UF UNIVERSITY of FLORIDA
The Foundation for The Gator Nation

Gradient-free Optimization

Nelder-Mead Sequential Simplex Algorithm

How to optimize when a function is not smooth

Gradient-free method

LOCAL SEARCH ALGORITHMS

- Most local search algorithms are based on derivatives to guide the search.
- For differentiable function it has been shown that even if you need to calculate derivatives by finite differences, derivative-based algorithms are better than ones that do not use derivatives.
- However, we will start with Nelder-Mead sequential simplex algorithm that can deal with non-differentiable functions, and even with some discontinuities.

NON-GRADIENT BASED METHODS

- Simplest in a random search
- Easy to implement
- Very robust
- Very efficient
- Improve random search by adding some logic
  - Ex) DoE search with move-limits
  - Referred to as a structured random search
- We will consider two structured random searches in this course

NON-GRADIENT BASED METHODS *cont*.

- Non-gradient based optimization algorithms have gained a lot of attention recently
  - Easy to program
  - Global properties
  - Require no gradient information
  - High computational cost
  - Tuned for each problem
- Typically based on some physical phenomena
  - Genetic algorithms
  - Simulated annealing
  - Particle swarm optimization

NON-GRADIENT BASED METHODS *cont*.

- Can be classified as a structured random search
- Does not move from one design point to the next
  - Makes use of a population of design points
- Numerically robust
- Increased chances of finding global or near global optimum designs
- Provides a number of good designs instead of a single optimum

NELDER MEAD SEQUENTIAL SIMPLEX ALGORITHM

- An old local-search algorithm that contains the ingredients of modern search techniques:
  - No derivatives
  - Population based
  - Simple idea that does not require much mathematics
- Basis for Matlab's fminsearch testimonial to its robustness.
- Simplex is the simplest body in *N* dimensions
- Has *N*+1 vertices (triangle in 2D and tetrahedron in 3D)

NM Simplex Algorithm

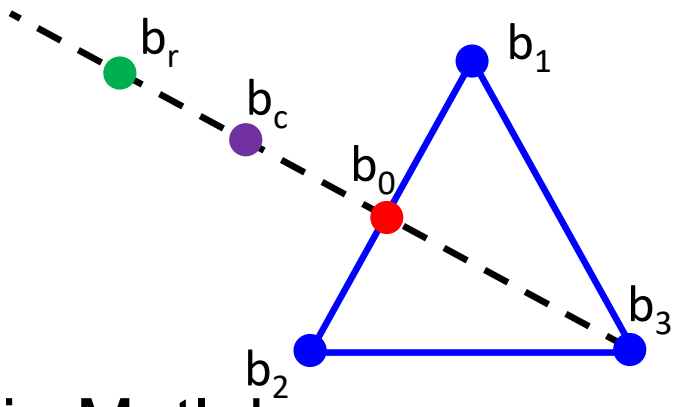
- Start with a randomly generated simplex (n+1 vertices)
- Reflection: the point with highest objective value is reflected to the opposite side (preserve volume, nondegeneracy)
- Expansion: If the reflected point is the best, expand the simplex further(possibly optimum is further in that direction)
- Contraction: If the reflected point is worse, contract the simplex (possible valley floor)
- Reduction: If the contracted point is still worst, reduce the size of the simplex (possibly the best point is within the simplex)

SEQUENTIAL SIMPLEX METHOD

- In N dimensional space start with N+1 vertices of a selected simplex, evaluate the function there and order points by function value: $f(\mathbf{b}_1) \le f(\mathbf{b}_2) \le \cdots \le f(\mathbf{b}_{N+1})$

1. Calculate $\mathbf{b}_0$, the center of gravity of all the points except $\mathbf{b}_{N+1}$

$$\mathbf{b}_0 = \frac{1}{N}\sum_{i=1}^{N}\mathbf{b}_i$$



2. Reflect $\mathbf{b}_{N+1}$ about $\mathbf{b}_0$

3. Reflect worst point about c.g.

$$\mathbf{b}_r = \mathbf{b}_0 + \alpha(\mathbf{b}_0 - \mathbf{b}_{N+1}) \qquad \alpha\text{=1 in Matlab}$$

4. If new point is better than 2nd worst, but not best, use to replace worst and go back to step 1. ($\mathbf{b}_{N+1} = \mathbf{b}_r$)

- Read about expansion and contraction

EXPANSION AND CONTRACTION

5. Expansion (when the new point is best).

   If $f(\mathbf{b}_r) < f(\mathbf{b}_1)$, then move further

$$\mathbf{b}_e = \mathbf{b}_0 + \gamma(\mathbf{b}_0 - \mathbf{b}_{N+1}) \qquad \gamma = 2 \text{ in Matlab}$$

   If $f(\mathbf{b}_e) < f(\mathbf{b}_r)$, then replace $\mathbf{b}_{N+1}$ by $\mathbf{b}_e$. Otherwise replace $\mathbf{b}_{N+1}$ by $\mathbf{b}_r$. Go back to Step 1.

6. Contraction (new point is worst than 2nd worst).

   If $f(\mathbf{b}_r) >= f(\mathbf{b}_N)$, then compute contracted point

$$\mathbf{b}_c = \mathbf{b}_0 + \rho(\mathbf{b}_0 - \mathbf{b}_{N+1}) \qquad \text{in Matlab } \rho=0.5$$

   If $f(\mathbf{b}_c) < f(\mathbf{b}_{N+1})$, then replace $\mathbf{b}_{N+1}$ by $\mathbf{b}_c$.

7. If the contracted point is not better than worst, then contract all points about the best one

$$\mathbf{b}_i = \mathbf{b}_1 + \sigma(\mathbf{b}_i - \mathbf{b}_1), \quad i = 2,\ldots,N+1 \qquad \text{In Matlab } \sigma = 0.5$$

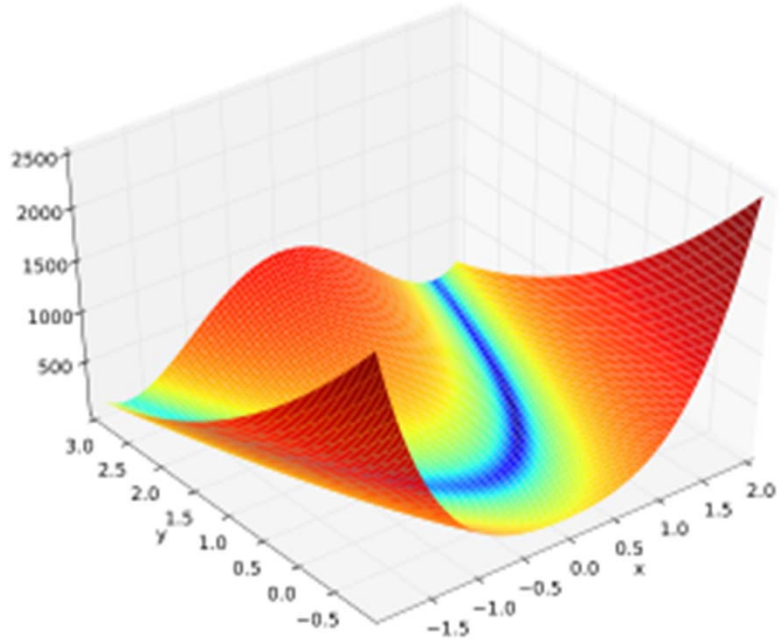
EXERCISES: NELDER MEAD OPERATORS

- For a 2D problem, the current simplex has $f(0,1)=3$, $f(0,0)=1$, $f(1,0)=2$. Where will you evaluate f next?
- If the next two points gave us function values of 4 and 5, respectively, where will you evaluate the function next?
- If instead, the next point gave us a function value of 0, where will you evaluate the function next?

ROSENBROCK BANANA FUNCTION

- Commonly called a banana function
- Many versions of Rosenbrock function exists, we use the one from Wikeipedia (minimum at (1,1))

$$f(\mathbf{b}) = 100(b_2 - b_1^2)^2 + (1 - b_1)^2$$



MATLAB fminsearch for BANANA FUNCTION

- Matlab code for 20 optimization iterations

```
global z1 z2 yg count
count =1;
options=optimset('MaxFunEvals',20)
[b,fval] = fminsearch(@banana,[-1.2,
1],options)
mat=[z1;z2;yg]
```

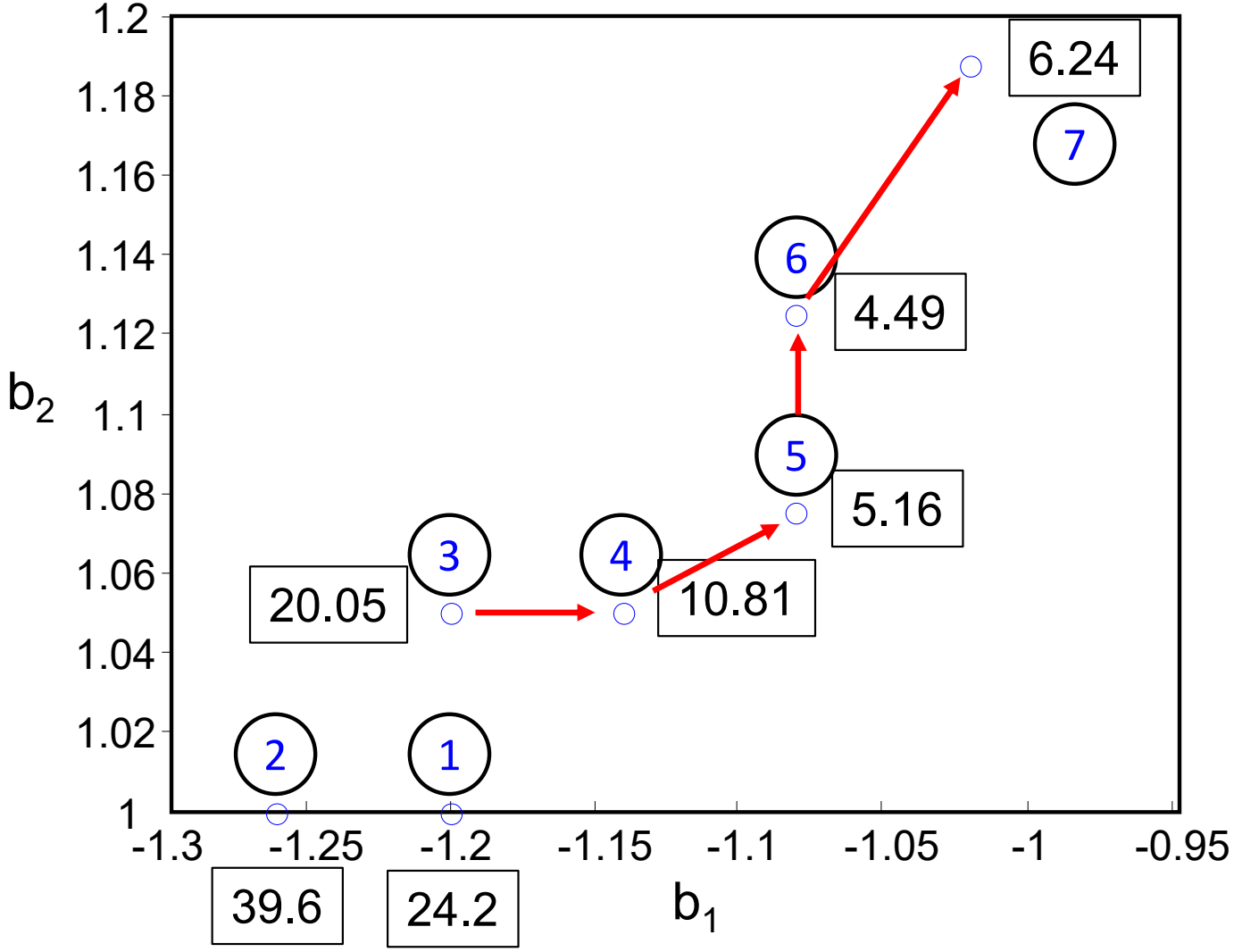
```
function [y]=banana(b)
global z1 z2 yg count
y=100*(b(2)-b(1)^2)^2+(1-b(1))^2;
z1(count)=b(1); z2(count)=b(2);
yg(count)=y;
count=count+1;
```

```
mat =
  Columns 1 through 8
   -1.200   -1.260   -1.200   -1.140   -1.080   -1.080   -1.020   -0.960
    1.000    1.000    1.050    1.050    1.075    1.125    1.1875    1.150
   24.20     39.64    20.05    10.81     5.16     4.498    6.244     9.058

  Columns 9 through 16
   -1.020   -1.020   -1.065   -1.125   -1.046   -1.031   -1.007   -1.013
    1.125    1.175    1.100    1.100    1.119    1.094    1.078    1.113
    4.796    5.892    4.381    7.259    4.245    4.218    4.441    4.813
```

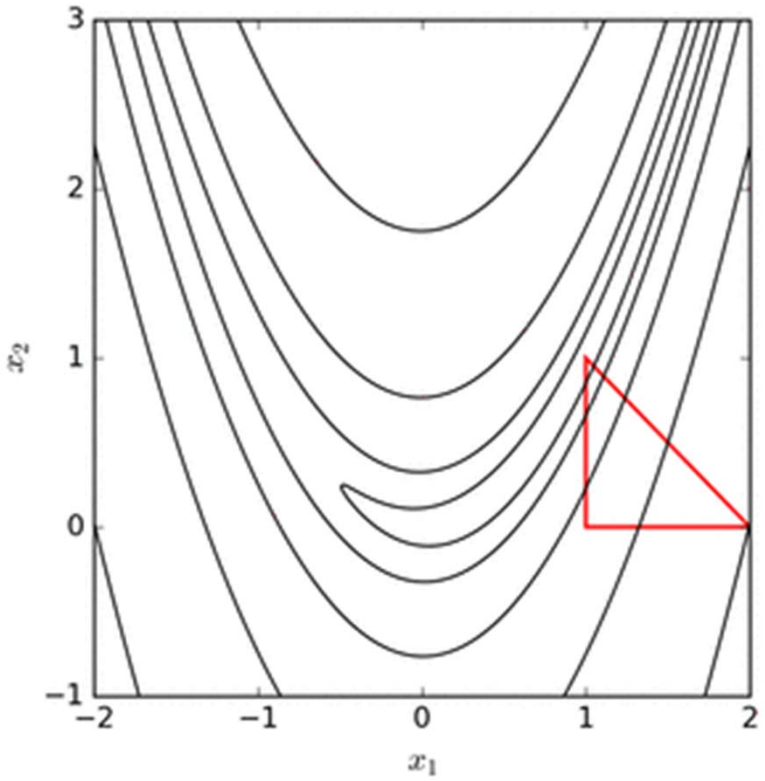
REFLECTION AND EXPANSION

$$f(\mathbf{b}) = 100(b_2 - b_1^2)^2 + (1 - b_1)^2$$

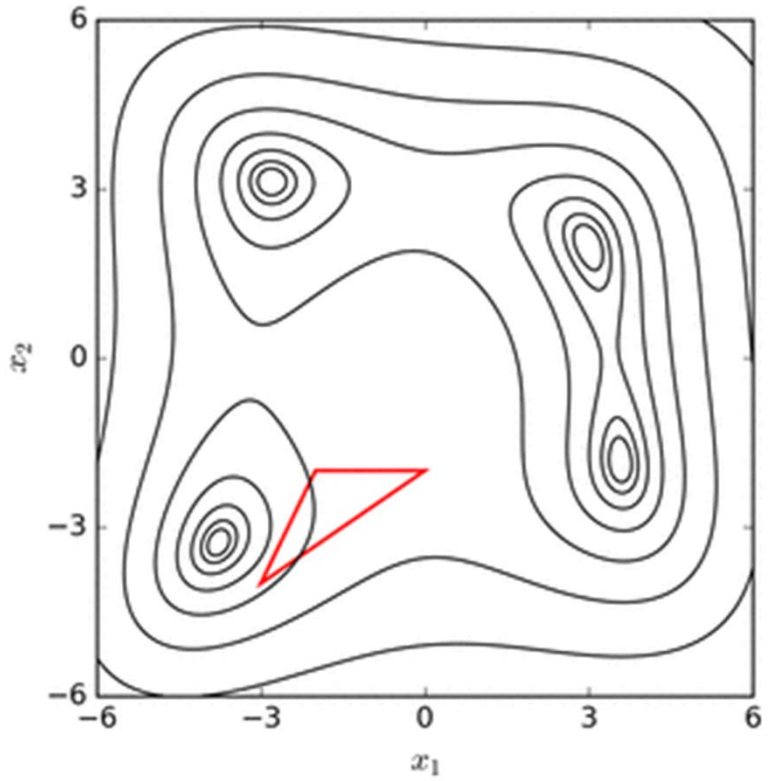


COMPLETE SEARCH EXAMPLES FROM WIKIPEDIA

- http://en.wikipedia.org/wiki/Nelder-Mead_method
- Shows progress for the Banana function as well as for Himmelblau's function.



Banana function



Himmelblau's function

EXERCISE: fminsearch

- Track and plot the first few iterations of fminsearch on Himmelblau's function starting from (1,1).

UF UNIVERSITY of FLORIDA
The Foundation for The Gator Nation

Global Optimization

DIRECT Method

How to search optimum over the entire domain?

Exploration and exploitation

GLOBAL OPTIMIZATION ISSUES

- Optimization problem is NP-hard
- No-free-lunch theorem (Wolpert and Macready)
  - No single algorithm can do well on all problems
  - If an algorithm is improved for one problem, it will suffer for others.
- Great opportunity for engineers to use problem knowledge to tailor algorithms.
- Big headache for journals because they get many worthless new algorithms.

GLOBAL OPTIMIZATION *cont.*

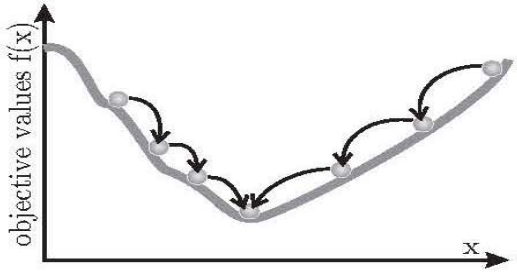


Fig. 1.19.a: Best Case

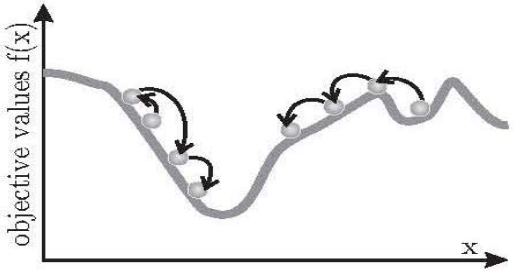


Fig. 1.19.b: Low Variation

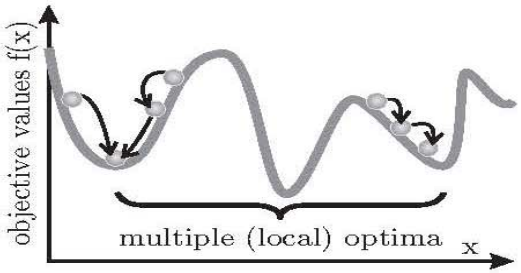


Fig. 1.19.c: Multimodal

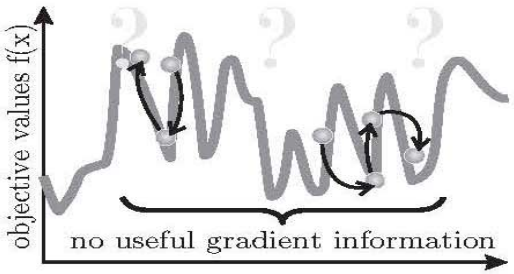


Fig. 1.19.d: Rugged

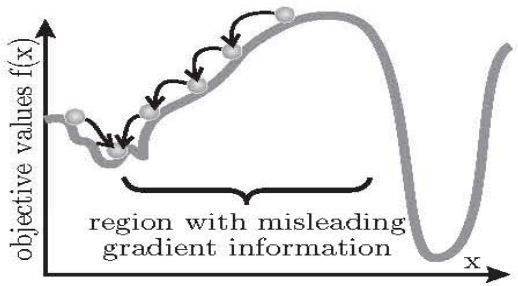


Fig. 1.19.e: Deceptive

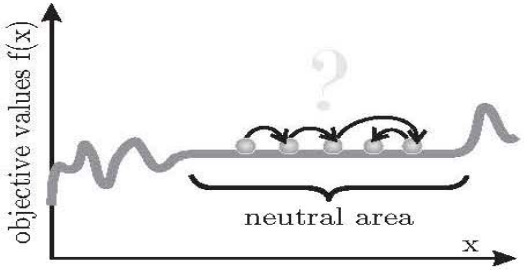


Fig. 1.19.f: Neutral

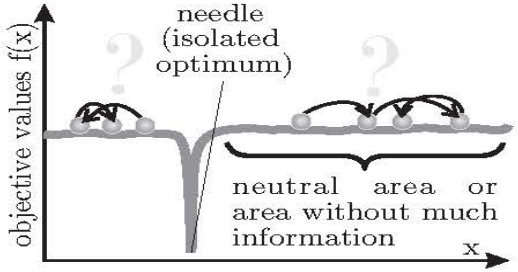


Fig. 1.19.g: Needle-In-A-Haystack

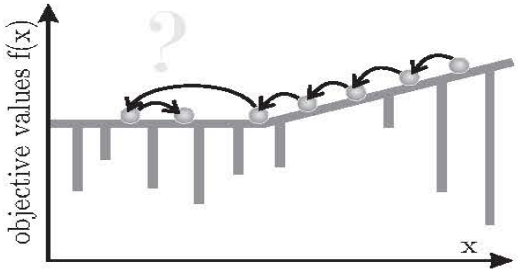


Fig. 1.19.h: Nightmare

Global optimization algorithms by Thomas Weise

CLASSIFICATION OF GLOBAL OPTIMIZATION ALGORITHMS

- The most popular algorithms imitate natural processes, including genetic algorithms, particle swarm optimization, ant colony optimization, and simulated annealing.
- They rely on randomness for exploration, so every time you run them you may get a different result.
- DIRECT is an example of a systematic deterministic exploration of the design space.
- Adaptive sampling algorithms based on surrogates, such as EGO, are gaining popularity.

EXERCISE: GLOBAL OPTIMIZATION

- What does it mean that global optimization is NP hard?
- What is the no-free-lunch principle, and how does it affect engineering optimization.
- When should we use local optimizers to solve global optimization problems and when we should not?

Lipschitzian Optimization

- A sequential method seeking the global minimum of a function (Shubert, 1972)
- Definition: A function $f: D \in R^d \rightarrow R$ is called Lipschitz-continuous if there exists a positive constant $K \in R^+$ such that

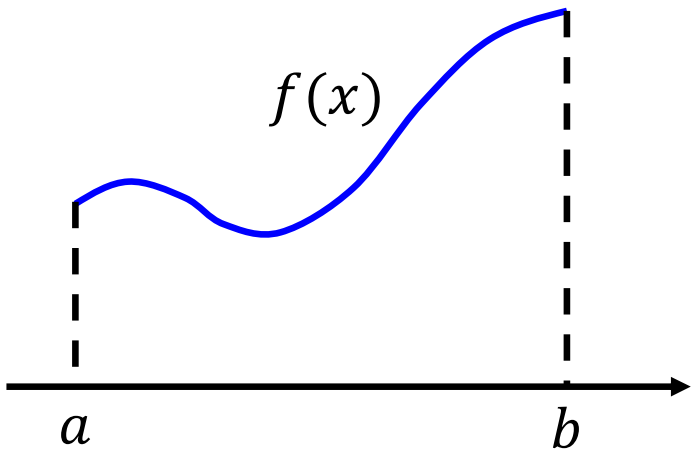
$$|f(x) - f(x')| \leq K|x - x'|, \qquad \forall x, x' \in D$$

- K: Lipschitz constant
- For $x \in [a, b]$, replace a, b with x'

$$f(x) \geq f(a) - K(x - a)$$

$$f(x) \geq f(b) + K(x - b)$$

KH [2]1



KH [2]1 Assume f(x) < f(a) and f(b)

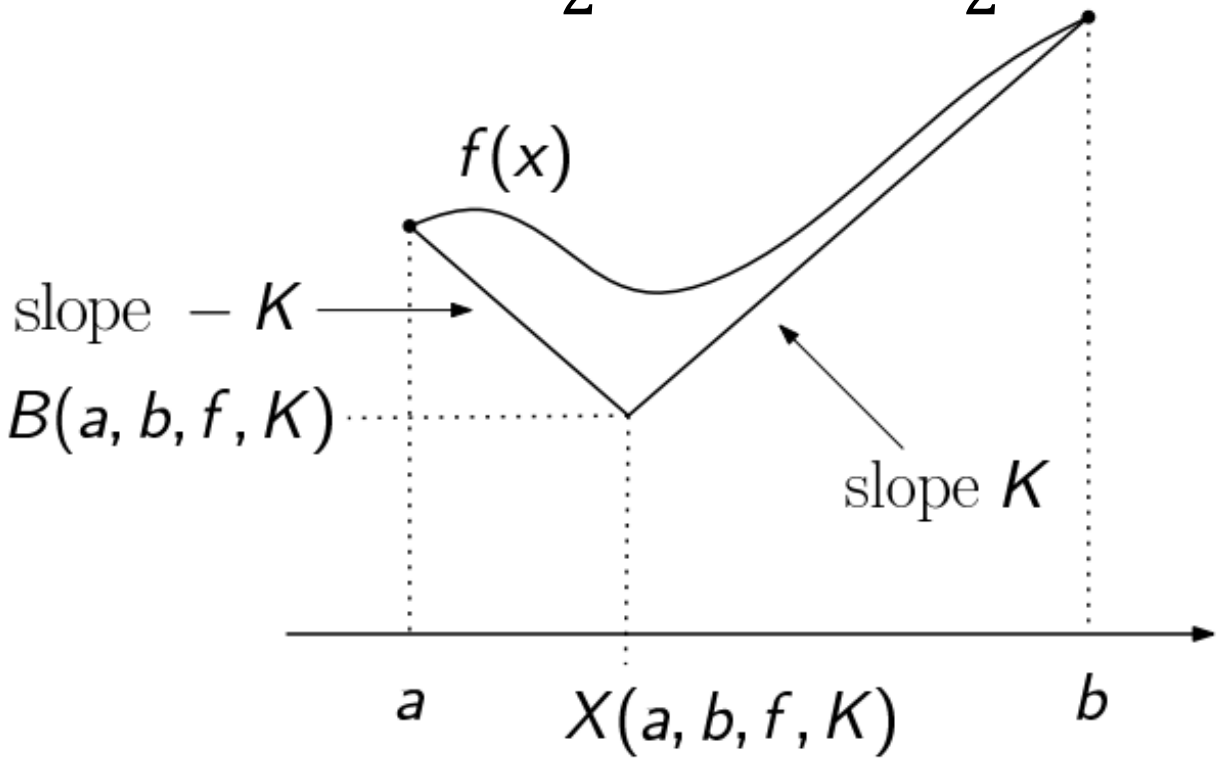
Kim,Nam Ho, 9/9/2020

Lipschitzian Optimization *cont.*

- Straight lines from a and b with slope K and find intersection (x_1, the possible lowest point)

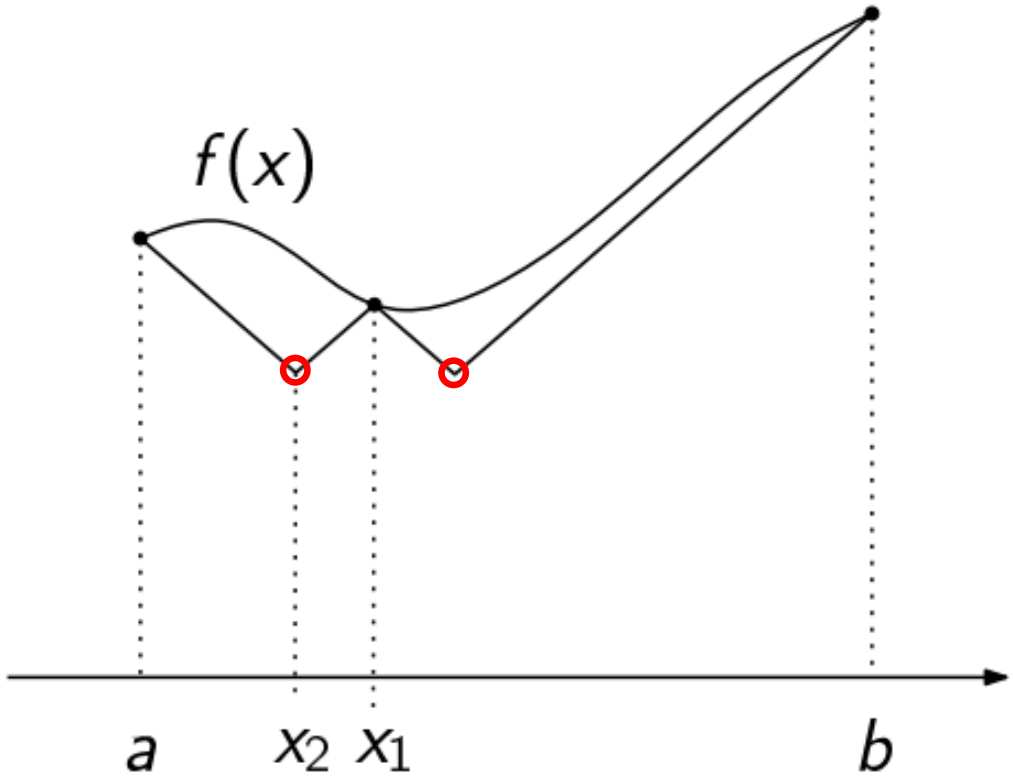
$$X(a,b,f,K) = \frac{a+b}{2} + \frac{f(a)-f(b)}{2K}$$

$$B(a,b,f,K) = \frac{f(a)+f(b)}{2} - \frac{K(b-a)}{2}$$



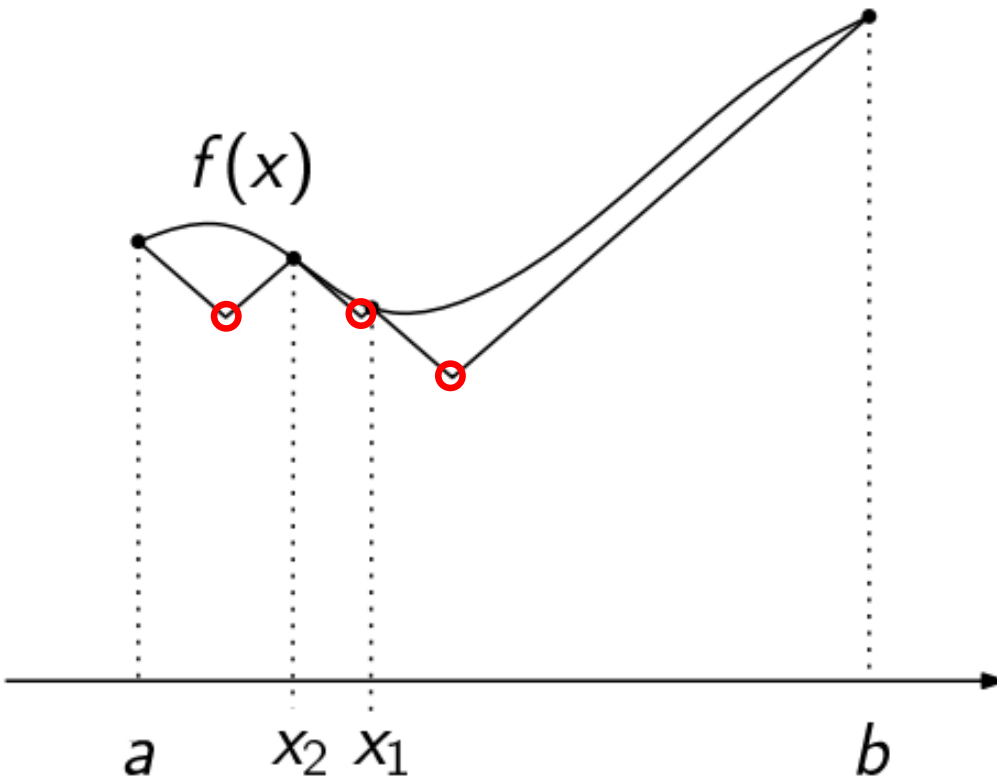
Lipschitzian Optimization *cont.*

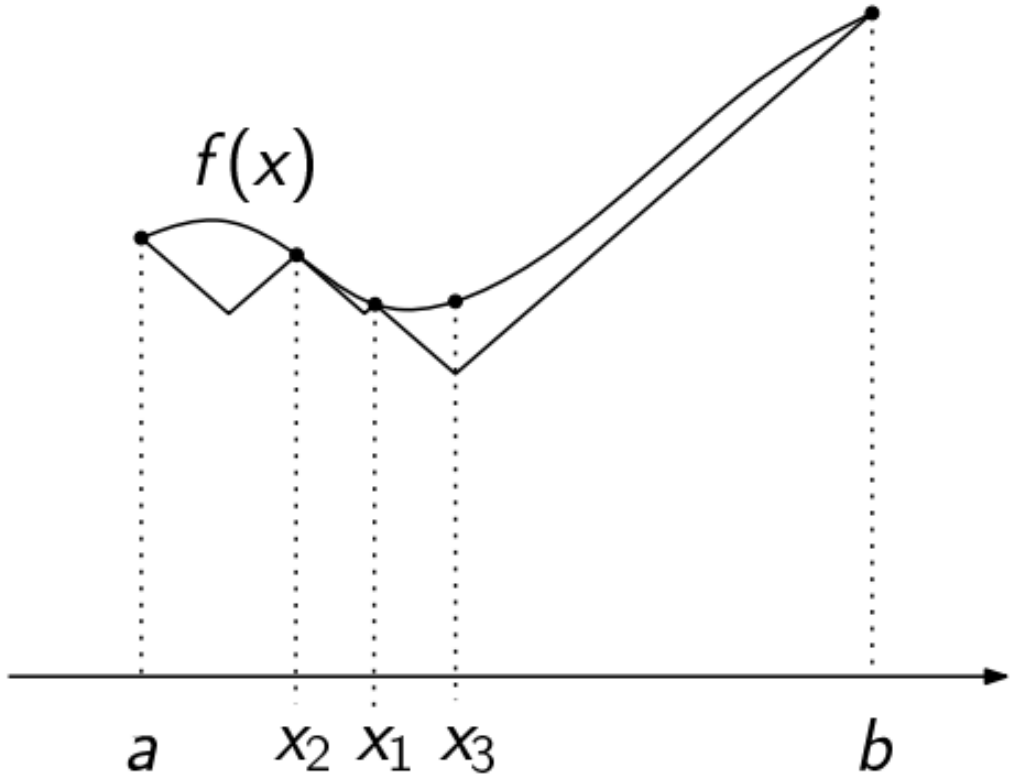
- Evaluate actual function at x_1
- Repeat the process at the two intervals that were created by adding x_1, that is (a, x_1) and (x_1, b)
  - Between two intersection points, choose the lowest one as x_2



Lipschitzian Optimization *cont.*

- Evaluate actual function at x_2
- Repeat the process at the additional two intervals that were created by adding x_2, that is (a, x_2) and (x_2, x_1)
  - Among all intersection points, choose the lowest one as x_3



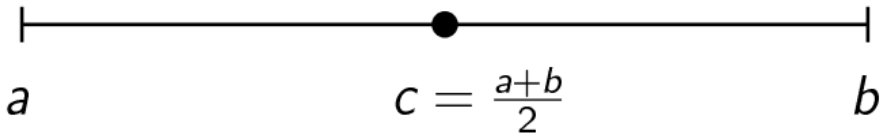


LIPSCHITZIAN OPTIMIZATION

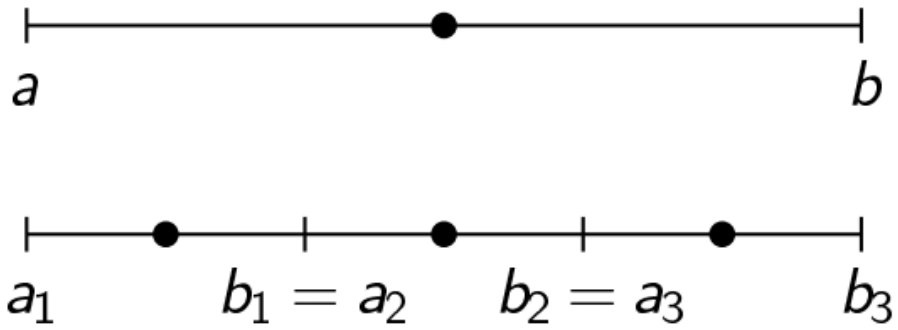
- Pros
  - Global search possible
  - Deterministic, no need for multiple runs, reproducible
  - Few parameters for fine-tuning, except for K
- Cons
  - Lipschitz constant has to be known: K might not be easily accessible
  - Convergence speed: K is a trade-off between global and local search
  - Computational complexity for high dimensions: Initially, need to evaluate the function at all corners of a hyperrectangle $\sim O(2^d)$

DIRECT in 1D

- Jones, Perttunen, Stuckman (1993), Lipschitzian optimization without the Lipschitz constant
- DIviding RECTanbles
  - DIRECT was inspired by Lipschitzian optimization.
  - Optimizer divides space into boxes and sample the function at the center of rectangles



  - When dividing a box, use trisection so that previous point is useful

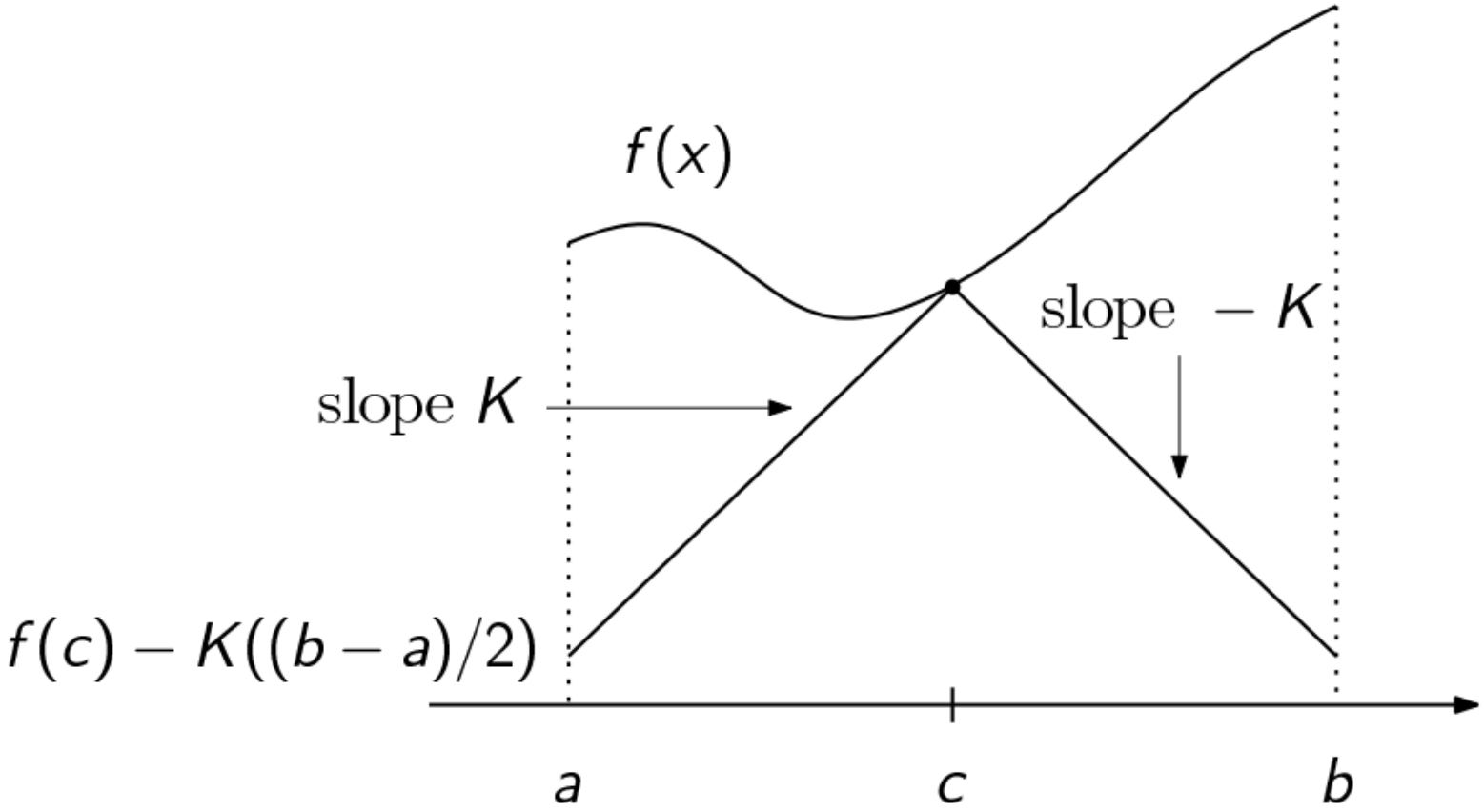


Lipschitz Bound

- DIRECT uses the following Lipschitz bound

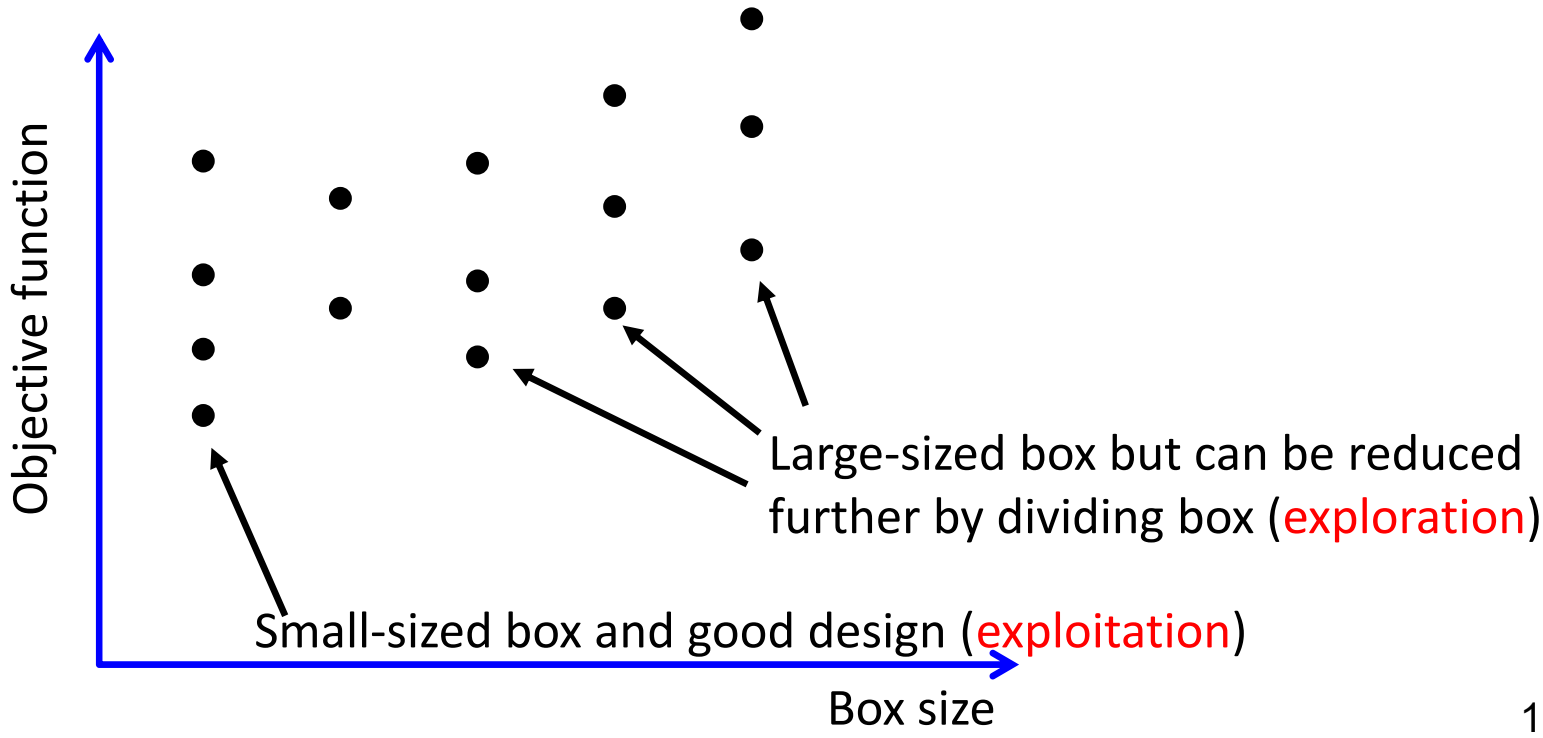
$$f(x) \geq f(c) + K(x - c) \quad \text{for } x \leq c$$

$$f(x) \geq f(c) - K(x - c) \quad \text{for } x \geq c$$



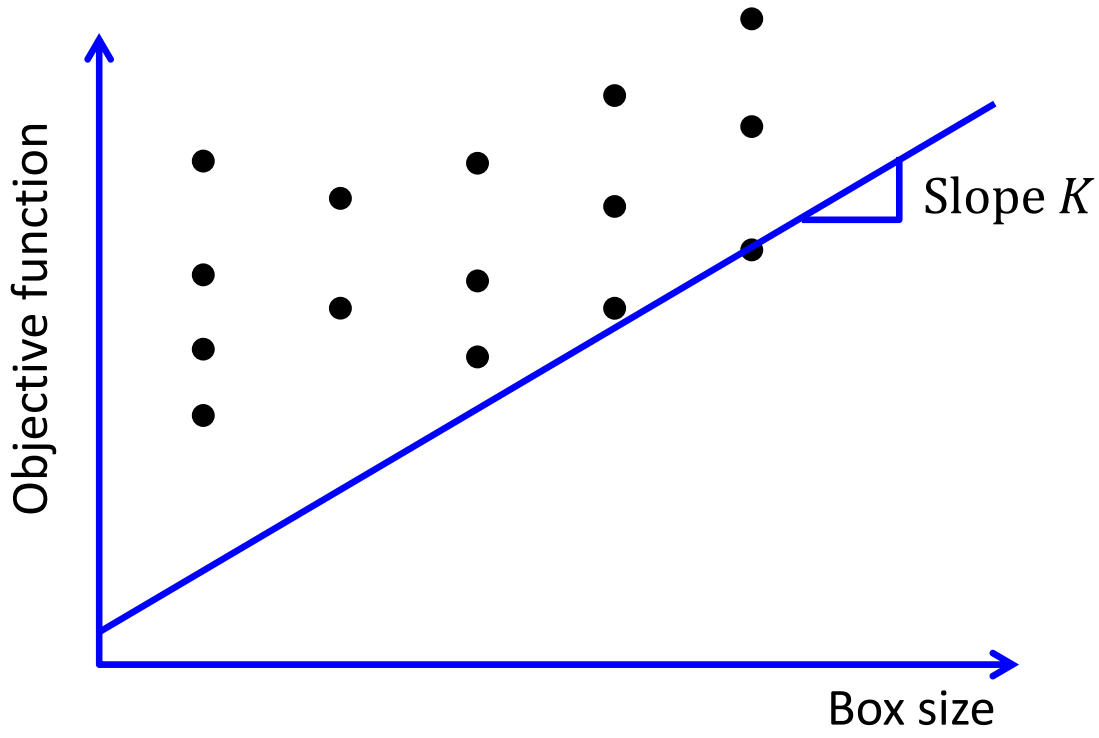
Direct Algorithm

- At a given iteration the design space is divided into boxes, and we have the value of the function at the center of each box
- We then look at a plot of box sizes (determined by the largest dimension of the box) vs. function value and look for boxes that should be divided



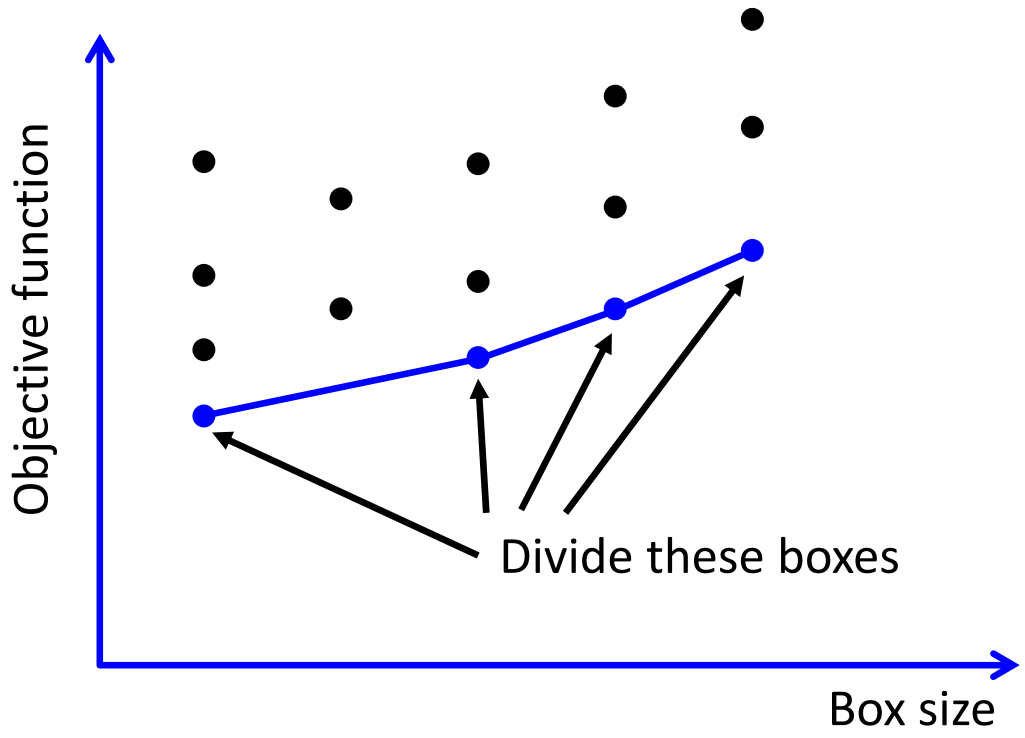
Direct Algorithm *cont*.

- This can be seen as a Pareto front of multi-objective optimization problem.
- Those points on the Pareto front correspond to Lipschitzian optimization for all possible Lipschitz constants



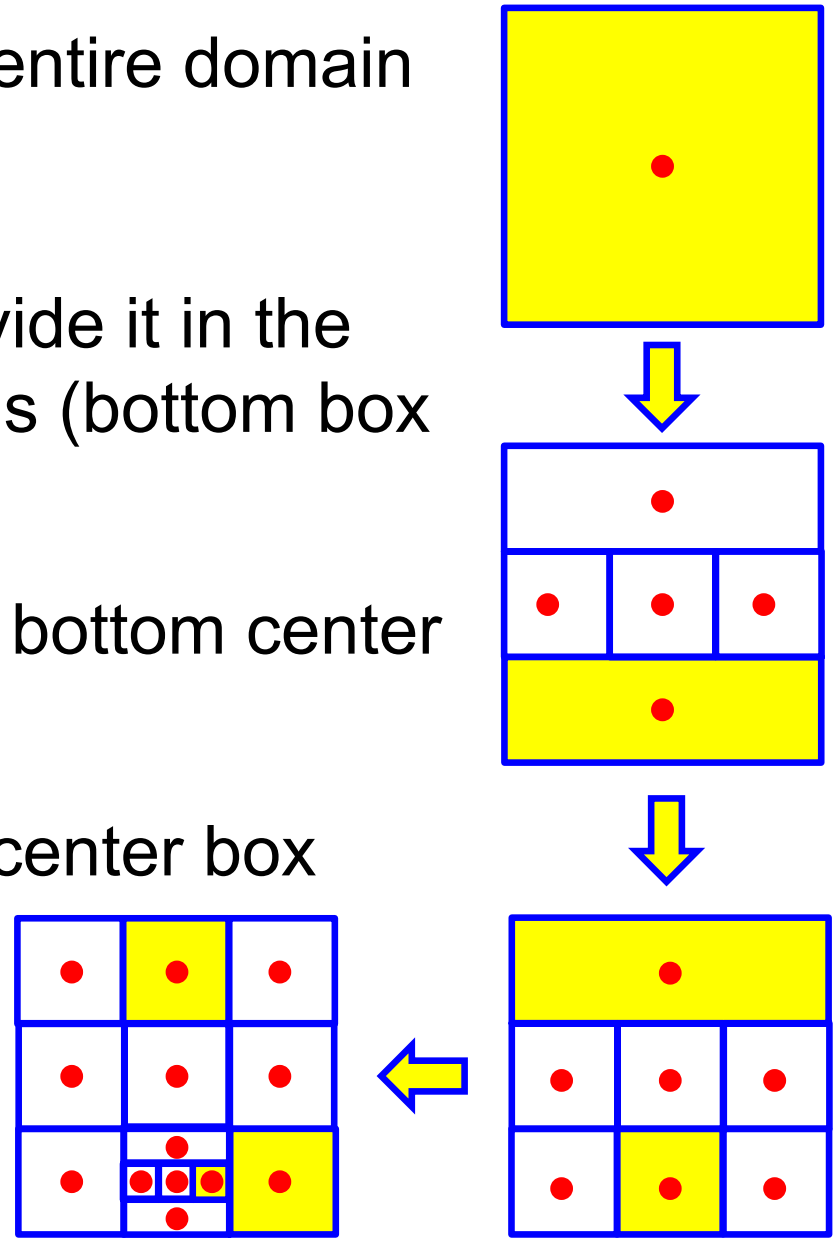
Direct Algorithm *cont*.

- Sample at all the points that are on the Pareto front of size and function value, which includes all the points that are on the bottom part of the convex hull



DIRECT Box Division

- Start with a center point of the entire domain
  - Yellow box is potential optimum
- Since the domain is square, divide it in the vertical and horizontal directions (bottom box is the best)
- Divide the bottom box (top and bottom center box are the best)
- Divide the top box and bottom center box (top center, bottom right, bottom center right boxes are the best)



EXPLORATION VS. EXPLOITATION

- DIRECT uses convex hull of box sizes to balance exploitation vs. exploration
- With enough function evaluations every region in design space will be explored
- This is clearly not feasible for high-dimensional spaces
- Cox's paper compares DIRECT to repeated local optimization with random start

MATLAB CODE direct.m

```
function [ret_minval,final_xatmin,history] = Direct...
    (Problem,bounds,opts,varargin)
% Written by : Dan Finkel (definkel@unity.ncsu.edu)
% Parameters:
%  Problem.f              = Objective function handle
%  Problem.numconstraints = number of constraints
%  Problem.constraint(i).func    = i-th constraint handle
%  Problem.constraint(i).penalty = penalty parameter for
%  bounds    - an n x 2 vector of the lower and upper bounds.
%  opts      - (optional) MATLAB structure.
%   opts.ep         = Jones factor                  (default is 1e-4)
%   opts.maxevals   = max. number of function evals (default is 20)
%   opts.maxits     = max. number of iterations     (default is 10)
%   opts.maxdeep    = max. number of rect. divisions(default is 100)
%   opts.showits    = 1 if disp. stats shown, 0 oth.(default is 1)
%   opts.tol        = tolerance for term. if tflag=1(default is 0.01)
%OUT: minval    -  minimum value found
%     xatmin    - (optional) location of minimal value
%     history   - (optional) array of iteration historyory, useful
for tables and plots. The three columns are iteration, fcn evals,
and min value found.
```

SAMPLE FOR CALLING direct.m

```
% 1. Establish bounds for variables
bounds = [-2 2;-2 2];

% 2. Send options to Direct
options.showits   = 1;
options.tol       = 0.01;

% 2a. NEW!
% Pass Function as part of a Matlab Structure
Problem.f = 'gp';
Problem.numconstraints = 1;
Problem.constraint(1).func    = 'circlecon'; %constraint function
Problem.constraint(1).penalty = 1;           %penalty value you choose
%                                                    %
% Problem.constraint(2).func = 'anotherconstraint';  %
% Problem.constraint(2).penalty = 1;                 %

% 3. Call DIRECT
[fmin,xmin,hist] = Direct(Problem,bounds,options);

% 4. Plot iteration statistics
plot(hist(:,2),hist(:,3))
xlabel('Fcn Evals');
ylabel('f_{min}');
title('Iteration Statistics for GP test Function');
```

EXERCISE: DIRECT

- What is the Lipschtiz constant? What value would be appropriate for the function x^2+x in the interval (-2,2)?
- Invent function values at every point where the function is evaluated in Slide 204 that are consistent with the diagram.
- What are the meanings of the term exploration and exploitation in the context of global optimization?

GENETIC ALGORITHMS

- Genetic algorithms imitate a natural optimization process: natural selection in evolution.
- Developed by John Holland at the University of Michigan for machine learning in 1975.
- Similar algorithms developed in Europe in the 1970s under the name evolutionary strategies
- Main difference has been in the nature of the variables: Discrete vs. continuous
- Class is called evolutionary algorithms
- Key components are population, parent and child designs, and randomness (e.g. mutation)

BASIC SCHEME

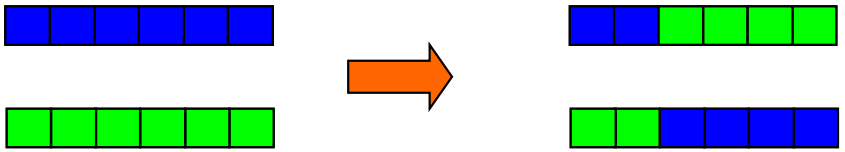
- Coding: replace design variables with a continuous string of digits or “genes”
  - Binary
  - Integer
  - Real
- Population: Create population of design points
- Selection: Select parents based on fitness
- Crossover: Create child designs
- Mutation: Mutate child designs

Outline of algorithm

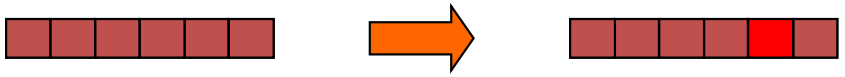
- Create random initial population
- Generate next population based on the following steps
  - Scores each member of the current population based on its fitness
  - Selects members, called parents, based on their fitness
  - Choose lower/higher fitness members as elite. These elite members are passed to the next population
  - Produces children from the parents using genetic operations
  - Replaces the current population with the children to form the next generation
- Repeat the process until stopping criteria are satisfied

GENETIC OPERATORS

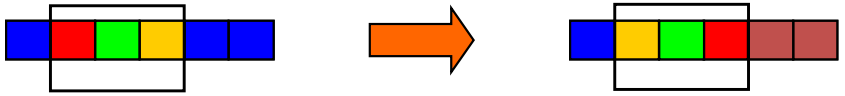
- Crossover: portions of strings of the two parents are exchanged



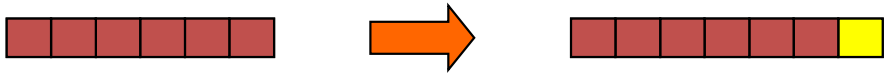
- Mutation: the value of one bit (gene) is changed at random



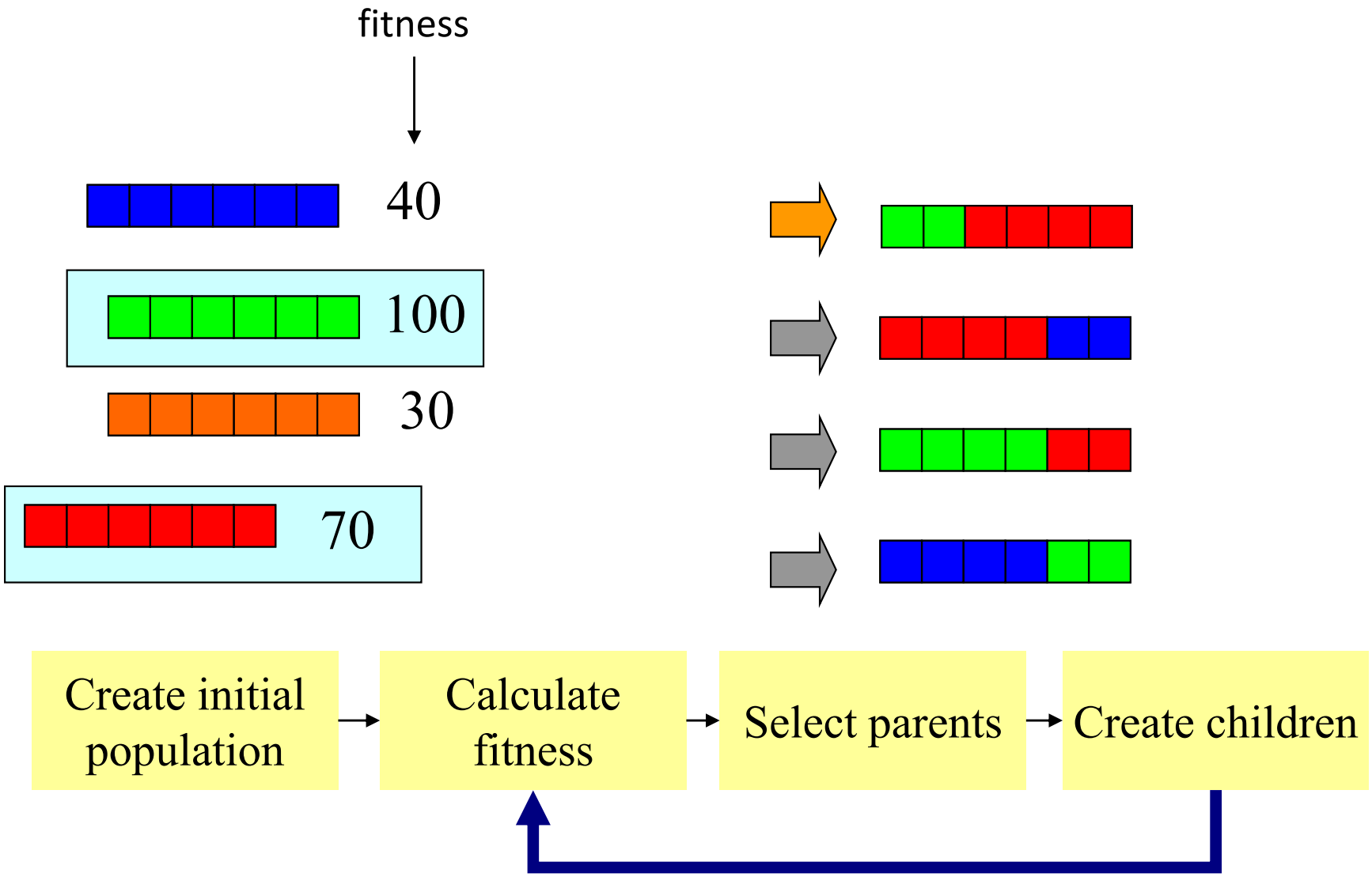
- Permutation: the order of a portion of the chromosome is reversed



- Addition/deletion: one gene is added to/removed from the chromosome



ALGORITHM OF STANDARD GA



CODING

- Integer variables are easily coded as they are or converted to binary digits

$$\text{Minimize} \quad f(\mathbf{b}), \quad \mathbf{b} = \{ b_1 \quad b_2 \quad b_3 \quad b_4 \}$$

$$\underbrace{0 \quad 1 \quad 1 \quad 0}_{b_1} \quad \underbrace{1 \quad 0 \quad 1}_{b_2} \quad \underbrace{1 \quad 1}_{b_3} \quad \underbrace{1 \quad 0 \quad 1 \quad 1}_{b_4}$$

Represent integer using bits

$b_1 = 6, b_2 = 5, b_3 = 3, b_4 = 11$

$\{0 \le b_1, b_4 \le 15\}, \ \{0 \le b_2 \le 7\}, \ \{0 \le b_3 \le 3\}$

CODING *cont.*

- Real variables require more care
- Key question is resolution or interval
- Range $\{b_i^L \leq b_i \leq b_i^U\}$ with a resolution b^{incr}
- The number m of required digits found from

$$2^m \geq \frac{b_i^U - b_i^L}{b^{\text{incr}}} + 1$$

- Ex) $\{0.01 \leq b \leq 1.81\}$ with $b^{\text{incr}} = 0.001$

$$2^m \geq \frac{b_i^U - b_i^L}{b^{\text{incr}}} + 1 = 1801$$

- $m = 11$, $b^{\text{incr}} = 0.00088$

EXAMPLE: STACKING SEQUENCE OPTIMIZATION

- For many practical problems, angles limited to 0-deg, ±45-deg, 90-deg.
- Ply thickness given by manufacturer
- Stacking sequence optimization a combinatorial problem
- Genetic algorithms effective and easy to implement, but do not deal well with constraints

CODING - STACKING SEQUENCE

- Binary coding common. Natural coding works better.
  ($0^0 \rightarrow 1$, $45^0 \rightarrow 2$, $-45^0 \rightarrow 3$, $90^0 \rightarrow 4$)
  $(45/-45/90/0)_s$ => (2/3/4/1)
- To satisfy balance condition, convenient to work with two-ply stacks
  ($0_2 \rightarrow 1$, $\pm 45 \rightarrow 2$, $90_2 \rightarrow 3$) or
  $(45/-45/90_2/0_2)_s$ => (2/3/1)
- To allow variable thickness add empty stacks
  (2/3/1/E/E)=> $(45/-45/90_2/0_2)_s$

CODING - DIMENSIONS

- Binary coding most common. Real number coding possible but requires special treatment.
- Genetic algorithm not effective for getting high precision. It is better to go for coarse grid of real values. With n binary digits get 2^n values.
- Segregate stacking sequence and geometry chromosomes.

INITIAL POPULATION

- Random number generator used
- Typical function call is rand(seed)
- In Matlab rand(*n*) generates *n*x*n* matrix of uniformly distributed (0,1) random numbers
- Seed updated after call to avoid repeating the same number. See Matlab help on how to change seed (state).
- If we want repeatable runs must control seed.
- Need to transform random numbers to values of possible gene values.

FITNESS WITH CONSTRAINT VIOLATION PENALTY

- Augmented objective

$$f^* = f + pv - bm + \text{sign}(v)\Delta$$

  - v = max violation
  - m = min margin

- As we get near the optimum, difference in f^* between individuals gets smaller.
- To keep evolutionary pressure, fitness is
  - normalized objective

$$fit_i = \frac{\left| f_i^* - f_{\max}^* \right|}{\left| f_{\min}^* - f_{\max}^* \right|}$$

SELECTION

- Roulette wheel selection
- Tournament selection
  - Randomly select two individuals and pick the best one as one parent
- Elitist strategies
  - Select the best fitness individual of the current population
- Selection pressures versus exploration
- No twin rule

ROULETTE WHEEL

- Slice of the roulette wheel proportional to its fitness
  - Example fitness {0.62, 0.60, 0.65, 0.61, 0.57, 0.64}
  - Example reverse rank {4/21, 2/21, 6/21, 3/21, 1/21, 5/21}

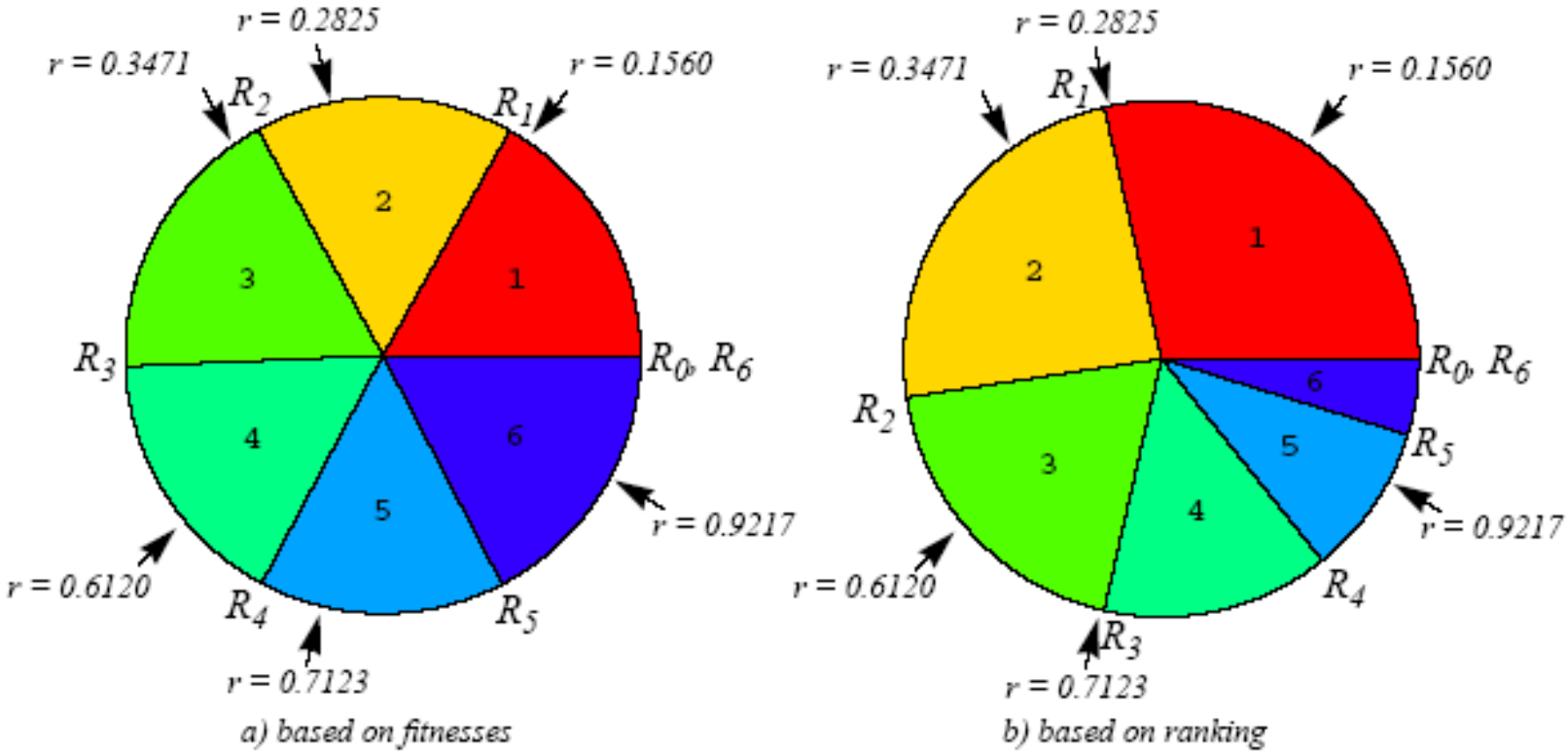


Figure 5.7: Roulette wheel for example 5.4.2

SINGLE POINT CROSSOVER

- Parent designs $[0_4/\pm45_2/90_2]_s$ and $[\pm45_4/0_2]_s$

- Parent 1 [1/1/2/2/3]
- Parent 2 [2/2/2/2/1]
- One child [1/1/2/2/1]
- That is: $[0_4/\pm45_2/0_2]_s$

OTHER KINDS OF CROSSOVER

- Multiple point crossover
- Uniform crossover
- Bell-curve crossover for real numbers
- Multi-parent crossover

MUTATION AND STACK SWAP

- [1/1/2/2/3]=> [1/1/2/3/3]
- $[0_4/\pm45_2/90_2]_s$ => $[0_4/\pm45/90_4]_s$

- [1/1/2/2/3]=> [1/2/1/2/3]
- $[0_4/\pm45_2/90_2]_s$ => $[(0_2/\pm45)_2/90_2]_s$

Stopping Criteria

- Stop when the maximum number of generations reaches
- Stop after running for the maximum computational time reaches
- Stop when the best fitness becomes less than or equal to the limit
- Stop when the relative change in the fitness is less than the tolerance
- Stop if there is no improvement during an interval of time
- Stop if the average relative change in the fitness is less than the tolerance.

Genetic algorithm in Matlab

- [x,fval,exitflag,output,population,scores] = ga(fun,nvars,A,b,[],[],lb,ub,nonlcon,IntCon,options)
- Outputs
  - x: the best location
  - fval: objective function value at x
  - exitflag: >0 normal stop, <0 erroneous stop
  - output: opt process information
  - population: final population
  - scores: fitness function values of the population

Genetic algorithm in Matlab cont.

- [x,fval,exitflag,output,population,scores] = ga(fun,nvars,A,b,Aeq,beq,lb,ub,nonlcon,IntCon,options)
- Inputs
  - fun: objective function
  - nvars: the number of design variables
  - A, b: inequality constraints $Ax \leq b$
  - Aeq, beq: equality constraints $A_{eq}x = b_{eq}$
  - lb, ub: lower- and upper-bounds of design variables
  - nonlcon: nonlinear constraint function
  - IntCon: integer variable
  - options: optimization options

EXERCISE: GENETIC ALGORITHM

- Global optimization balances exploration and exploitation. How is that reflected in genetic algorithms?
- What are all possible balanced and symmetric child designs of $[0_2/\pm45/90]_s$ and $[\pm45_2/0]_s$ with uniform cross-over?
- When we breed plants and animals we do not introduce randomness on purpose into the selection procedure. Why do we do that with GAs?.

RELIABILITY

- Genetic algorithm is random search with random outcome.
- Reliability can be estimated from multiple runs for similar problems with known solution
- Variance of reliability, *r*, from *n* runs

$$\sigma_r = \sqrt{\frac{r(1-r)}{n}}$$