation	Neural network
Random selection	Rulebase/ Decision tree	Entropy	Inductive learning

Response Surface Method (RSM)

- Box and Wilson (1951)
- Model: Error is independent of x
  - $y(\mathbf{x}) = g(\mathbf{x}) + \epsilon,\ \epsilon = N(0, \sigma),$
  - $\hat{y}(\mathbf{x}) = g(\mathbf{x})$
- Assume g(x), usually parametric linear or quadratic

$$\hat{y} = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \sum_{j=1}^{k} \beta_{ij} x_i x_j$$

- β_i detern rom observed data
  - Optimize to minimize error: Find mean β_i
  - Variance of β_i determine predictive capability

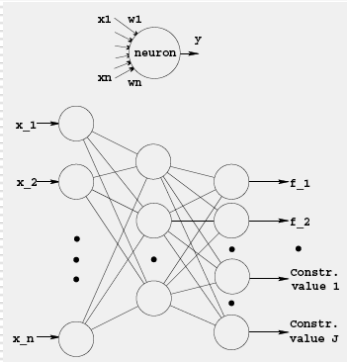
RSM Tips

- Popular and most widely used
- Best suited in applications with random error
- However, limited handling on non-linearity
- Usually applied for k<10
- Sequencial RSM with move limits and trust region approach is better

Artificial Neural Networks (ANN)

- Mimicking natural neural processing
- Input: Variables (n)
- Output: Objective and constraint functions
- Sigmoidal activation function for hidden and output neurons

$$y = \frac{1}{1 + \exp\left(-(\sum_i w_i x_i + \beta)/T\right)}$$



ANN (cont.)

- Obtain relationship among input-output
- Optimal connection and its weights for minimum error

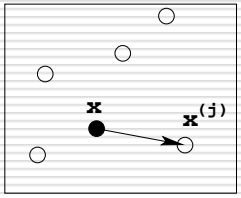
$$E = \|\mathbf{y}_p - \hat{\mathbf{y}}_p\|$$

- A non-linear regression method
- Suitable for large (104) parameters
- Best suited for deterministic problems (no random error)
- Computationally expensive procedure

Kriging Procedure

- D. G. Krige (a geologist): Statistical analysis of mining data
- Predict a value at a point from a given set of observations

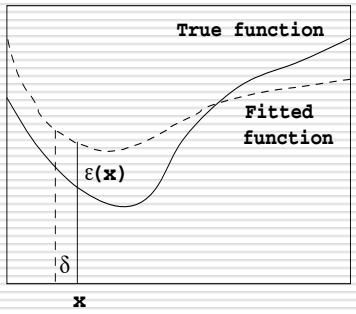
$$f(\mathbf{x}) = \sum_{i=1}^{M} \lambda_i y(\mathbf{x}^{(j)})$$



- λ_i depends on distance of X from observed points
- Flexible, but complex
- Suited for k<50, deterministic problems

Fundamentals of Kriging

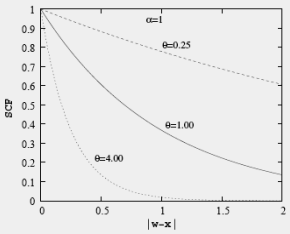
- F(x) is true function
- f(x) is fitted function
- ε(x) = F(x)-f(x)
- If error at x is large, it is reasonable to believe that
  - Error at x+δ is also large



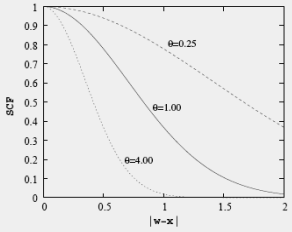
Fundamentals of Kriging (cont.)

- Say, M data points: $(\mathbf{x}^{(j)}, y_j), \quad j = 1, 2, \ldots, M$
- Assume model: $Y(\mathbf{x}) = \sum_{k=1}^{K} \beta_k \mathbf{f}_k(\mathbf{x}) + Z(\mathbf{x})$
- $\mathbf{f}_k(\mathbf{x})$ assumed global model, can be taken as constant, polynomial or others
- $Z(\mathbf{x})$ is a local variation and stochastic:
  - $E(Z(\mathbf{x})) = 0$
  - $Cov(Z(\mathbf{w}), Z(\mathbf{x})) = \sigma_Z^2 R(\mathbf{w}, \mathbf{x})$
  - σ_Z^2 is the process variance
  - $R(\mathbf{w}, \mathbf{x})$ is spatial correlation function (SCF)
  - $R(\mathbf{w}, \mathbf{x}) = \Pi_{j=1}^{n} R_j(\mathbf{w}_j, \mathbf{x}_j)$
  - SCF (R_j) is function of distance between $\mathbf{w}_i$ and $\mathbf{x}_i$

Commonly Used SCFs



Exponential:
$\exp\left(-\theta_j \mid \mathbf{w}_j - \mathbf{x}_j \mid^{\alpha}\right) \quad 0 < \alpha \leq 2$



Gaussian:
$\exp\left(-\theta_j \mid \mathbf{w}_j - \mathbf{x}_j \mid^{2}\right)$

Kriging Procedure

Step 1: Normalize $\mathbf{x}^{(j)}$ and $y^{(j)}$ and set $\mathbf{Y} = [y^{(j)}]$

Step 2: Maximize log-likelihood function:

$$\max_{\boldsymbol{\theta}} \quad -\frac{1}{2}\left[M \log \sigma_Z^2 + \log R(\boldsymbol{\theta})\right]$$

and obtain $\boldsymbol{\theta}^*$

- $R(\boldsymbol{\theta})$ is a matrix: $\Pi_{j=1}^{n} R_j(\mathbf{w}_j, \mathbf{x}_j;\ \theta_j)$
- MLE of σ_Z^2: $\sigma_Z^2 = \frac{1}{M}\left(\mathbf{Y} - \mathbf{I}\hat{\beta}(\boldsymbol{\theta})\right)^T R(\boldsymbol{\theta})^{-1}\left(\mathbf{Y} - \mathbf{I}\hat{\beta}(\boldsymbol{\theta})\right)$
- MLE of $\hat{\beta}$: $\hat{\beta}(\boldsymbol{\theta}) = (\mathbf{I}^T R(\boldsymbol{\theta})^{-1}\mathbf{I})^{-1}\mathbf{I}^T R(\boldsymbol{\theta})^{-1}\mathbf{Y}$

Step 3: Compute $\hat{\beta}^* = \hat{\beta}(\boldsymbol{\theta}^*)$, $R(\boldsymbol{\theta}^*)$, and ${\sigma_Z^2}^*$

Kriging Procedure (cont.)

Step 4: Compute fitted function at any point $\mathbf{x}$:

$$\hat{y}(\mathbf{x}) = \hat{\beta}^* + r^T(\mathbf{x})R(\boldsymbol{\theta}^*)^{-1}\left(\mathbf{Y} - \mathbf{I}\hat{\beta}^*\right)$$

where $r(\mathbf{x}) = [R(\mathbf{x}, \mathbf{x}^{(i)})] \quad i = 1, 2, \ldots, M$

Step 5: The corresponding MSE (error):

$$MSE(\mathbf{x}) = {\sigma_Z^2}^* \left[1 - \begin{pmatrix} 1 & r^T(\mathbf{x}) \end{pmatrix} \begin{pmatrix} 0 & \mathbf{I}^T \\ \mathbf{I} & R(\boldsymbol{\theta}^*) \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ r(\mathbf{x}) \end{pmatrix}\right]$$

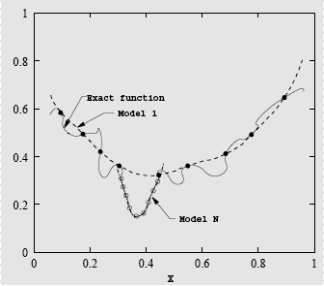
- If $\mathbf{x}$ is closer to an observed point, the MLE will be small

Model Verification

- Leave-one-out-cross-validation scheme
- Leave out one observation (say $x^{(J)}$ and $y^{(J)}$)
- Find the kriging model with (M-1) points and find $\hat{y}^{(J)}$
- Compute the error $s^J = \sqrt{(MSE(x^{(J)}))}$
- If the normalized error $(y^{(J)} - \hat{y}^{(J)})/s^J$ is within [-3,3], the model is acceptable

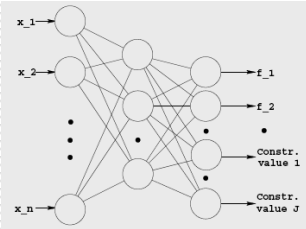
Successive Modeling Procedure

- Nain and Deb (2003)
- Successive approximations to the problem
- Initial coarse approximate model defined over the whole range of decision variables with small database
- Gradual finer approximate models localized in the search space

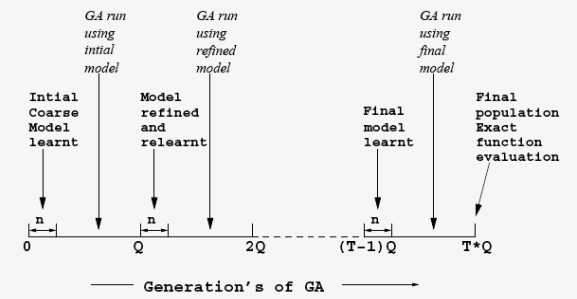


ANN Model

- A feed-forward neural network
- Input: Decision variables (size n)
- Output: Objective functions and constraint violations (size M+J)
- One hidden layer with H neurons
- Sigmoidal activation for hidden and output neurons



Generation-wise Sketch of Proposed Approach



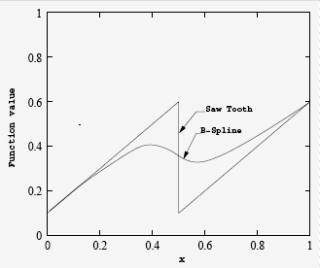
n/Q fraction of exact evaluations, although the ratio can be made smaller later

Flip Side: Parameters Involved

- Frequency of ANN modeling
- Number of training points (can reduce with iteration)
- Learning rate in training
- Final RMS error in ANN training
- Learning models (incremental or batch)
- ANN architecture (connectivity and hidden layers

A Case Study from Curve-Fitting

- Two-objective problem:
  - Minimize error from the curve
  - Minimize the maximize curvature
- 41 control points forming a B-spline curve (39 varied)



Formulation of the Problem

- A B-spline curve S having 41 control points in [0,1]
- Numerical computation of the integral with 400 intermediate points:

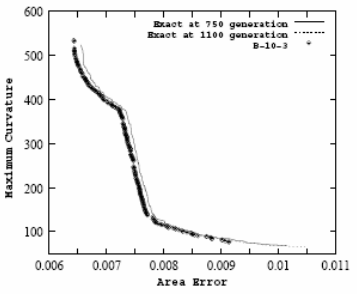
$$f_1(\mathcal{S}) = \int_{x=0}^{x=1} | f_{\text{saw-tooth}} - \mathcal{S} | \, dx$$

- Differentials are computed exactly

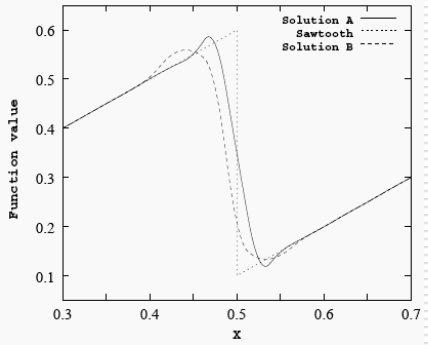
$$f_2(\mathcal{S}) = \max_{x=0}^{x=1} \frac{\frac{d^2\mathcal{S}}{dx^2}}{\left[1 + \left(\frac{d\mathcal{S}}{dx}\right)^2\right]^{3/2}}$$

Savings in Function Evaluations

- B-10-3 finds a front in (750x200) evaluations similar to NSGA-II in (1100x200) evaluations
- A saving of 32% evaluations

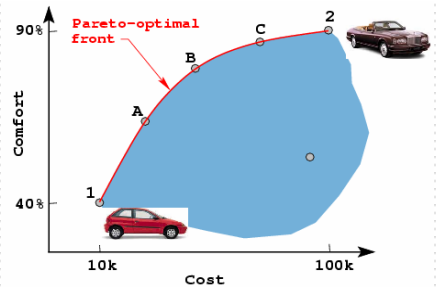


Extreme Trade-off Solutions



Multi-Objective Optimization:
Handling multiple conflicting objectives

- We often face them

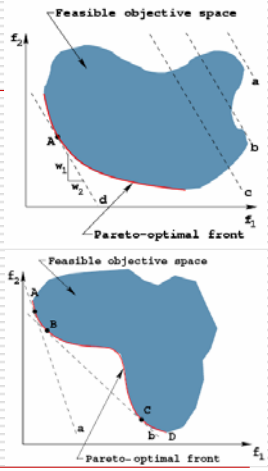


Classical Approach:
Weighted-Sum Method

- Construct a weighted sum of objectives and optimize

$$F(x) = \sum_{i=1}^{M} w_i f_i(x)$$

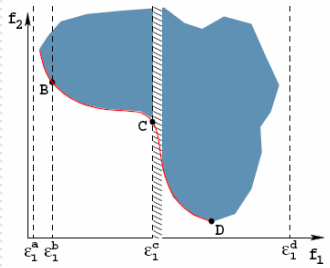
- User supplies weight vector *w*
- Non-convexity a problem



Classical Approach:
ε-Constraint Method

- Constrain all but one objective
- Need to know relevant **ε** vectors
- Non-uniformity in Pareto-optimal solutions
- Any Pareto-optimal solutions can be found with this approach

$$\text{Minimize} \quad f_\mu(\mathbf{x}),$$
$$\text{subject to} \quad f_m(\mathbf{x}) \le \epsilon_m, \; m \ne \mu;$$



Evolutionary Multi-Objective Optimization (EMO)

Step 1 :

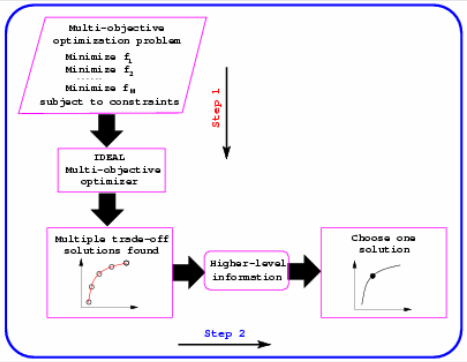
Find a set of Pareto-optimal solutions

Step 2 :

Choose one from the set

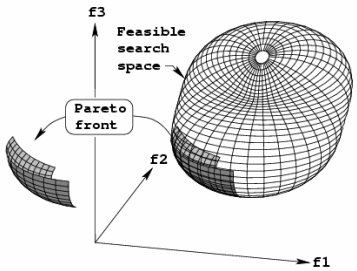
(Deb, 2001)

- Ideal for an EA



Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II)

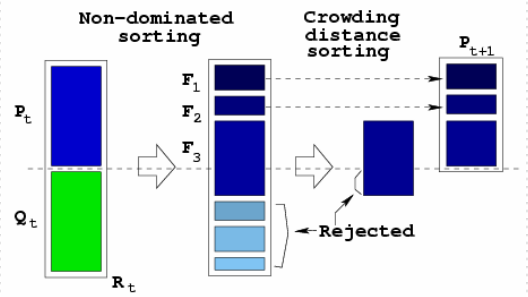
- NSGA-II can extract Pareto-optimal frontier
- Also find a well-distributed set of solutions
- iSIGHT and modeFrontier adopted NSGA-II



Fast-Breaking Paper in Engineering by ISI Web of Science (Feb'04)

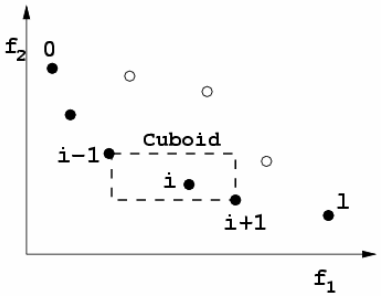
NSGA-II Procedure

Elites are preserved
Non-dominated solutions are emphasized



NSGA-II (cont.)

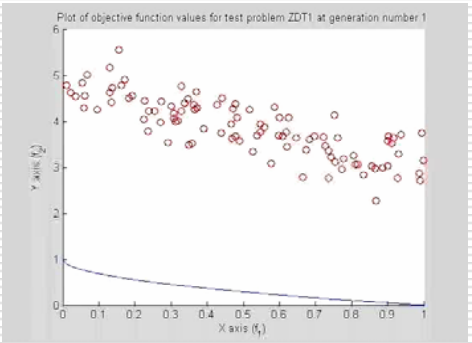
Diversity is maintained



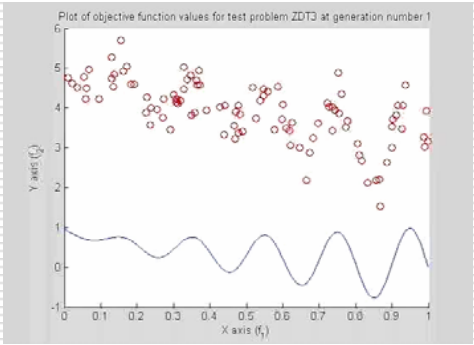
Overall Complexity
$O(N \log^{M-1} N)$

Improve diversity by
- k-mean clustering
- Euclidean distance measure
- Other techniques

Simulation on ZDT1

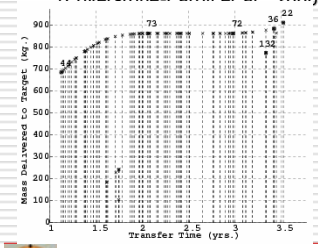


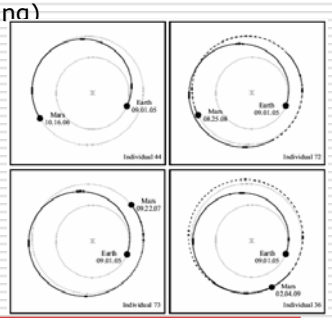
Simulation on ZDT3



EMO Applications

- Identify different trade-off solutions for choosing one (Better decision-making)
- Inter-planetary trajectory (Coverstone-Carroll et al., 2000)

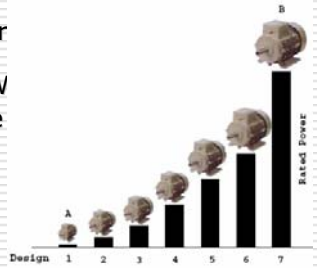




Innovization:

Discovery of Innovative design principles through optimization

- Understand important design principles in a routine design scenario
- Example: Electric motor design with varying ratings, say 1 to 10 kW
  - Each will vary in size and power
  - Armature size, number of turns etc.
- How do solutions vary?
  - Any common principles!



Single versus Multiple Objectives

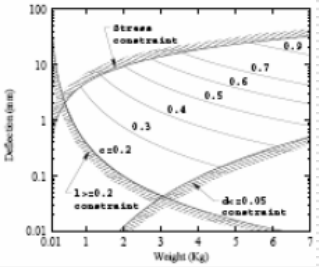
- Say, a cantilever beam design for minimum weight

$$\begin{aligned} \text{Minimize} \quad & f_1(d,l) = \rho\frac{\pi d^2}{4}l \\ \text{subject to} \quad & \frac{32Pl}{\pi d^3} \leq S_y \\ & 0.01 \leq d \leq 0.05 \\ & 0.2 \leq l \leq 1.0 \end{aligned}$$

- Optimal design:
  - d=18.94 mm, l=200 mm, defl. = 2 mm
- Want defl.=1 mm, what design?
- Redo optimization with a constraint
- Turns out: d=22.52 mm, l=200 mm

Knowledge Discovery

- Minimize (weight,defl.)
- Try if *l*=*c* is true
- Innovization:
  - Set l = c = 0.2m to be optimal
  - Range of d: (18.94,50) mm
- Knowledge discovery!
- How do systemize the procedure?



Innovization Procedure

- Choose two or more conflicting objectives (e.g., size and power)
  - Usually, a small sized solution is less powered
- Obtain Pareto-optimal solutions using an EMO
- Investigate for any common properties manually or automatically
- Why would there be common properties?
  - Recall, Pareto-optimal solutions are all optimal!

In Search of Common Optimality Properties

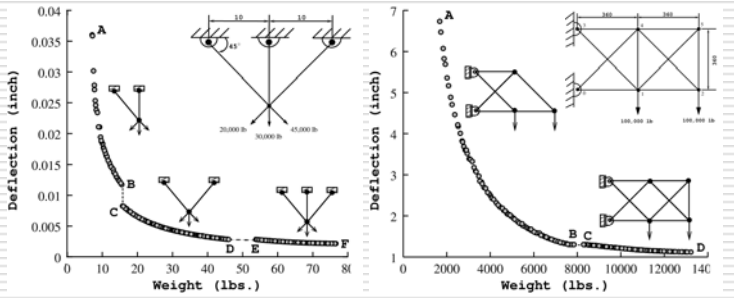
Fritz-John Necessary Condition:

Solution x^* satisfy

1. $\sum_{m=1}^{M} \lambda_m \nabla f_m(x^*) - \sum_{j=1}^{J} u_j \nabla g_j(x^*) = 0,$ and
2. $u_j g_j(x^*) = 0$ for all j = 1, 2, 3, ……,J
3. $u_j \geq 0$, $\lambda_j \geq 0$, for all j and $\lambda_j > 0$ for at least one j

- To use above conditions requires differentiable objectives and constraints
- Yet, it lurks existence of some properties among Pareto-optimal solutions

Revealing Salient Insights: Truss Structure Design

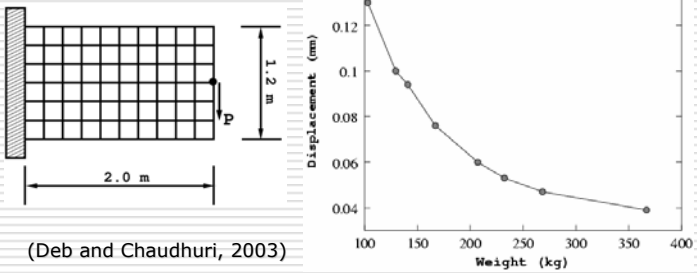


(Deb, Khan and Jindal, 2000)

Revealing Salient Insights: A Cantilever Plate Design

Eight trade-off solutions are chosen

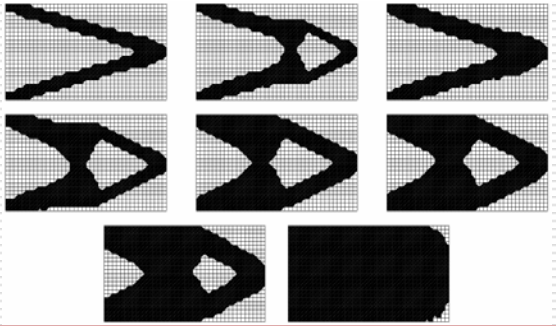
Base Plate



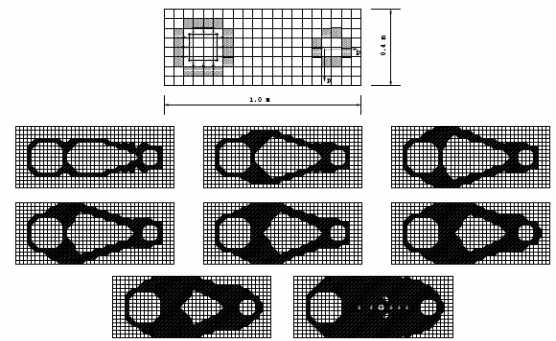
(Deb and Chaudhuri, 2003)

Trade-Off Solutions

- Symmetry in solutions about mid-plane, discovery of stiffener



A Connecting Rod



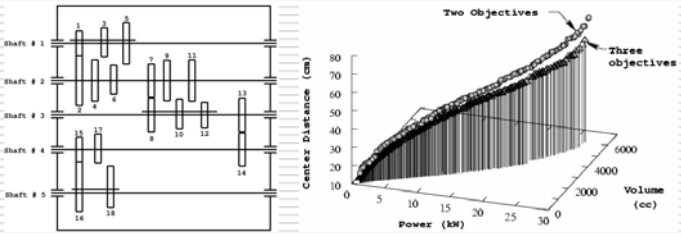
Innovized Principles

- Mid-line symmetry
- Straight arms to reach load is minimum-weight strategy
- Two ways to increase stiffness
  - Thickening of arms
  - Use of a stiffener
  - Additional stiffening by a combination
- Chamfering of corners helpful

Gear-box Design

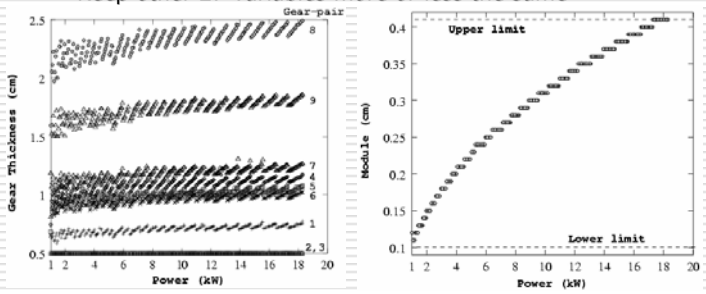
- A multi-spindle gear-box design (Deb and Jain, 2003)
- 28 variables (integer, discrete, real-valued)
- 101 non-linear constraints
- Important insights obtained

(larger module for more power)



Innovized Principles

- Module varies proportional to square-root of power
- Keep other 27 variables more or less the same

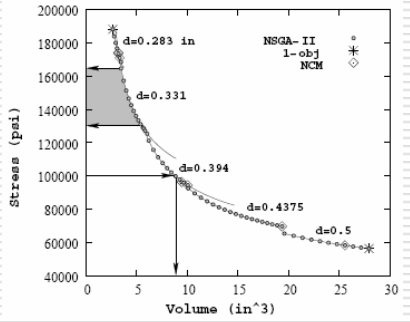


Mechanical Spring Design

- Minimize material volume
- Minimize developed stress
- Three variables: (d, D, N):
  discrete, real, integer
- Eight non-linear constraints
  - Solid length restriction
  - Maximum allowable deflection (P/k≤6in)
  - Dynamic deflection (P_m-P)/k≥1.25in
  - Volume and stress limitations

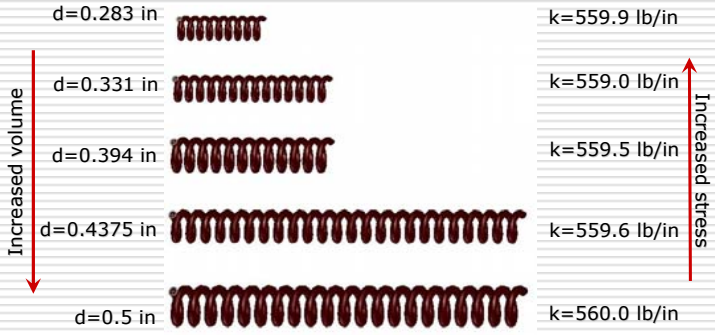
Innovized Principles

- Pareto-optimal front have niches with d
- Only 5 (out of 42) values of d (large ones) are optimal
- A blue-print for optimal design



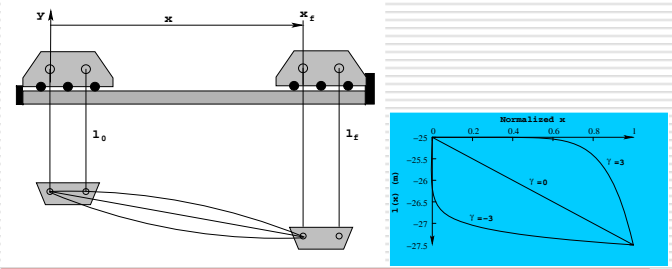
(Deb and Srinivasan, 2005)

Optimal Springs, Optimal Recipe



Overhead Crane Maneuvering

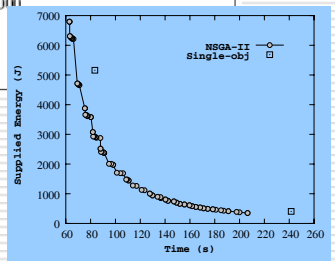
- Minimize time of operation
- Minimize operating energy



Simulation Results

γ	Force (N)	Pattern
2.98	149.26	11111111001000100000000000000000000000000000000000000000
3.00	187.76	1111111100100010000000000000000000000000000000
3.00	238.56	111111110010001000000000000000000
3.00	327.86	11111011001001100000000000
2.98	427.93	1111101100100110000000
3.00	544.94	111111110010000100
2.87	629.62	11111111001001
2.88	818.99	1101110010010001

- NSGA-II finds trade-off and interesting properties



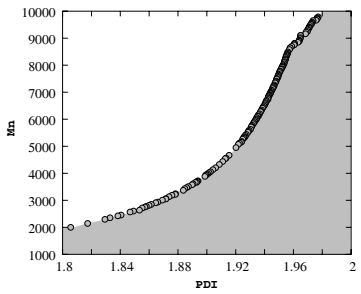
Innovative Principles

- For optimum operation:
  - Lower load suddenly at the end
  - Spend energy only at the beginning
- Fast unloading demands more energy
  - For fixed θ, E ∞ l
  - Delay lowering for saving energy

$$E = mg(1 - \cos\theta)l$$

Epoxy Polymerization

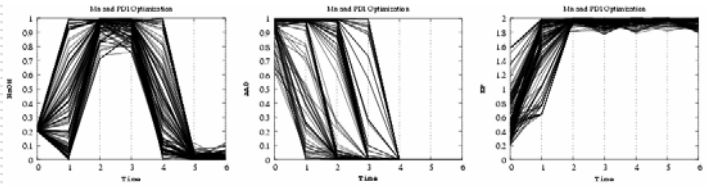
- Three ingredients added hourly
- 54 ODEs solved for a 7-hour simulation
- Maximize chain length (Mn)
- Minimize polydispersity index (PDI)
- Total 3x7 or 21 variables
- (Deb et al., 2004)



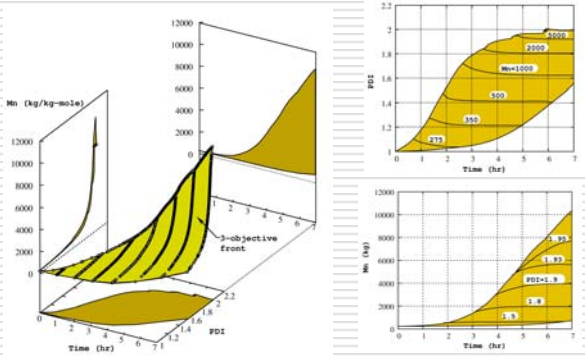
A non-convex frontier

Epoxy Polymerization (cont.)

- Some patterns emerge among obtained solutions
- Chemical significance unveiled

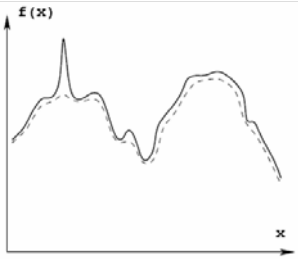


Innovized Principles: *An Optimal Operating Chart*



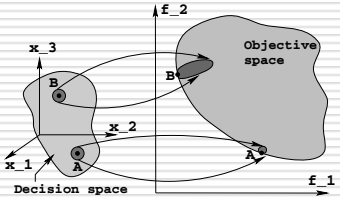
Robust Optimization: *Handling uncertainties in variables*

- Parameters are uncertain and sensitive to implementation
  - Tolerances in manufacturing
  - Material properties are uncertain
  - Loading is uncertain
- Who wants a sensitive optimum solution?
- Single-objective robust EAs (Branke and others)



Multi-Objective Robust Solutions

- Solutions are averaged in δ-neighborhood
- Not all Pareto-optimal points may be robust
- A is robust, but B is not
- Decision-makers will be interested in knowing robust part of the front



Multi-Objective Robust Solutions of Type I and II

- Similar to single-objective robust solution of type I

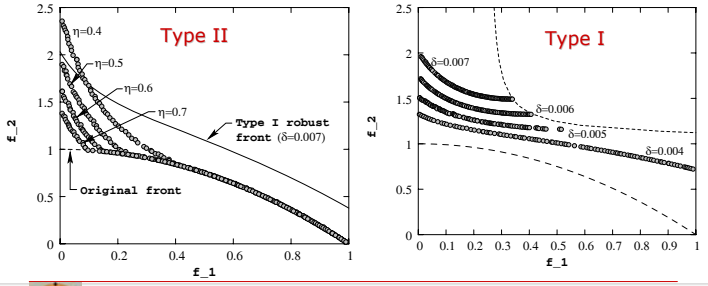
$$\begin{aligned} &\textit{Minimize} && (f_1^{\text{eff}}(\mathbf{x}), f_2^{\text{eff}}(\mathbf{x}), \ldots, f_M^{\text{eff}}(\mathbf{x})), \\ &\textit{subject to} && \mathbf{x} \in \mathcal{S}, \end{aligned}$$

- Type II

$$\begin{aligned} &\textit{Minimize} && \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x})), \\ &\textit{subject to} && \frac{\|\mathbf{f}^p(\mathbf{x}) - \mathbf{f}(\mathbf{x})\|}{\|\mathbf{f}(\mathbf{x})\|} \leq \eta, \\ & && \mathbf{x} \in \mathcal{S}. \end{aligned}$$

Robust Frontier for Two Objectives

- Identify robust region
- Allows a control on desired robustness



Robustness with Constraints

- Constraints will make boundary solutions infeasible
- Simply declare a solution infeasible, if any of H neighboring points is infeasible
- Use constraint tournament selection in NSGA-II

Test Problem 1

Constrained effective front is different

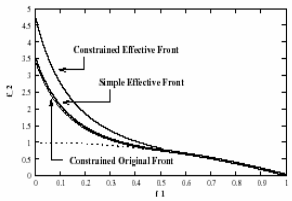


Fig. 1. Theoretical simple and constrained effective fronts on test problem 1.

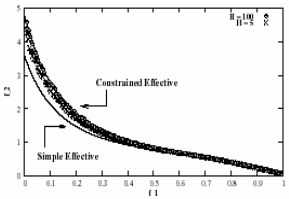


Fig. 2. Constrained effective fronts showing the effect of number of points H on test problem 1.

Effect of H

Larger H, more shift

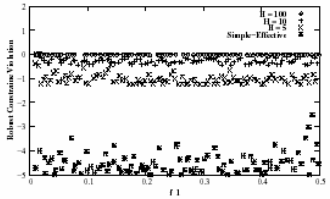


Fig. 3. Robust constraint violation for the points obtained on test problem 1 with different values of H.

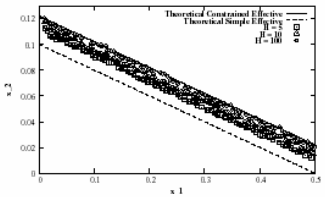


Fig. 4. x_1 vs x_2 relationship for different values of H on test problem 1.

Faster Computation for Robustness

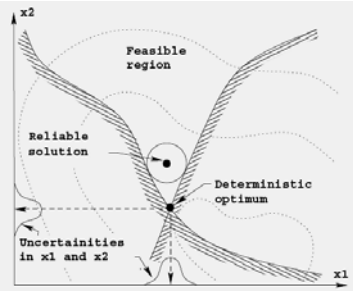
- Use of an archive to store already computed solutions
- Later, select solutions from the archive within δ-neighborhood
- If H points not found, create a solution
- Although reported quicker single-objective runs, multi-objective runs are not quicker
- A diverse set is needed, requiring archive to be large

Reliability-Based Optimization:

Making designs safe against failures

- Deterministic optimum is not usually reliable
- Reliable solution is an interior point
- Chance constraints with a given reliability

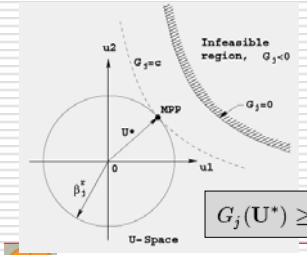
Minimize $\mu_f + k\sigma_f$
Subject to $Pr(g_j(x) \geq 0) \geq \beta_j$
β_j is user-supplied



Nested Optimization:
Check if a solution is reliable

- PMA approach

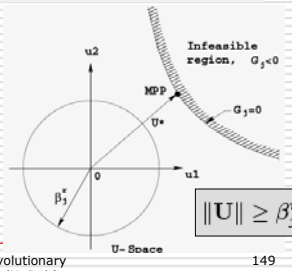
Minimize $G_j(\mathbf{U})$,
Subject to $\|U\| = \beta_j^r$,



$G_j(\mathbf{U}^*) \geq 0.$

- RIA approach

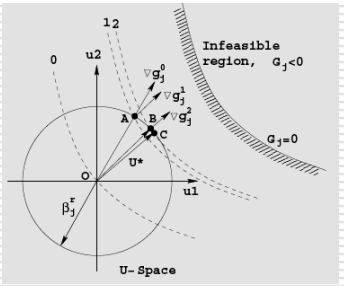
Minimize $\|\mathbf{U}\|$,
Subject to $G_j(\mathbf{U}) = 0.$



$\|\mathbf{U}\| \geq \beta_j^r.$

Proposed Faster Procedure

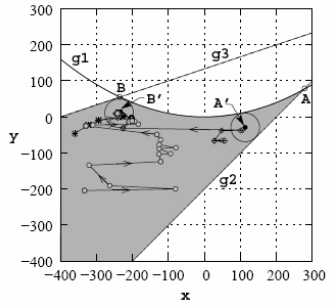
- Single-loop method
- Faster computation of U*
  - PMA: MPP as computed
  - RIA: Apply Newton's search to find MPP



Finding Global Reliable Solution:
Single-objective reliable solution

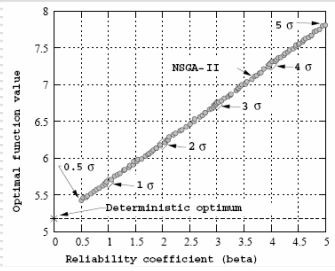
- Multiple deterministic optima
- Reliable solution for local is better
- Classical methods starts from deterministic global optimum
- GAs solve the overall problem and are better

PMA approach is used

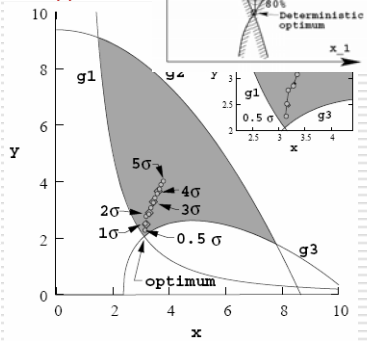


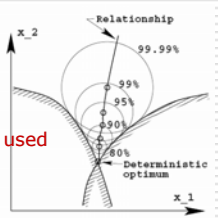
Multiple Reliability Solutions:
Get a better insight

RIA approach is used



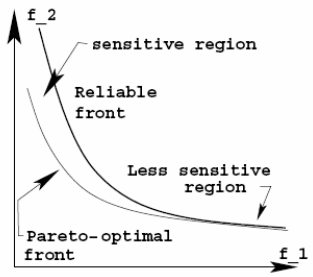
Maximize $x+y$,
Subject to $g_1(x,y) \equiv \frac{1}{20}x^2y - 1 \geq 0,$
$g_2(x,y) \equiv \frac{1}{30}(x+y-5)^2 + \frac{1}{120}(x-y-12)^2 - 1 \geq 0,$
$g_3(x,y) \equiv \frac{80}{x^2+8y+6} - 1 \geq 0,$
$0 \leq x,y \leq 10.$





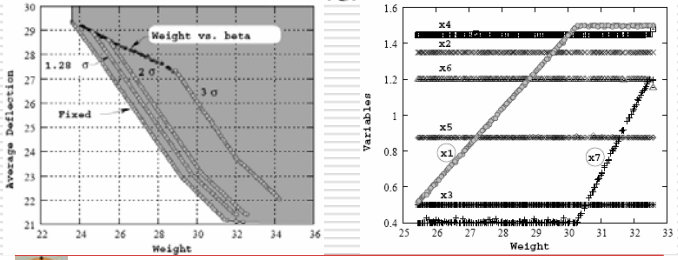
Multi-Objective Reliable Frontier

- Instead of finding deterministic Pareto-optimal front, find reliable front
- Chance constraints
- Objectives as they are
- PMA approach is used



Multi-Objective Reliability-Based Optimization

- Reliable fronts show rate of movement
- What remains unchanged and what



Interactive Optimizers

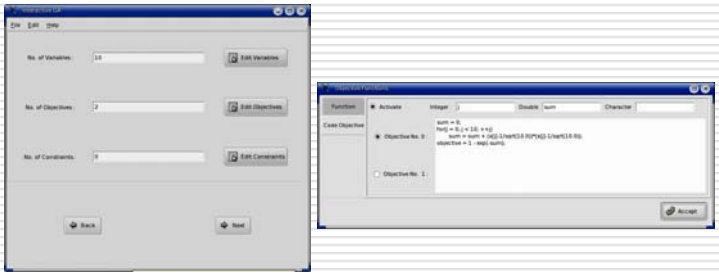
- Needs to involve a decision-maker (DM)
- No concrete study yet
- A few difficulties:
  - The act of a DM makes it a single-obj. problem
  - But, obj. is not known precisely and changes with iteration
  - EMO finds many solutions, but only one is desired
- EMO: Find potentially good solutions – robust, knee-like, etc.
- Classical: Concentrate in an area based DM's preference

I-EMO Being Developed at KanGAL

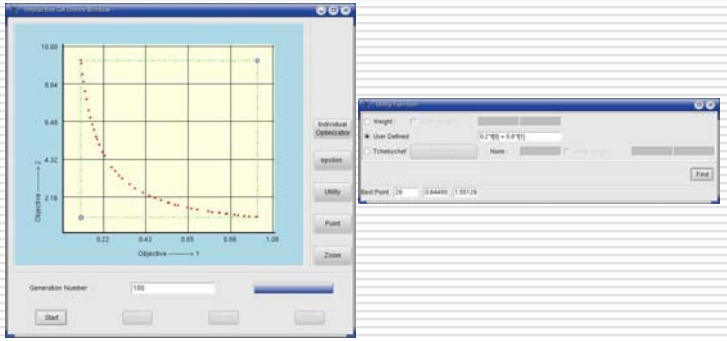
- GUI-based interactive EMO
  - Start with a front obtained with NSGA-II
  - Verify solutions with local searches & classical point-by-point approaches
  - Help choose a particular solution
    - Knee solutions
    - Robust solutions
    - Partial front
    - L_p and other utility-based metrics
- An iterative procedure

I-EMO Snap-shots

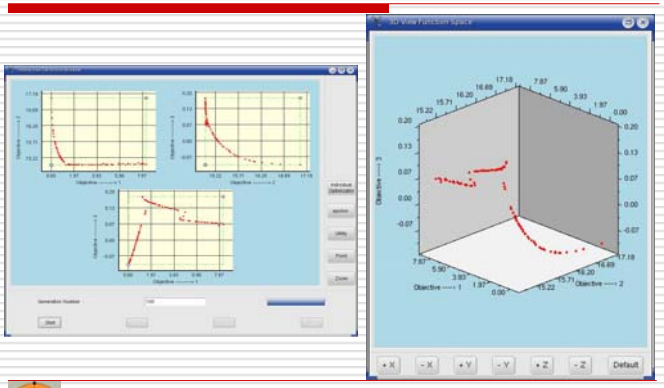
- On a Linux platform



I-EMO (cont.)

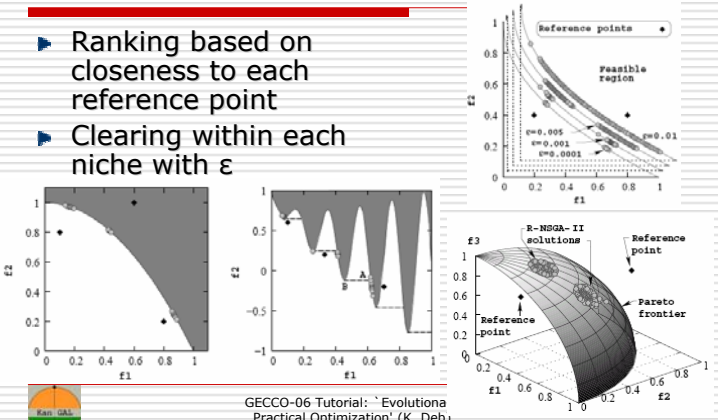


I-EMO (cont.)



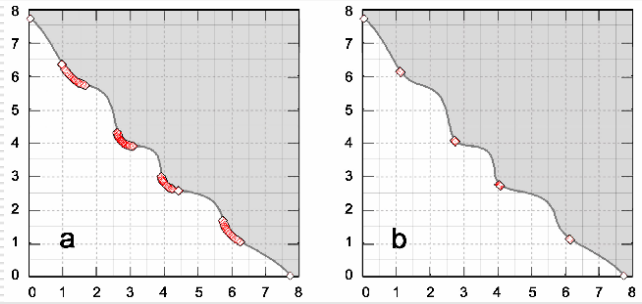
Making Decisions:
Reference Point Based EMO

- Ranking based on closeness to each reference point
- Clearing within each niche with ε



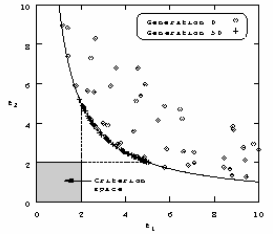
Finding Knee Solutions

Branke et al. (2004) for more details



Goal Programming:
Not to find Optimum

- Target function values are specified
- Convert them to objectives and perform domination check with them



Type	Goal	Objective function
$\leq$	$f_j(x) \leq t_j$	Minimize $\langle f_j(x) - t_j \rangle$
$\geq$	$f_j(x) \geq t_j$	Minimize $\langle t_j - f_j(x) \rangle$
$=$	$f_j(x) = t_j$	Minimize $\lvert f_j(x) - t_j \rvert$
Range	$f_j(x) \in [t_j^l, t_j^u]$	Minimize $\max(\langle t_j^l - f_j(x) \rangle, \langle f_j(x) - t_j^u \rangle)$

Omni-Optimizer:
Motivation from Computation

- Multiple is a generic case, single is specific
- Single objective as a degenerate case multi-objective case
- One algorithm for single and multi-objective problem solving (Deb and Tiwari, 2005)
- Accommodating NFL theorem
  - Single-objective, uni-optimum problems
  - Single-objective, multi-optima problems
  - Multi-objective, uni-optimal front problems
  - Multi-objective, multi-optimal front problems

Single-Objective, Uni-Optimum

- Dominance reduced to simple '<'
- Epsilon-dominance to $f^a < f^b - \varepsilon$
  - Allows multiple solutions within ε to exist
- Elite-preservation is similar to CHC and (μ+λ)-ES

Function	$f(x)$	n	Range	N	Target f^*	Func. Eval. Best	Median	Worst
Rastrigin	$\sum_{i=1}^{n} x_i^2 + 10(1 - \cos(2\pi x_i))$	10	$[-10, 10]$	40	0.01	8,120	15,520	53,480
Rastrigin	$\sum_{i=1}^{n} x_i^2 + 10(1 - \cos(2\pi x_i))$	20	$[-10, 10]$	40	0.01	24,520	41,760	106,440
Schwefel	$418.9829n - \sum_{i=1}^{n} x_i \sin\sqrt{\lvert x_i \rvert}$	10	$[-500, 500]$	16	0.01	6,304	11,360	24,704
Schwefel	$418.9829n - \sum_{i=1}^{n} x_i \sin\sqrt{\lvert x_i \rvert}$	20	$[-500, 500]$	16	0.01	26,128	36,272	82,096

Shinn et al.

12 problems

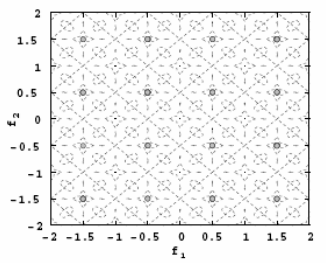
Definition of benchmark function	Variable bound	Optimum value
$f_1 = \sum_{i=1}^{D} \lvert \sin(x_i) + \sin\left(\frac{2x_i}{3}\right) \rvert$	[3, 13]	1.21598D (max)
$f_2 = -\sum_{i=1}^{D-1} \lvert \sin(x_i + x_{i+1}) + \sin\left(\frac{2x_i - x_{i+1}}{3}\right) \rvert$	[3, 13]	$\approx$ 2D (max)
$f_3 = \sum_{i=1}^{D} \lvert x_i + 0.5 \rvert^2$	[-100, 100]	0 (min)
$f_4 = \sum_{i=1}^{D} \lvert x_i^2 - 10\cos(2\pi x_i) + 10 \rvert$	[-5.12, 5.12]	0 (min)
$f_5 = \sum_{i=1}^{D} x_i^2$	[-5.12, 5.12]	0 (min)
$f_6 = \sum_{i=1}^{D} (x_i \sin(10\pi x_i))$	[-1.0, 2.0]	1.85D (max)
$f_7 = \sum_{i=1}^{D} \lvert \frac{\sin(10\pi x_i + \pi)}{10\pi x_i} \rvert$	[-0.5, 0.5]	0 (min)
$f_8 = 20 + e - 20e^{-0.2\sqrt{\frac{\sum_{i=1}^{D} x_i^2}{D}}} - e^{\frac{\sum_{i=1}^{D} \cos(2\pi x_i)}{D}}$	[-30, 30]	0 (min)
$f_9 = 418.9829D - \sum_{i=1}^{D} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	[-500, 500]	0 (min)
$f_{10} = \sum_{i=1}^{D-1} \lvert 100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2 \rvert$	[-5.12, 5.12]	0 (min)
$f_{11} = 6D + \sum_{i=1}^{D} \lvert x_i \rvert$	[-5.12, 5.12]	0 (min)
$f_{12} = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[-600, 600]	0 (min)

Dimension / Function	60	70	80	90	100	Shinn et al. D = 100
f_1	1.36×10^{-3}	2.14×10^{-3}	3.00×10^{-3}	3.95×10^{-3}	5.05×10^{-3}	1.20×10^{02}
f_2	3.32×10^{-1}	3.50×10^{-1}	3.59×10^{-1}	3.72×10^{-1}	3.88×10^{-1}	1.53×10^{02}
f_3	7.00×10^{-2}	1.61×10^{-1}	2.96×10^{-1}	5.29×10^{-1}	8.91×10^{-1}	6.21×10^{02}
f_4	3.75×10^{-1}	5.00×10^{-1}	6.40×10^{-1}	7.85×10^{-1}	9.37×10^{-1}	2.13×10^{02}
f_5	2.09×10^{-4}	3.67×10^{-4}	6.72×10^{-4}	1.08×10^{-3}	1.90×10^{-3}	1.60×10^{00}
f_6	1.74×10^{-1}	2.17×10^{-1}	2.68×10^{-1}	3.05×10^{-1}	3.41×10^{-1}	1.31×10^{02}
f_7	1.20×10^{-3}	1.50×10^{-3}	1.81×10^{-3}	2.14×10^{-3}	2.40×10^{-3}	6.50×10^{-1}
f_8	1.46×10^{-2}	1.94×10^{-2}	2.29×10^{-2}	2.52×10^{-2}	2.64×10^{-2}	3.69×10^{00}
f_9	9.62×10^{0}	1.16×10^{1}	1.37×10^{1}	1.48×10^{1}	1.63×10^{1}	8.01×10^{03}
f_{10}	3.70×10^{0}	4.22×10^{0}	4.68×10^{0}	5.09×10^{0}	5.57×10^{0}	2.08×10^{03}
f_{11}	0.00×10^{0}	0.00×10^{0}	1.26×10^{-4}	1.26×10^{-4}	1.26×10^{-4}	4.39×10^{01}
f_{12}	1.37×10^{-2}	1.42×10^{-2}	1.34×10^{-2}	1.28×10^{-2}	1.29×10^{-2}	3.29×10^{01}

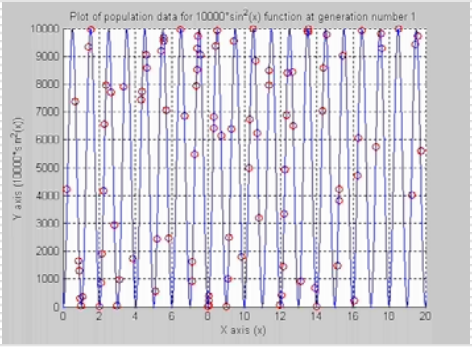
Single-Objective, Multi-Optima

- Variable-space niching help find multiple solutions
- Weierstrass function
  - 16 minima with f=0

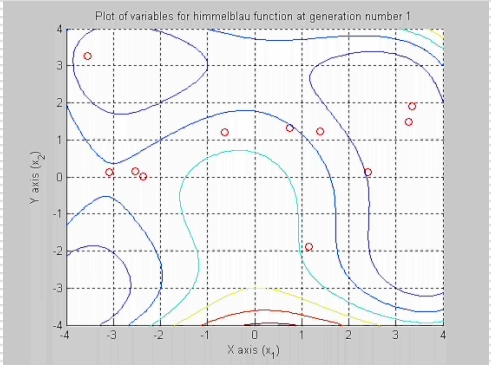
$$f(x) = \sum_{i=1}^{D} \left[\sum_{k=0}^{k_{max}} \left[a^k cos(2\pi b^k x_i) \right] \right], \; x_i \in [-2, 2]$$



$10^4 Sin^2(x)$: 20 Minima

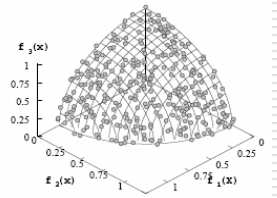


Himmelblau's Function: 4 Minima



Multi-Objective, Uni-Pareto front

- Constrained and unconstrained test problems



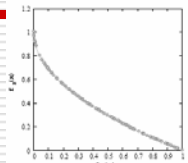


Fig. 7. Efficient points for ZDT4.

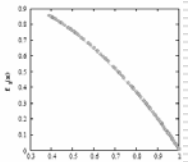


Fig. 8. Efficient points for ZDT6.

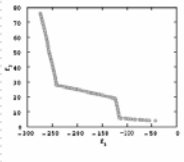


Fig. 9. Efficient points for OSY.

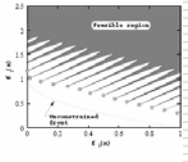


Fig. 10. Efficient points for CTP4.

More Results

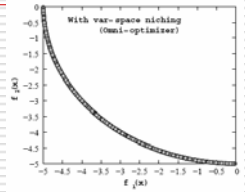
- Comparable performance to existing EMO methods

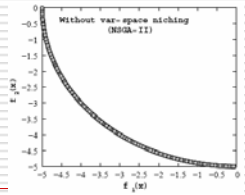
Test Problem	MOEA	Convergence measure		Sparsity		Hyper-volume	
		Avg.	SD	Avg.	SD	Avg.	SD
ZDT1	NSGA-II	0.00054898	6.62e-05	0.858	0.0202	0.8701	3.85e-04
	C-NSGA-II	0.00061173	7.86e-05	0.994	0.0043	0.8713	2.25e-04
	PESA	0.00053481	12.62e-05	0.754	0.0331	0.8680	6.76e-04
	SPEA2	0.00100589	12.06e-05	0.999	0.0014	0.8708	1.86e-04
	e-MOEA	0.00039545	1.22e-05	0.991	0.0050	0.8702	1.86e-04
	OMNI	0.01089320	9.14e-04	0.922	0.0260	0.8546	9.28e-04
	OMNI*	0.00536259	4.32e-04	0.911	0.0251	0.8626	6.54e-04
	OMNI^0	0.00416721	3.80e-04	0.960	0.0151	0.8649	6.00e-04
ZDT4	NSGA-II	0.00639002	0.0043	0.958	0.0328	0.8613	0.00610
	C-NSGA-II	0.00618386	0.0744	0.998	0.0029	0.8558	0.00301
	PESA	0.00730242	0.0047	0.708	0.0352	0.8566	0.00710
	SPEA2	0.00769278	0.0043	0.989	0.0132	0.8609	0.00536
	e-MOEA	0.00259063	0.0006	0.987	0.0076	0.8509	0.01537
	OMNI	1.69495000	1.0766	0.765	0.0708	0.4819	0.71756
	OMNI*	0.01105010	0.0097	0.899	0.0389	0.8483	0.01779
	OMNI^0	0.00545542	0.0025	0.958	0.0194	0.8566	0.01522
ZDT6	NSGA-II	0.07896111	0.0067	0.815	0.0157	0.3959	0.00894
	C-NSGA-II	0.07940667	0.0110	0.995	0.0029	0.3990	0.01154
	PESA	0.06415652	0.0073	0.748	0.0345	0.4145	0.00990
	SPEA2	0.00573584	0.0009	0.998	0.0029	0.4968	0.00117
	e-MOEA	0.06792800	0.0118	0.996	0.0023	0.4112	0.01573
	OMNI	0.05342110	0.0055	0.863	0.0228	0.4176	0.00645
	OMNI*	0.02528650	0.0024	0.872	0.0288	0.4511	0.00291
	OMNI^0	0.021418	0.0023	0.956	0.0143	0.4559	0.00275

Multi-Objective, Multi-Optima

$$\text{Minimize } f_1(x) = \sum_{i=1}^{n} \sin(\pi x_i),$$
$$\text{Minimize } f_2(x) = \sum_{i=1}^{n} \cos(\pi x_i),$$
$$0 \le x_i \le 6, \quad i = 1, 2, \ldots, n.$$

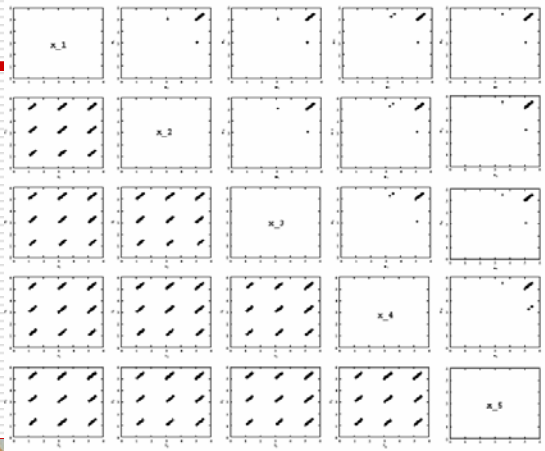
- Nine regions leading to the same Pareto-optimal front
- Multiple solutions cause a single Pareto-optimal point





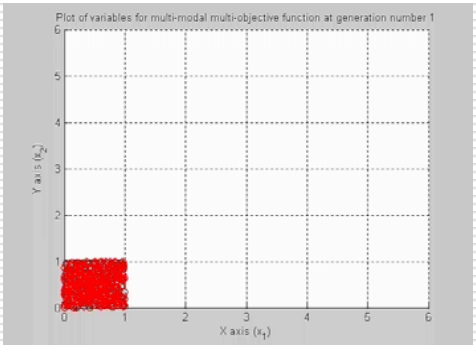
Nine Optimal Regions

Omni-optimizer



NSGA-II

Nine Optimal Fronts



Conclusions

- Most application activities require optimization routinely
- Classical methods provide foundation
  - If applicable, good accuracy is achievable
- Evolutionary methods enable applicability to near-optimality
  - Try when classical methods fail
  - Parallel search ability
- A good optimization task through EAs and local search

How are EAs Unique?

- Broader applicability
  - Mixed variables, non-linearity, non…
  - Constraint handling
- Easy to use for distributed computing
- Beyond optimization
  - Learn your problem better
  - What makes a solution optimum, robust, or reliable?

Practical Optimization
Final words

- Seems impossible to have one algorithm for many practical problems
- Needs a customized optimization
  - Calls for collaborations
- An application requires
  - Domain-specific knowledge
  - Thorough knowledge in optimization basics
  - Good knowledge in optimization algorithms
  - Good computing background
- Have successful show-cases, have a practice choose one

What we have not discussed?

- Combinatorial optimization
  - Non-linearity, large dimension
- Problem Formulation aspects
  - Objectives, constraints, etc.
- Very large computational overhead
  - One evaluation takes a day or more
- Parallel EAs
- Termination criteria

Thank You for Your Attention

- Acknowledgement:
  - KanGAL students, staff and collaborators
  - GM, GE, Honda R&D, STMicroelectronics
  - Governmental Research Labs

For further information:

http://www.iitk.ac.in/kangal
Email: deb@iitk.ac.in