Searching With No Flashlight

An overview of derivative-free optimization

Max Yi Ren, Arizona State University, 2015

What is a Derivative-Free Algorithm?

Derivative-free (non-gradient) algorithm:

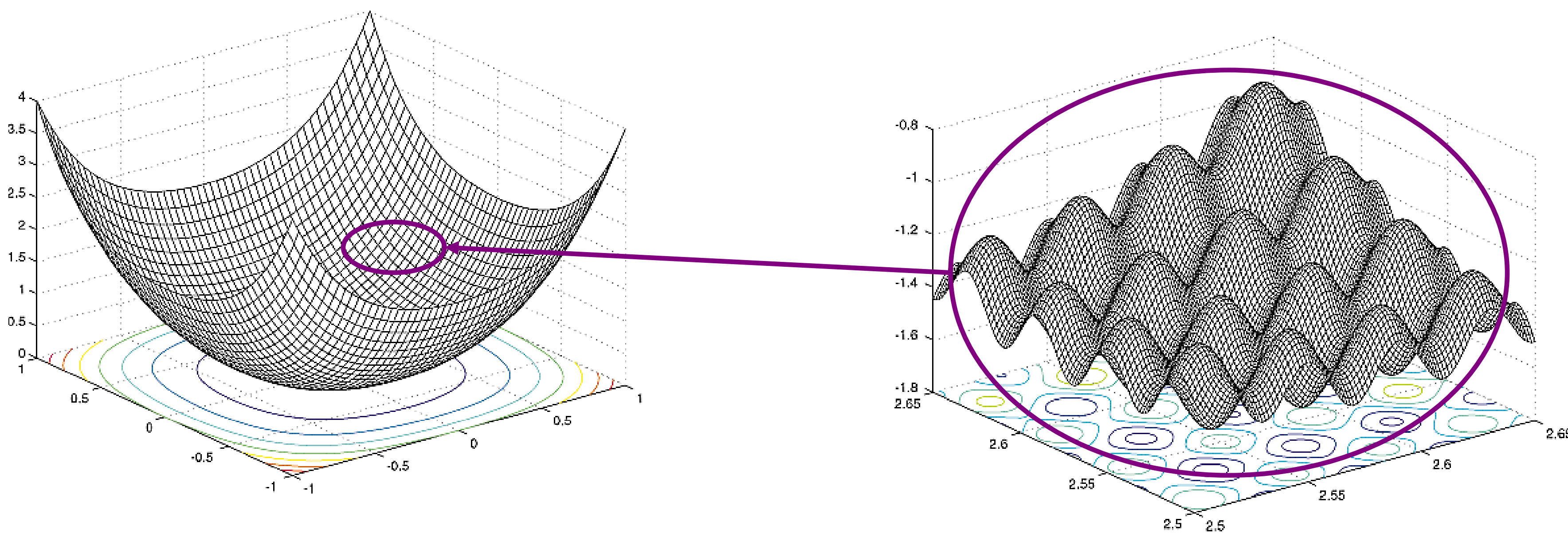
- No gradient information necessary
- "Smart" method of searching design space based upon some heuristics

Outline:

- Why use derivative-free algorithms? And why not?
- Review of existing algorithms

Why Derivative-Free Algorithms? (1)

- **Expensive function evaluation**
- **Noisy function evaluation**

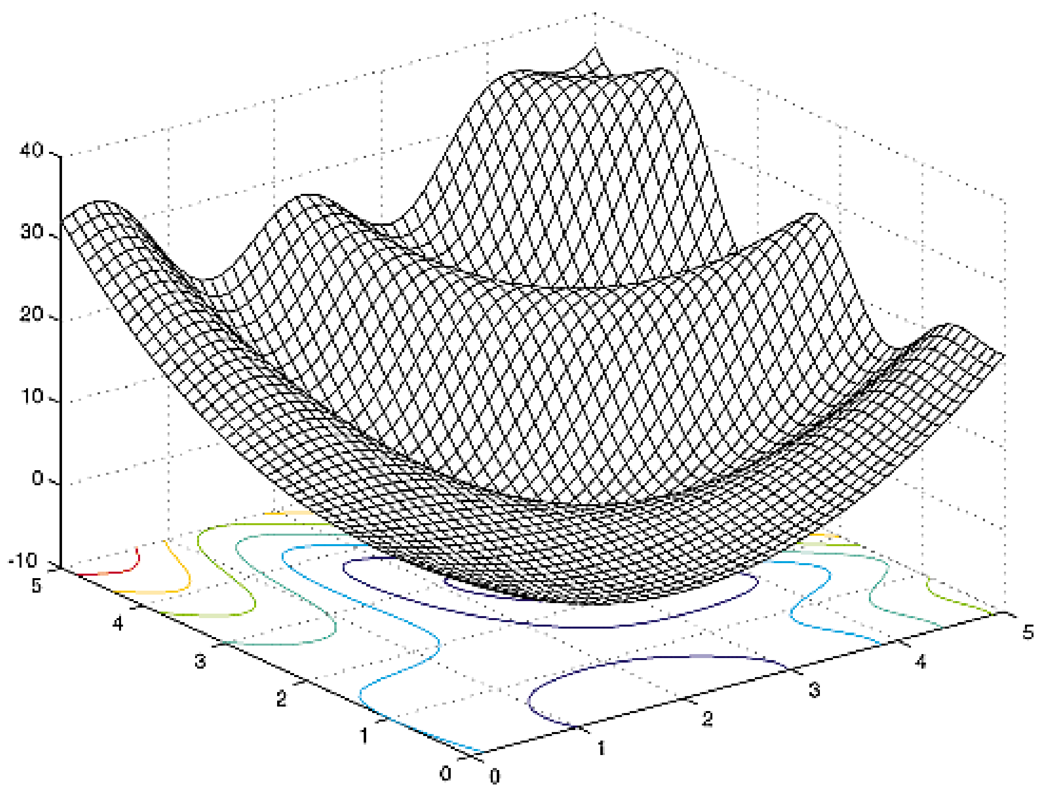


unimodal function

numerical noise

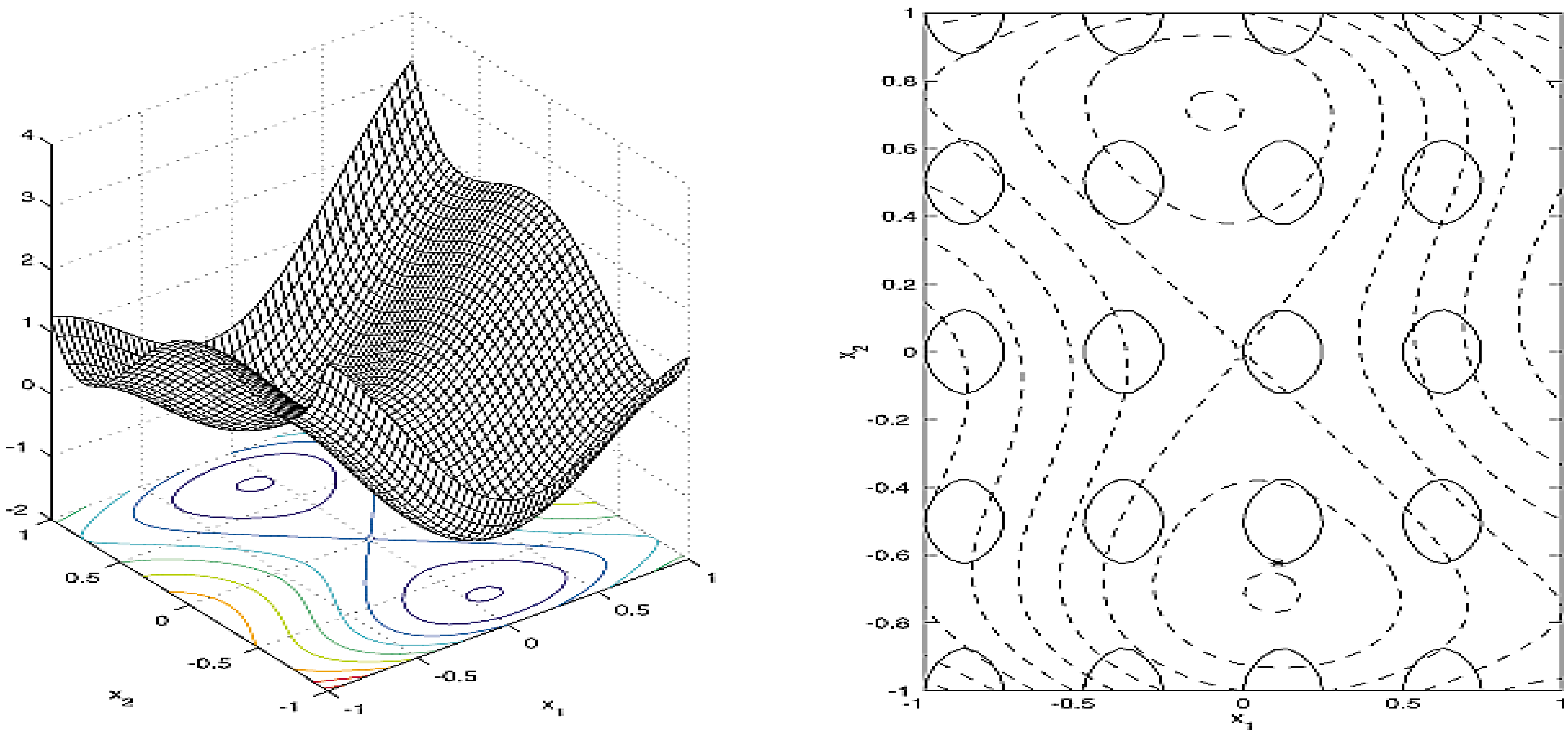
Why Derivative-Free Algorithms? (2)

- **Multiple optima exist**



Why Derivative-Free Algorithms? (3)

- **Disconnected feasible regions**
- **Difficulty in finding feasible points**



disconnected feasible region

Why Derivative-Free Algorithms? (4)

- **Discrete choice variables / combinatorial problems**
  - Material selection
  - Component selection
  - Routing problems
- **Integer Variables**

Why NOT Derivative-Free Algorithms?

Disadvantages

- Slow to converge
- Usually no guarantee of optimality
- Often require tuning of many algorithm parameters
- Constraint handling often through penalty functions
  - No guarantee of feasibility
  - Equality constraints are more difficult

Classes of Derivative-Free Algorithms

Stochastic

Search depends on probability/random number generation; Each run of algorithm will take different search path and may find different “best point”

Deterministic

Search follows distinct path (dependent on starting point, if specified); Each run of algorithm will have same result

Existing Derivative-Free Algorithms

Stochastic methods

- Simulated annealing
- Genetic algorithms
- Particle swarm

Deterministic methods

- DIRECT
- Multilevel coordinate search (MCS)
- Efficient global optimization (EGO)
- NOMAD (hybrid method)

and MANY others…

Survey of Derivative-Free Algorithms

Exhaustive survey by Rios and Sahinidic:

- 22 algorithms considered;
- On over 500 problems (convex/nonconvex + smooth/nonsmooth) with bounds only;
- With #variable from 1 to 30;
- Limit of 2500 iterations and 600 CPU seconds.

Conclusions

- There always exist a few problems that a certain solver has the best solution quality.

http://egon.cheme.cmu.edu/ewocp/docs/SahinidisEWO_DFO2010.pdf

Topics

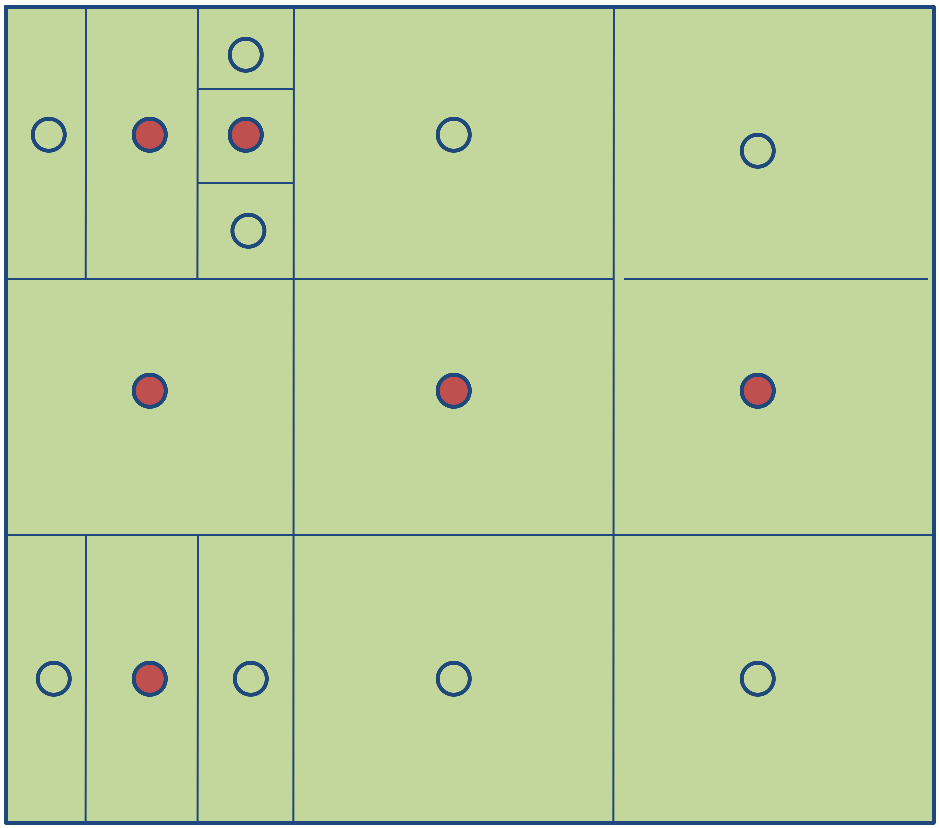
- **DIRECT**
- **Simulated annealing**
  - **Quantum annealing**
- **Genetic algorithm**
  - **CMA-ES**
- **Efficient global optimization (EGO)**
- **Pattern Search**
  - **NORMAD**

DIRECT Overview

DIRECT stands for “Divided Rectangles”

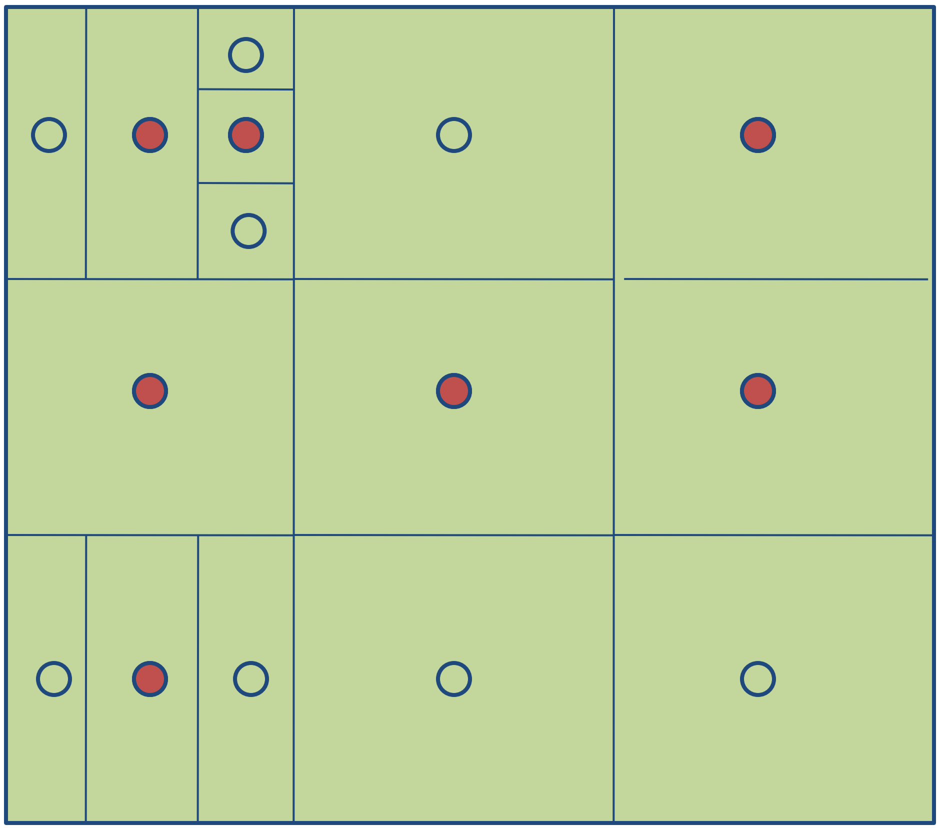
- Whole design space is sub-divided into rectangles;
- The “best” and “largest” rectangles are further divided.

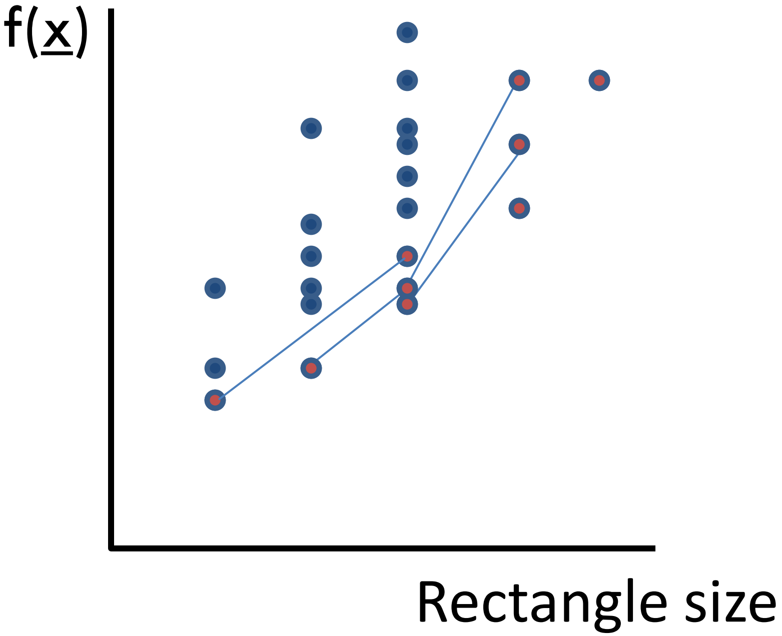
DIRECT with 2 Variables



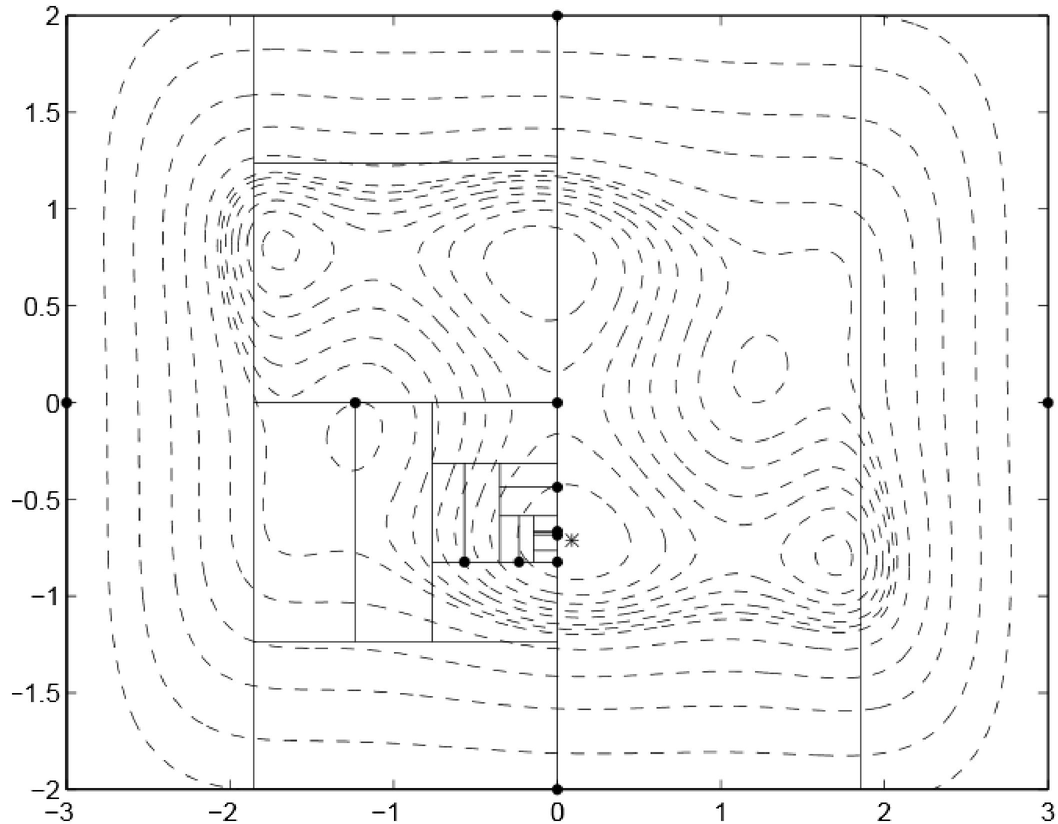
1. Sample center of design space
2. Select best candidate rectangles and divide into thirds along their longest dimensions
3. Best candidate rectangles based upon:
   - best f(x)
   - lowest constraint violation
   - size of rectangle
4. Iterate until max. number of function calls

DIRECT with 2 Variables





Multilevel Coordinate Search (MCS)



Extension of DIRECT to have “basepoints” not in the center of boxes

DIRECT Pros/Cons

- **Advantages**
  - Global and local search balance
  - Deterministic, has the ability to be restarted where it left off
  - No parameters to tune
  - Can handle integer variables
- **Disadvantages**
  - Dimensionality: For problems of 10 variables or larger, DIRECT has difficulties because of having to divide along each dimension
  - Slow local convergence
  - Cannot handle equality constraints

Simulated Annealing Overview

Idea: Simulate the cooling a metal to find the “strongest” configuration of atoms

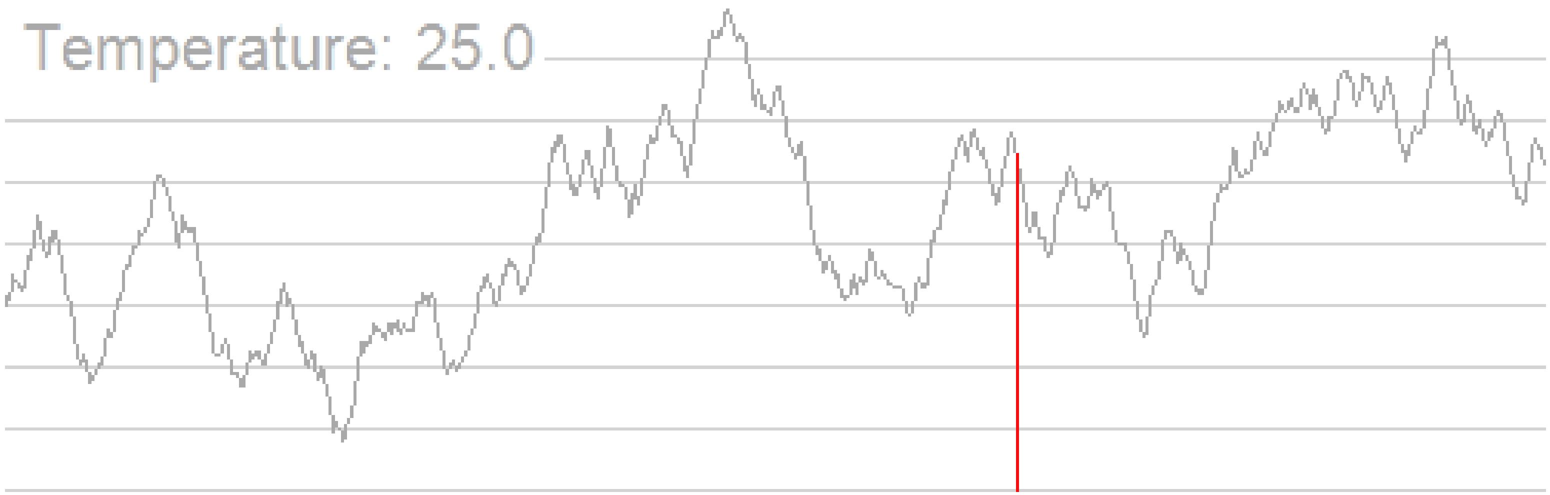
Simulated Annealing - Algorithm

- Cooling of metals: want to find lowest energy state
- Performs random search with some probability of accepting a worse point (to get out of local minima)

$$\text{Prob}(\mathbf{x} \leftarrow \mathbf{y}) = \begin{cases} 1 & \text{if } \Delta f < 0 \text{ (better: downhill)} \\ \exp(-\frac{\Delta f}{t}) & \text{if } \Delta f \geq 0 \text{ (worse: uphill)} \end{cases}$$

- *t* is the temperature at the current iteration. *t* decreases along the iteration number.

Simulated Annealing - Demo



Simulated Annealing - Constraints

Penalty function:

$$\min\ f_P(\bar{x}, Penalty) = f(\bar{x}) + \sum_{i=1}^{m} w_i \cdot \left(\max(0, g_i(\bar{x}))\right)^2$$

- Most common is quadratic penalty function, though others are possible
- No guarantee of feasibility
- For equality constraints, can use two inequalities for upper and lower bounds
- Scaling of constraints and objective is ESSENTIAL to ensure feasibility with reasonable descent

Simulated Annealing - Convergence

Proved by Geman brothers:

If the temperature drops by the following form:

$$T(t) = \frac{cN}{\log t},$$

where *c* is a problem dependent constant, *N* is the problem size (number of variables), then SA is guaranteed to find the global solution in infinite time limit.

Simulated Annealing – Pros/Cons

Advantages:

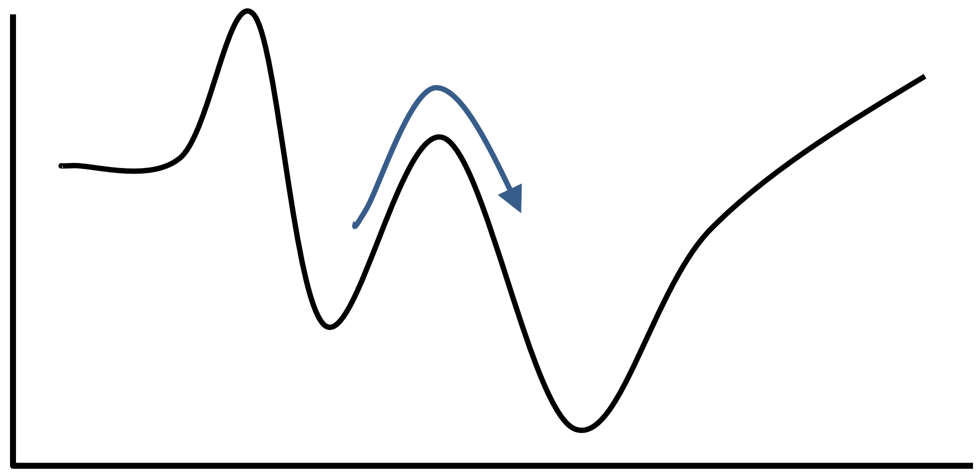
- Doesn't need to systematically cover space—better efficiency for large-dimension problems

Disadvantages:

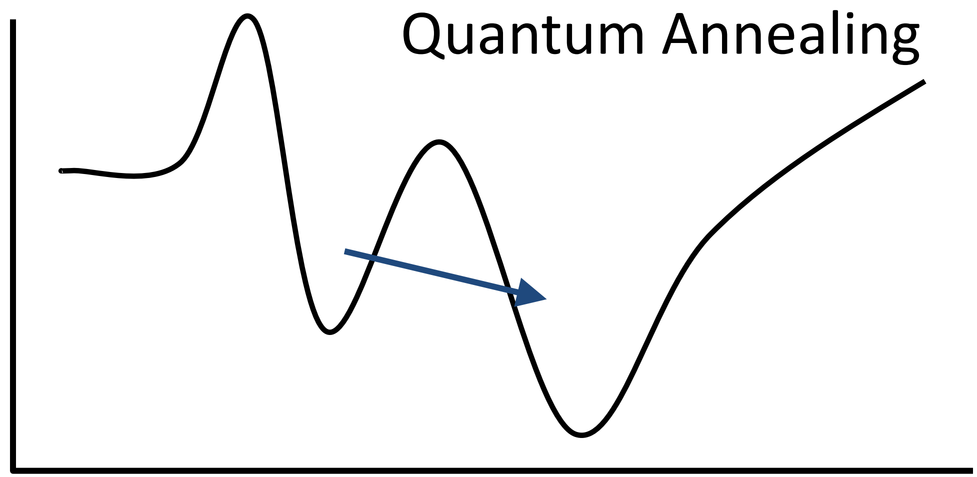
- Doesn't always cover the design space (quasi-global)
- Dependent on starting point
- Random directional search not very "smart"
  - Can repeat areas already searched
  - Can require large # of function calls
- Many parameters to tune – algorithm performance is dependent on these parameters
  - Penalty weights
  - Temperature cooling schedule

From Simulated Annealing to Quantum Annealing

Simulated Annealing – Search by **thermal** fluctuation

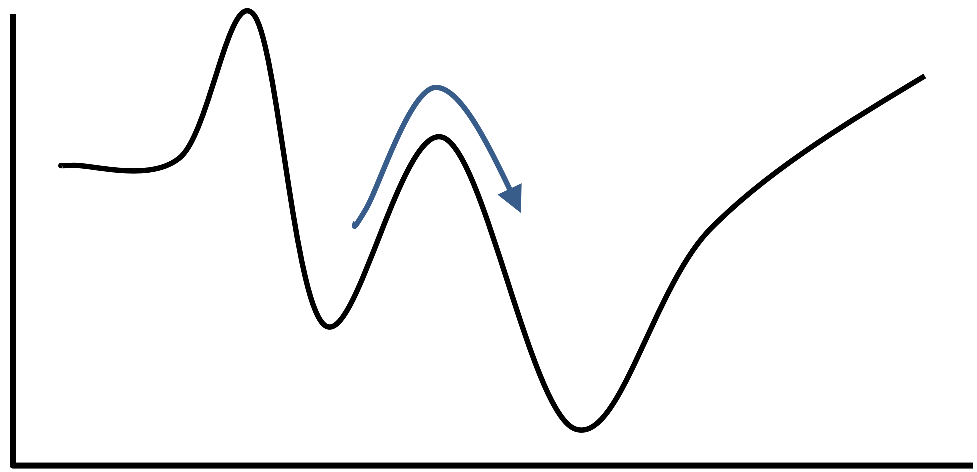


Quantum Annealing (Adiabatic Quantum Optimization) – Search by **quantum** fluctuation

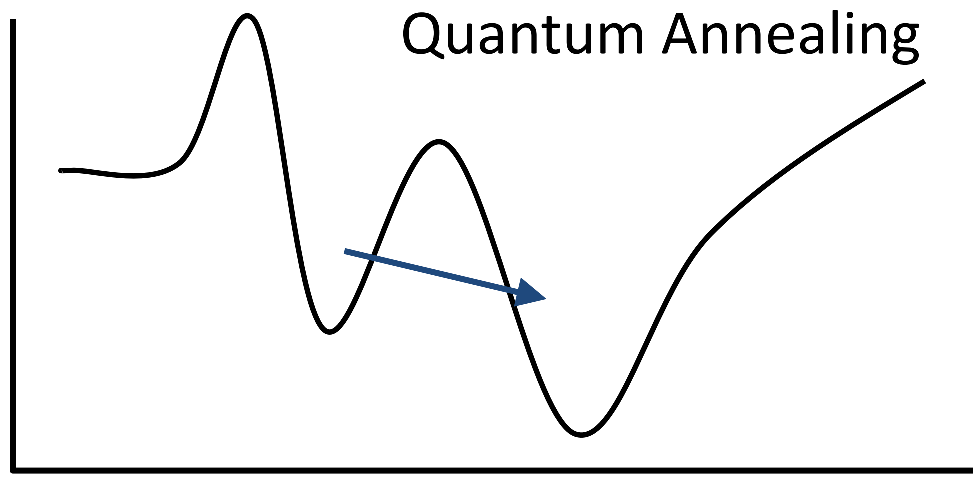


From Simulated Annealing to Quantum Annealing

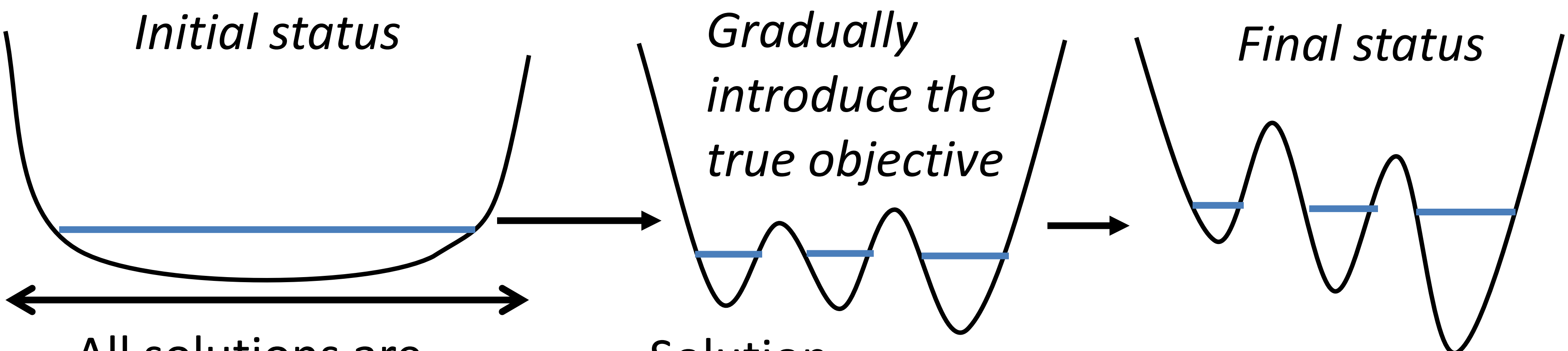
Simulated Annealing – probability to escape depends on the temperature and the barrier height (energy difference)



Quantum Annealing – probability to escape depends on the **quantum tunneling width** and the barrier width



Quantum Annealing - Idea



All solutions are equally likely

Solution probabilities change

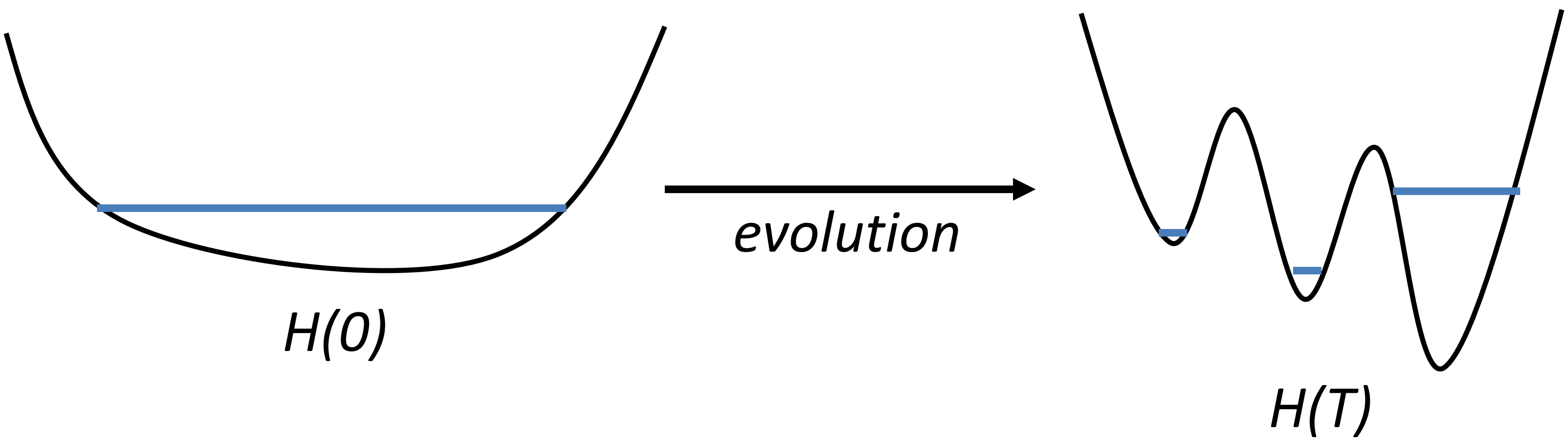
Probability mass concentrates at the true global optimum

The Adiabatic Theorem

Slowly varying Hamiltonian (objective) for evolution from *t = 0* to *t = T*

Provided *T* is "large enough", a quantum system starting in the ground state of *H(0)* evolves to the ground state of *H(T)*

Large enough *T*: $T = O(g^{-2})$, where *g* is the minimum gap between ground state and first excited state of *H(t)*



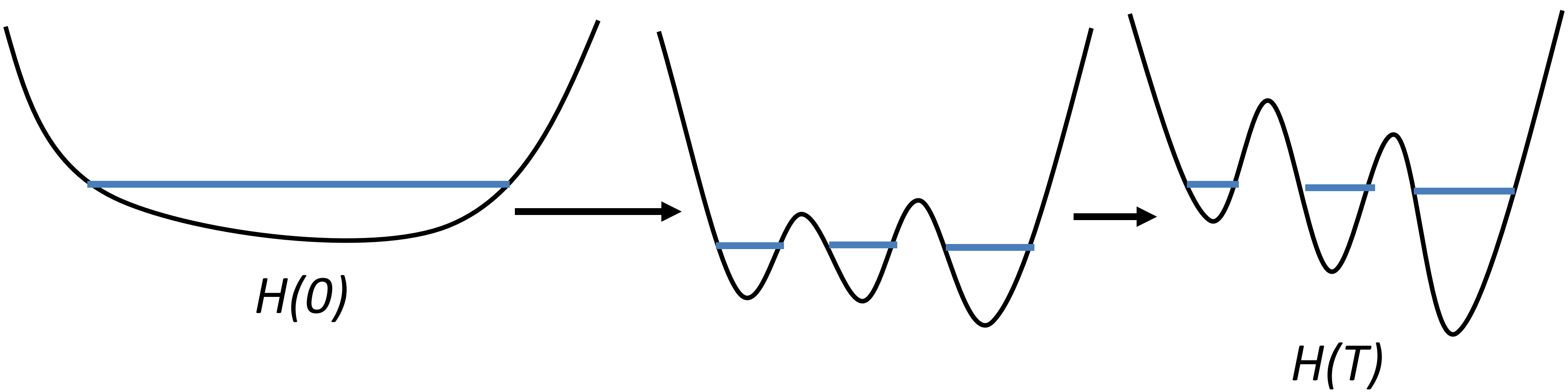
Max Yi Ren, Arizona State University, 2015

The Adiabatic Theorem

Slowly varying Hamiltonian (objective) for evolution from *t = 0* to *t = T*

Provided *T* is "large enough", a quantum system starting in the ground state of *H(0)* evolves to the ground state of *H(T)*

Large enough *T*: $T = O(g^{-2})$, where *g* is the minimum gap between ground state and first excited state of *H(t)*



Max Yi Ren, Arizona State University, 2015

Quantum Annealing – Convergence

SA convergence:

$$t = \exp(\frac{cN}{\delta})$$

QA convergence (faster than SA for small δ):

$$t = \exp(\frac{N \ln \delta}{2c'})$$

Quantum Annealing – Implementation

D-wave QA solves the Ising function (Quadratic Unconstrained Binary Optimization)

$$E(x_1,\ldots,x_N)=\sum_{i=1}^{N}h_i x_i+\sum_{i<j=1}^{N}J_{ij}x_i x_j$$

Can be applied to many machine learning problems (why?)

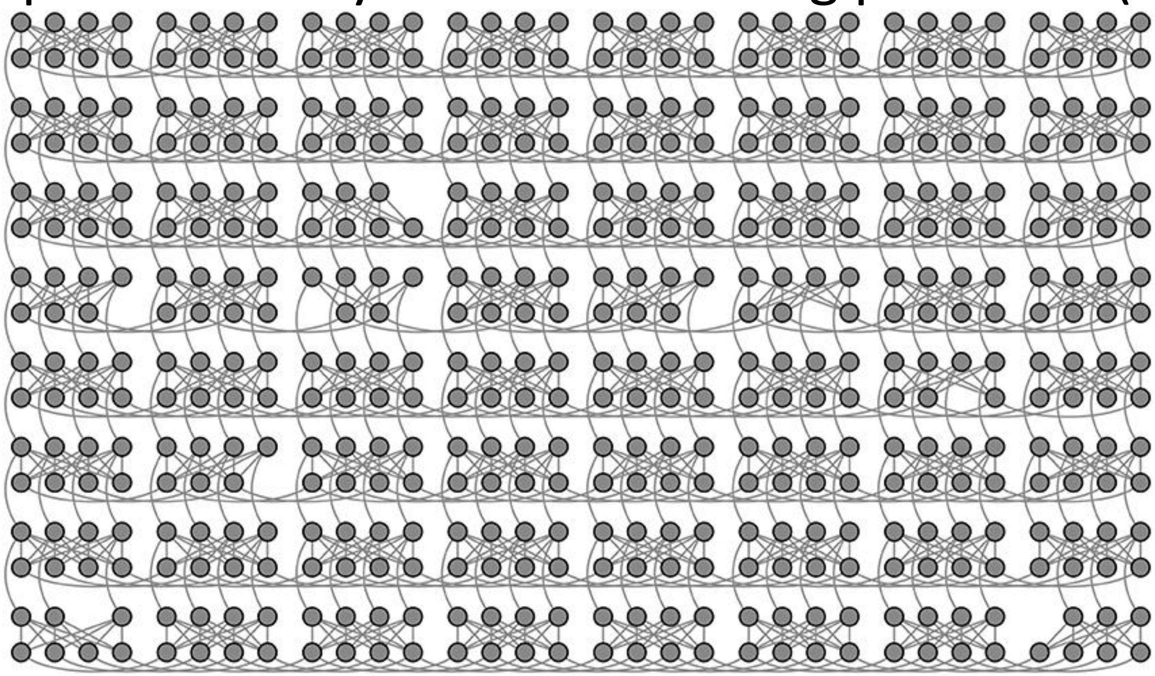
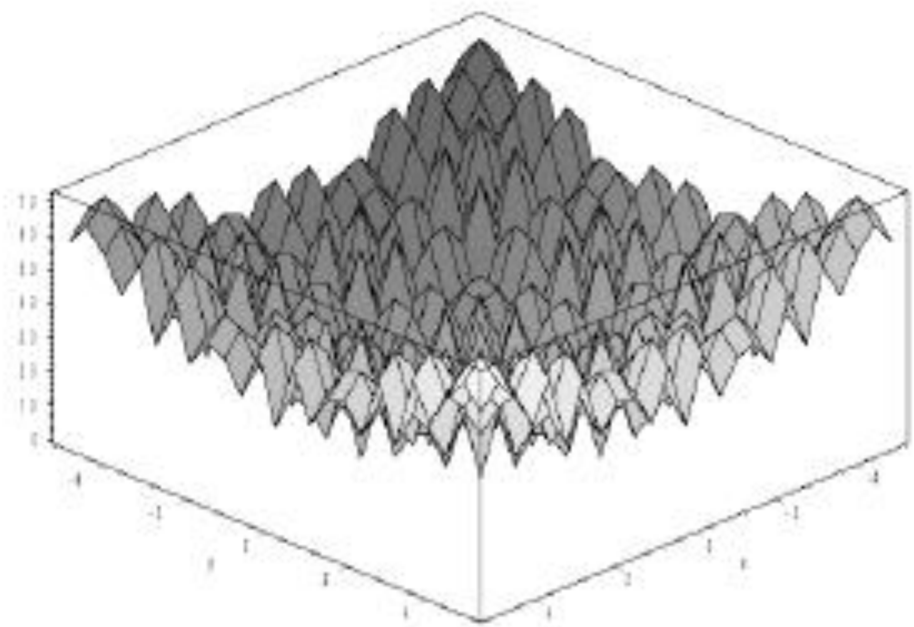


Figure from: http://www.frontiersin.org/files/Articles/107579/fphy-02-00056-HTML/image_m/fphy-02-00056-g001.jpg

Quantum Annealing – Summary

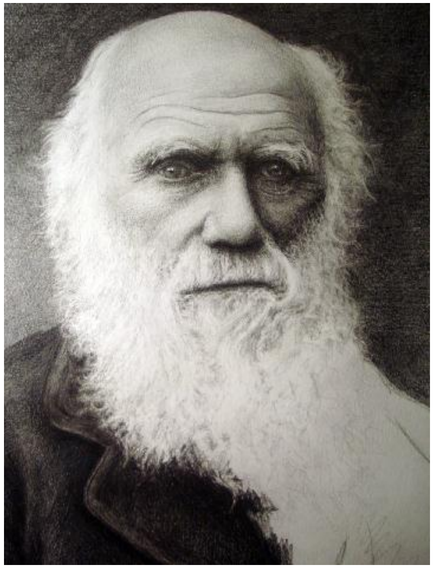
QA has an advantage on problems “with **thin barriers** that separate very **deep chasms** between local minima”.



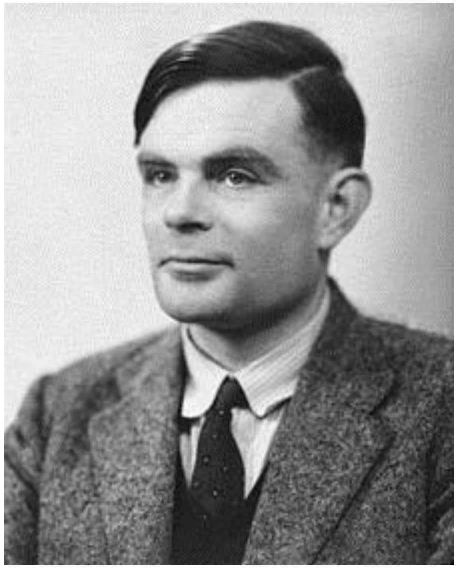
2D Rastrigin’s funciton

More info:
http://www.stat.phys.titech.ac.jp/~nishimori/QA/q-annealing_e.html
https://plus.google.com/+QuantumAILab/posts

Genetic Algorithm



Charles Darwin

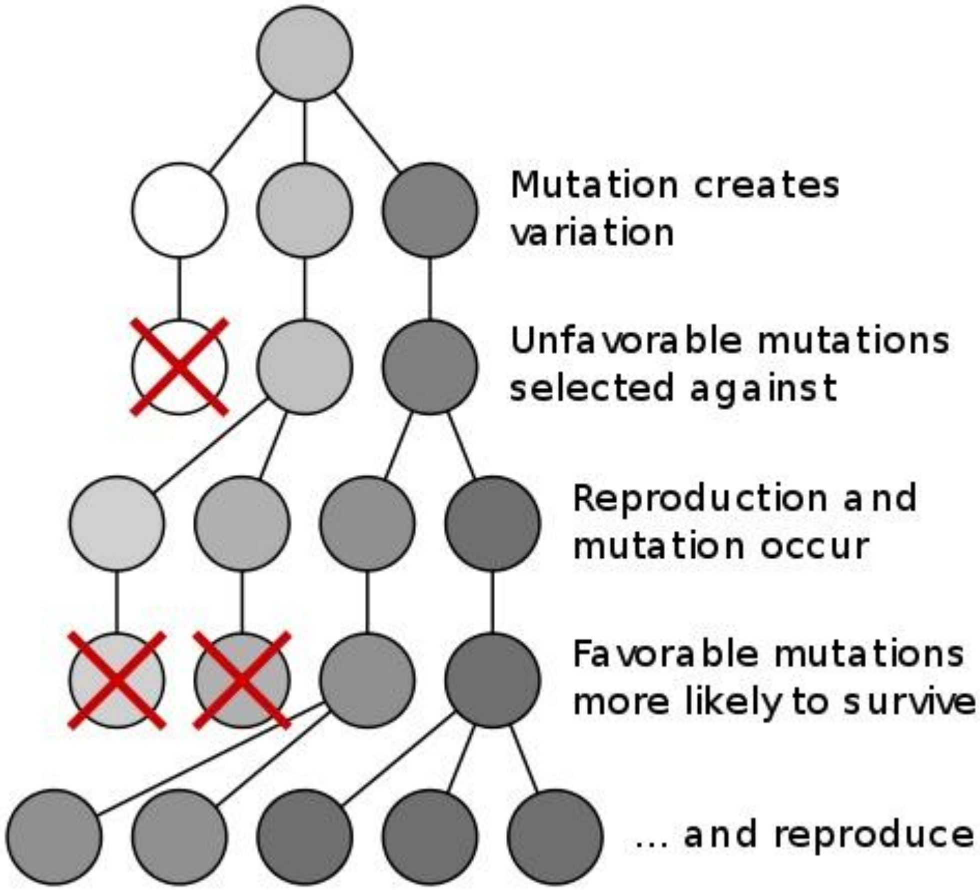


Alan Turing

Introduced by Turing, popularized in the 1980's

One of most popular nongradient methods

Genetic Algorithm - Idea



http://media.tumblr.com/036028e2fcf2db98eb94cb3dff0aca5f/tumblr_inline_mmzvcqpvQ71qz4rgp.jpg

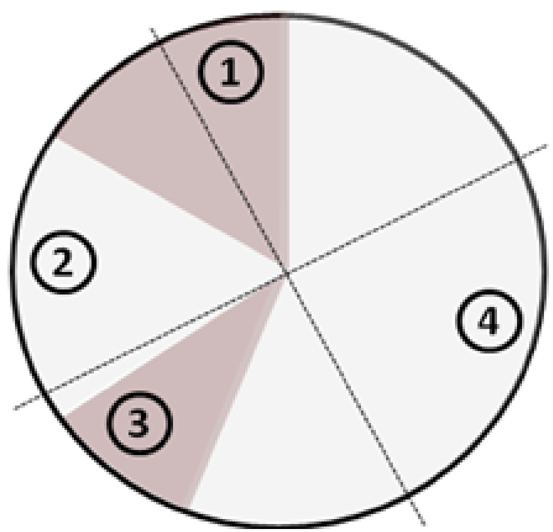
Idea: Only allow the "fittest" designs move their DNA to the next generation

Genetic Algorithm Overview

Starting with a population of random points in the feasible set, produce a new population of better points by *parent selection*, *crossover*, and *mutation*, until some conditions are satisfied.

GA - Parent Selection

- Many methods: roulette wheel, tournament, elitism, etc.
- Roulette wheel selection
  - Better individuals get larger portion of wheel
  - Random selection from wheel determines parents of next generation



GA - Parent Selection

- Many methods: roulette wheel, tournament, elitism, etc.
- Tournament selection
  - Randomly pick *k* chromosomes from the population
  - Pick the best one out of the subset
  - Iterate until all parents are picked

Each time pick three and compete

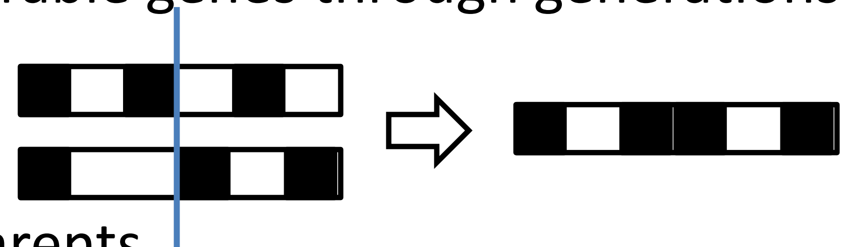
GA - Parent Selection

- Many methods: roulette wheel, tournament, elitism, etc.
- Elitism selection
  - Keep the best few chromosomes in the population
  - Can perform along with roulette wheel or tournament selection to prevent the solution from getting worse

GA - Crossover

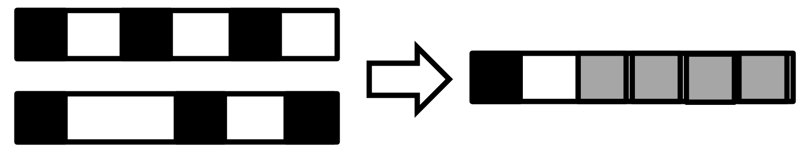
Crossover is used to propagate favorable genes through generations

- **Pure** (for binary chromosome):



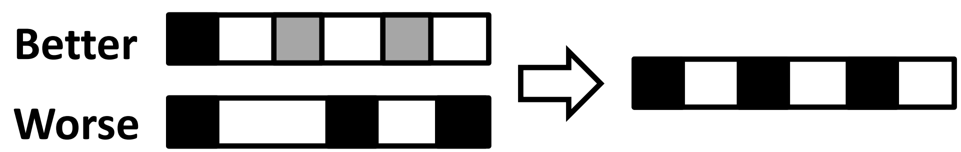
  Piecewise combination of two parents

- **Arithmetic** (for real chromosome):



  Creates linear interpolation of two parents

- **Heuristic**: Creates linear extrapolation of two parents in direction of better parent



The choice of crossover scheme is case dependent.

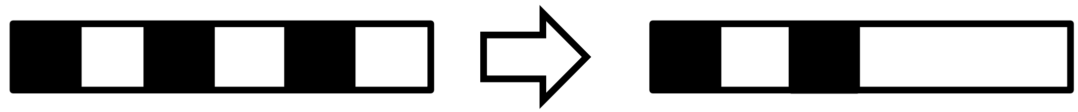
GA - Mutation

Many mutations are potentially good…

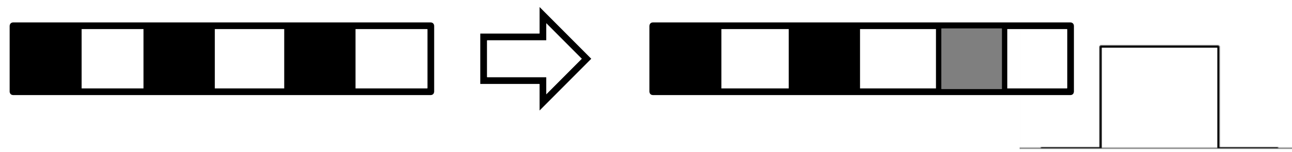
…but too much mutation can reduce fit designs

GA Mutation Methods

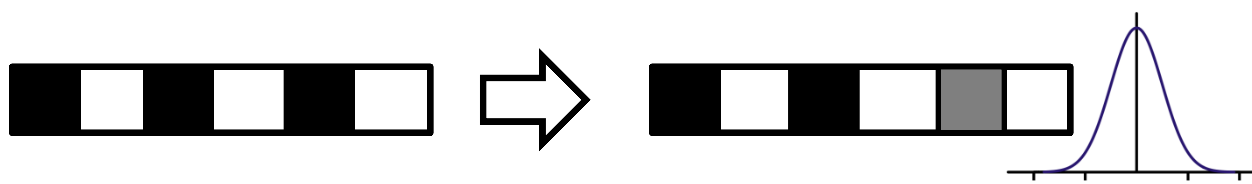
Boundary: Set one variable equal to its upper or lower bound



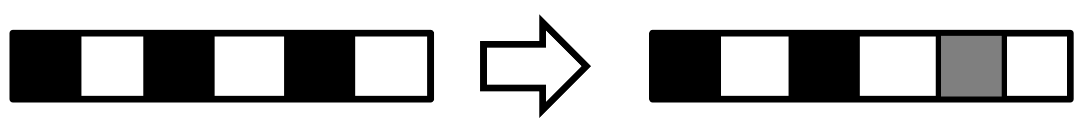
Uniform: Set one variable equal to a uniform random number



Non-uniform: Set one variable equal to a non-uniform random number



Incremental: Increments one variable a random amount



GA - Termination Criteria

Fixed number of generations

Run out of time

Highest ranking solution reached plateau over last K iterations

GA – Pros/Cons

- **Advantages**
  - Draws from a large body of designs: global search
  - Good performance on combinatorial problems
- **Disadvantages**
  - Difficulty balancing size of population/number of generations and overall time
  - Genetic operators may not create better designs
  - Not necessarily good at fine-tuning a design

Covariance Matrix Adaptation Evolution Strategy

CMA-ES **learns a covariance matrix** during the evolution, similar to approximating the inverse Hessian in classical methods.

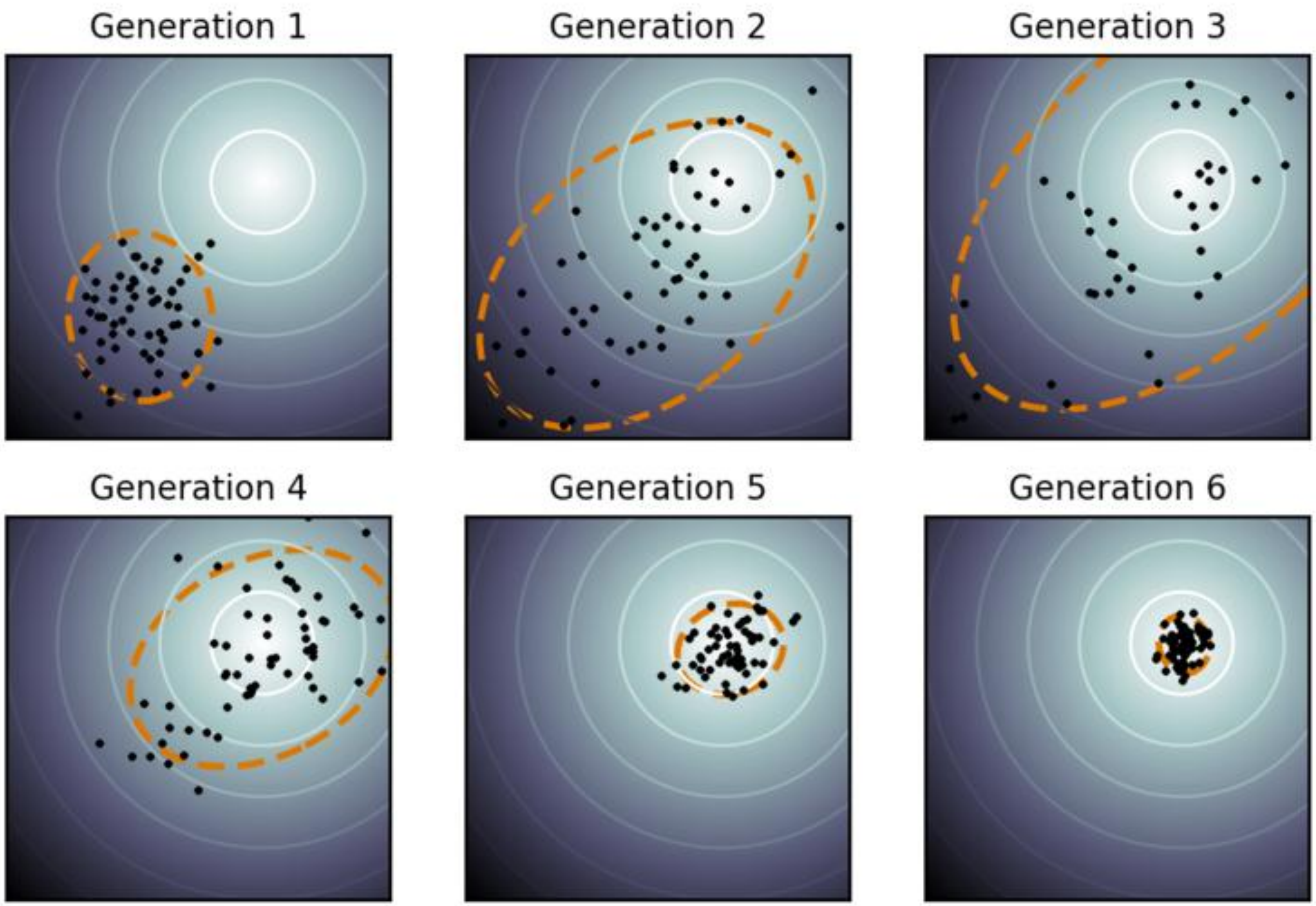


Figure from: http://en.wikipedia.org/wiki/File:Concept_of_directional_optimization_in_CMA-ES_algorithm.png

CMA-ES Outline

1. Initialize a distribution
2. While *not terminate*
    1. Order the population according to fitness
    2. Pick the top μ samples ("good samples")
    3. Update the mean of good samples
    4. Update the correlation and variance of the distribution
    5. Draw a new population

The updated mean and covariance maximizes the likelihood of "good samples".

CMA-ES Pros/Cons

Pros

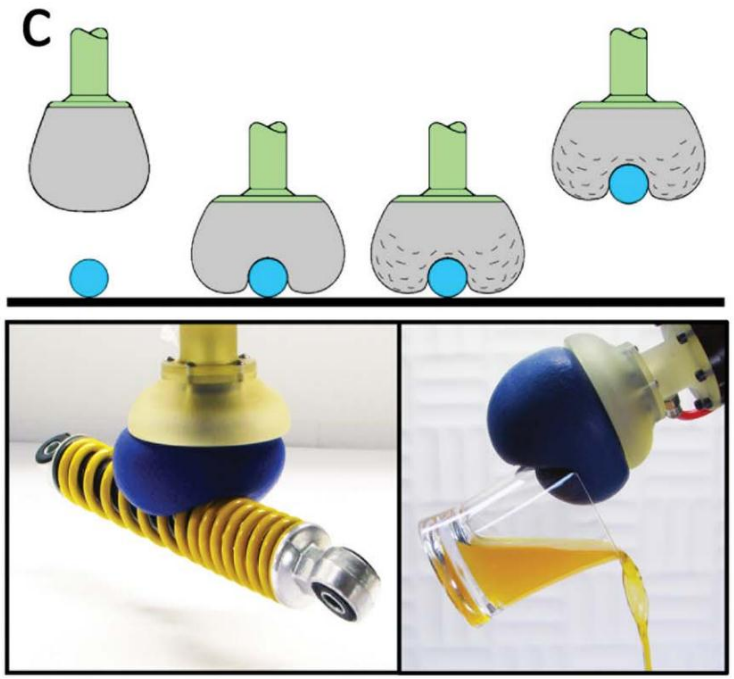
1. Learns a problem specific structure (e.g., better than GA, SA or DIRECT)
2. Suitable for non-convex, non-separable, ill-conditioned, multi-modal or noisy objective functions
3. Suitable for parallel computing
4. Suitable for problems that cannot be solved with a small number of function evaluations (<10*problem size)

Cons

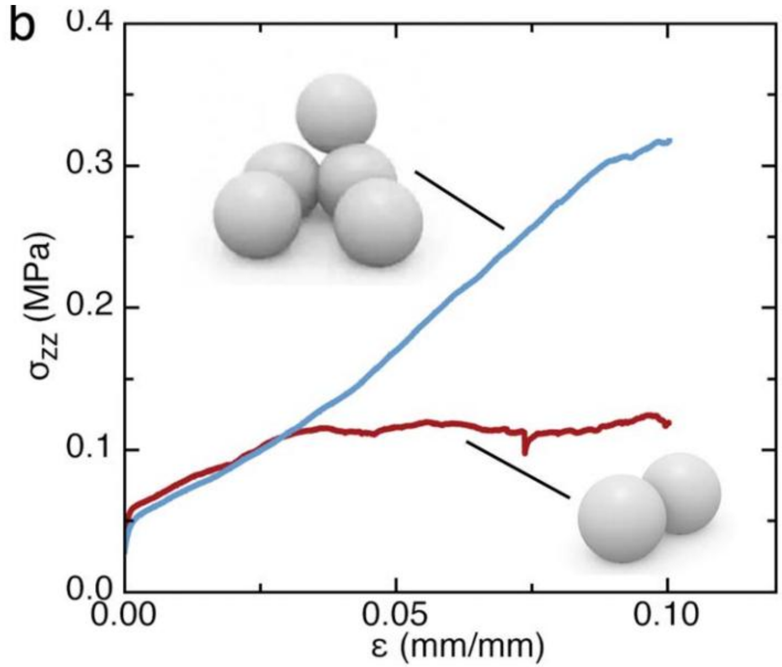
1. Learning does not involve fitness (only ranking)
2. Worse than response surface methods when problem size is small
3. Does not perform well on separable functions

CMA-ES Applications

Granular Material design



Jamming of granular material



Design material property using evolution strategy

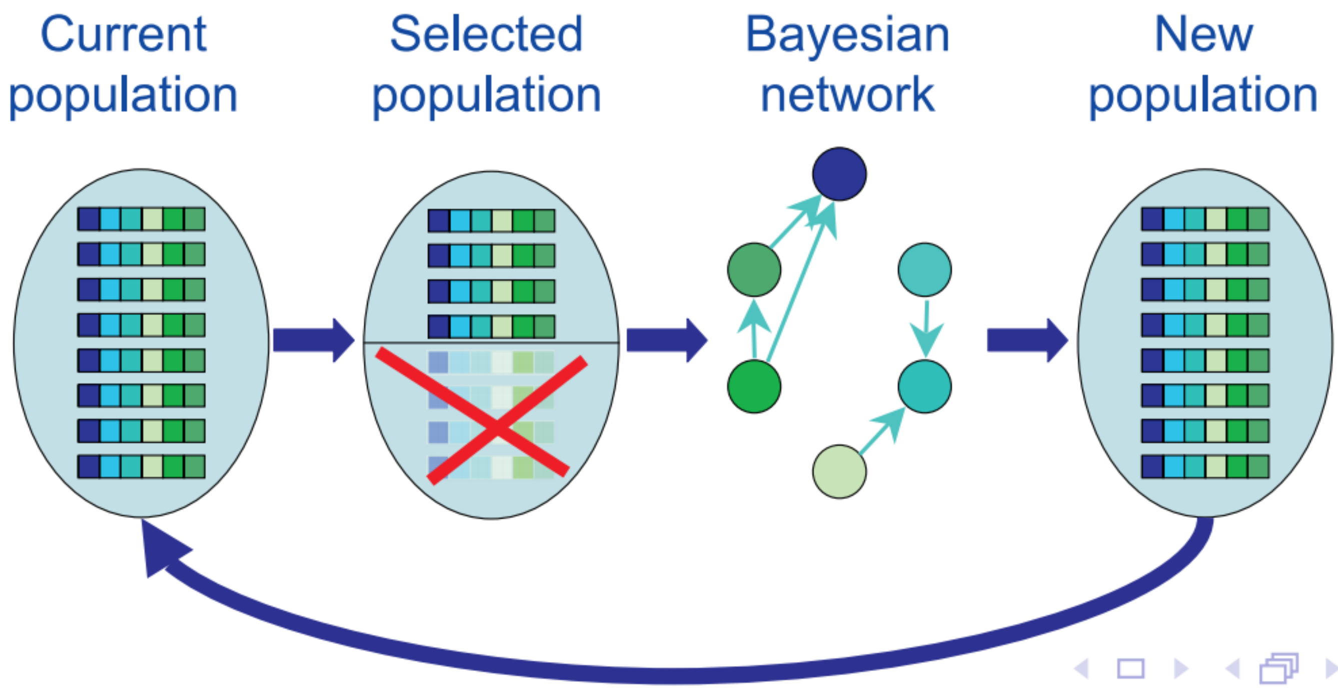
Figures from: http://jfi.uchicago.edu/~jaeger/group/JaegerGroupPapers/granular/Toward_jamming_by_design.pdf

The Bayesian Optimization Algorithm

- The idea of Genetic Algorithm is to mix promising “building blocks” to achieve good solutions.
- Traditional GA operations are shown to be inefficient in preserving partial solutions.
- More sophisticated operations were introduced to address this problem.

The Bayesian Optimization Algorithm

- BOA learns promising solutions (parents) using a Bayesian network and produces children that have similar properties as parents.



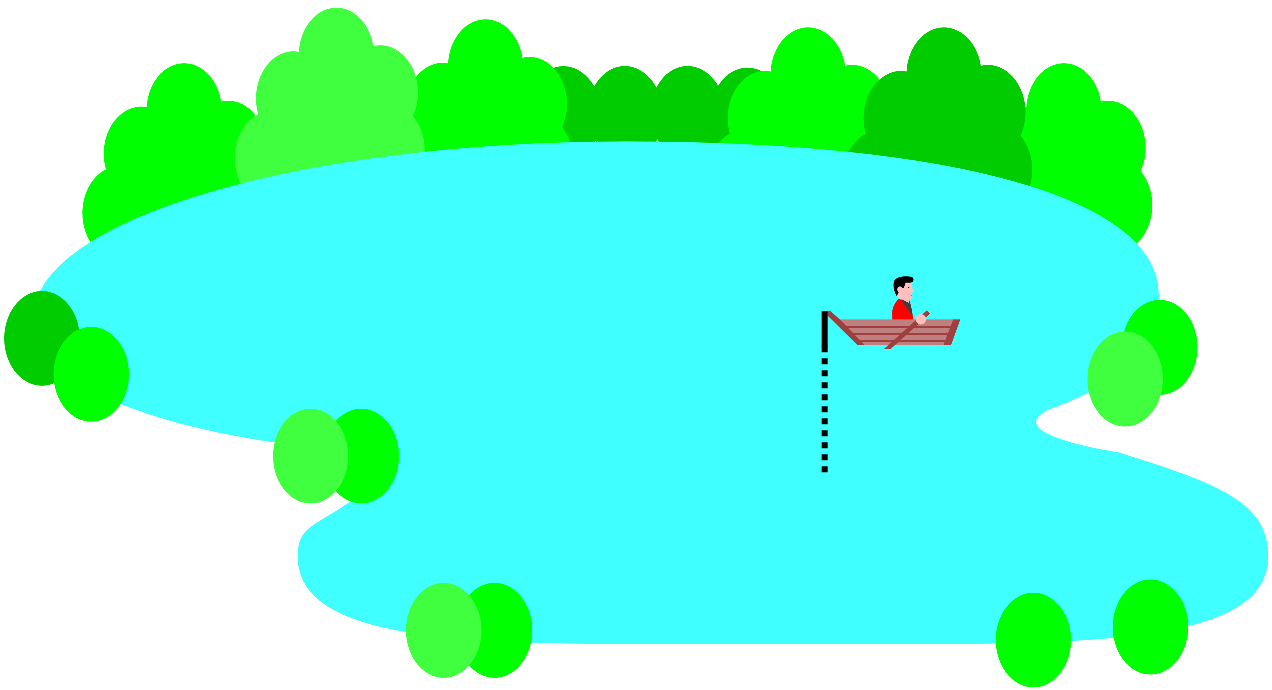
M. Hauschild, M. Pelikan, K. Sastry, D.E. Goldberg, *Using Previous Models to Bias Structural Learning in the Hierarchical BOA*

The Bayesian Optimization Algorithm

- **Advantages:**
    - The learned network preserves good “building blocks”
    - Can handle large decomposable problems more efficiently
- **Disadvantages:**
    - Training networks can be expensive

EGO – Response Surface

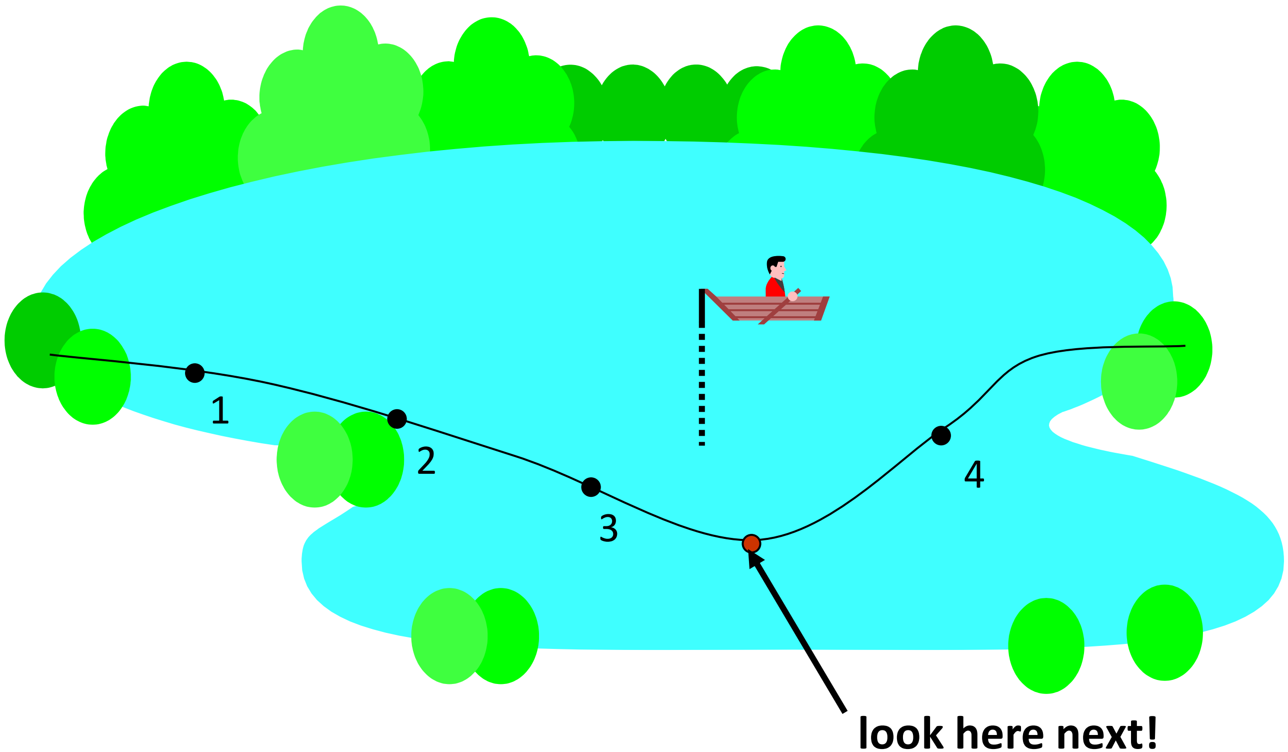
How do you find the deepest part of the lake when you can't see the bottom?



Take a series of depth measurements in strategic locations around the lake.

EGO – Response Surface

From an initial set of measurements, make a model of the bottom



Use the surrogate model to tell the boat driver where to measure the depth next

EGO - Kriging

Kriging: A geostatistical techniques to interpolate the elevation of the landscape as a function of the geographic location at an unobserved location from observations of its value at nearby locations.

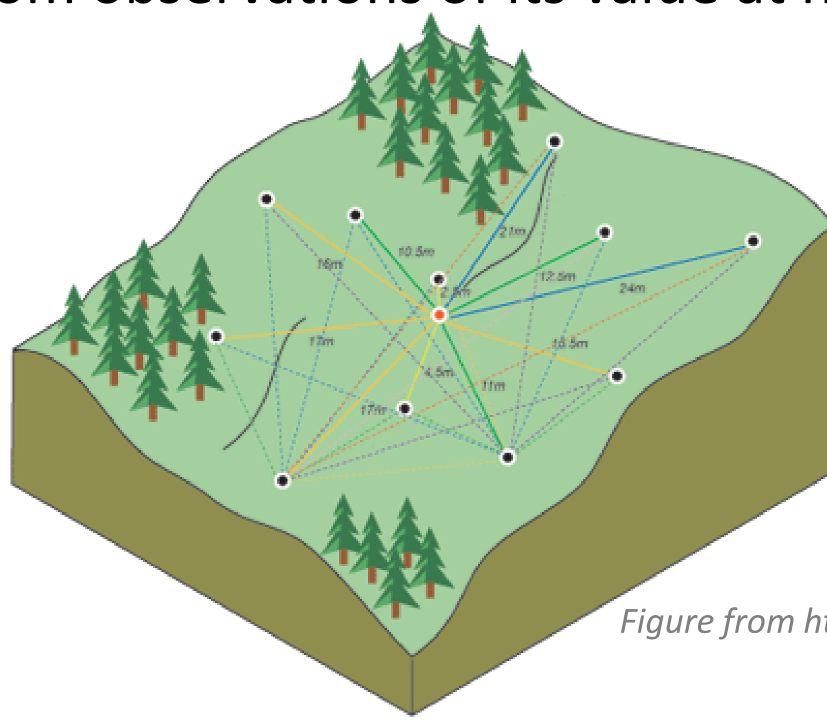


Figure from http://resources.esri.com

EGO – The Merit Function

In each iteration of EGO, we have two functions of x:

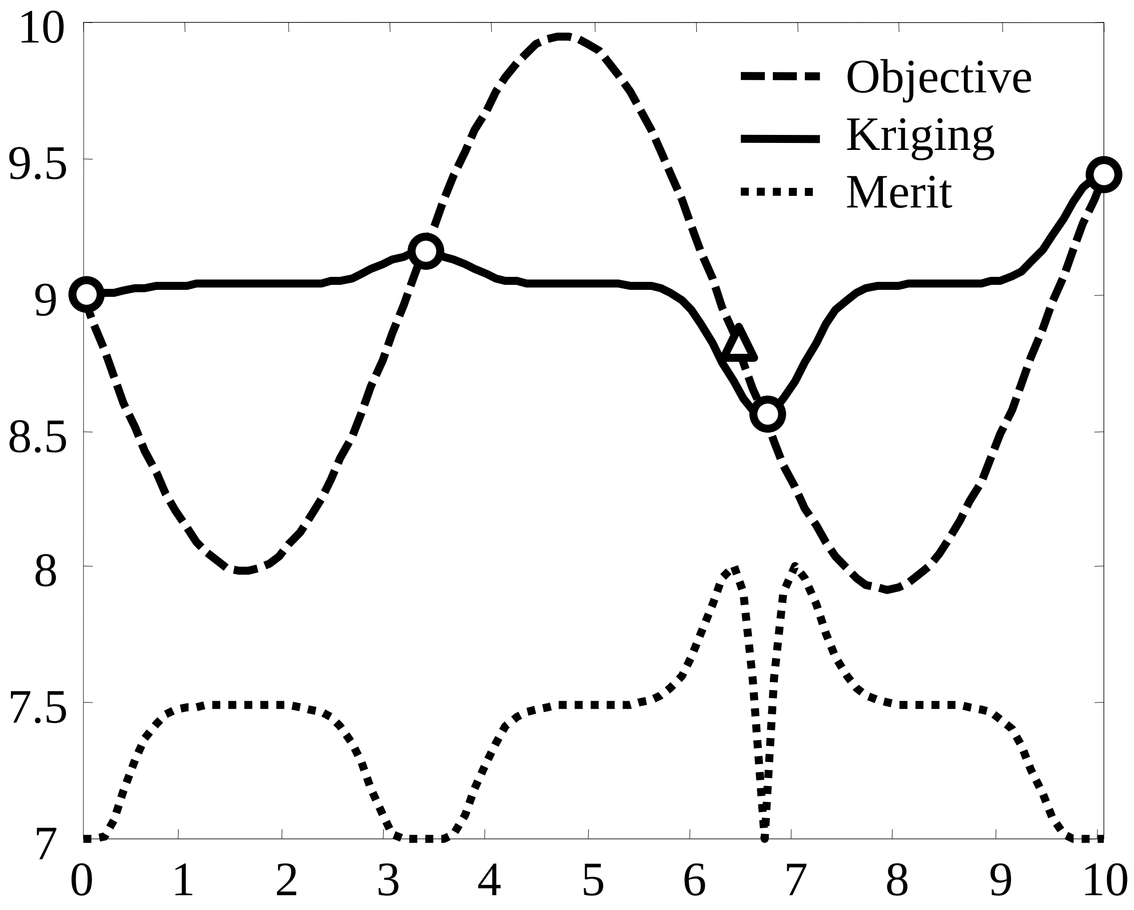
1) the Kriging model $\hat{y}$; 2) the MSE function *s*.

The best place to sample next will have low prediction $\hat{y}$ as well as high uncertainty *s.* The merit function reflects the "improvement" of the objective.

$$\boxed{f_{merit}(x) = (f_{min} - \hat{y})\Phi\left(\frac{f_{min} - \hat{y}}{s}\right) + s\phi\left(\frac{f_{min} - \hat{y}}{s}\right)}$$

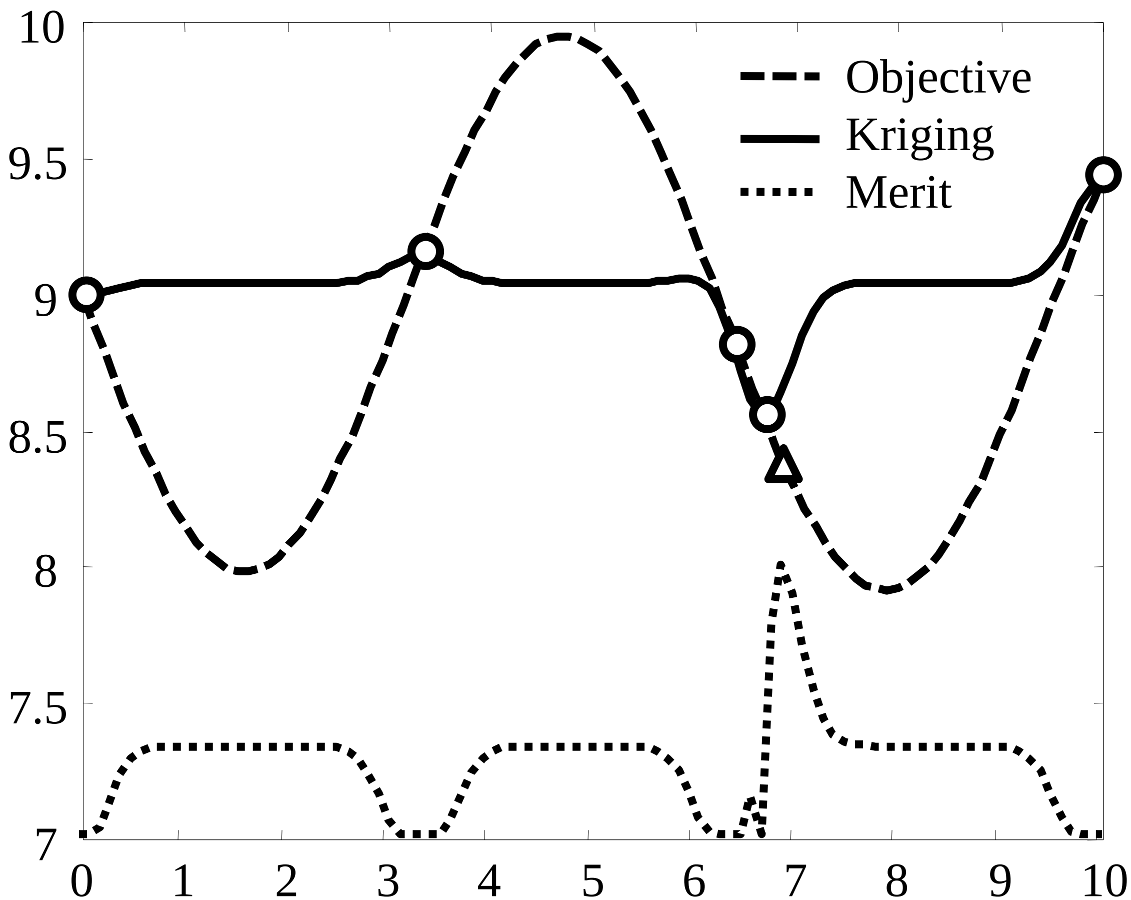
EGO - Example

Iteration #1



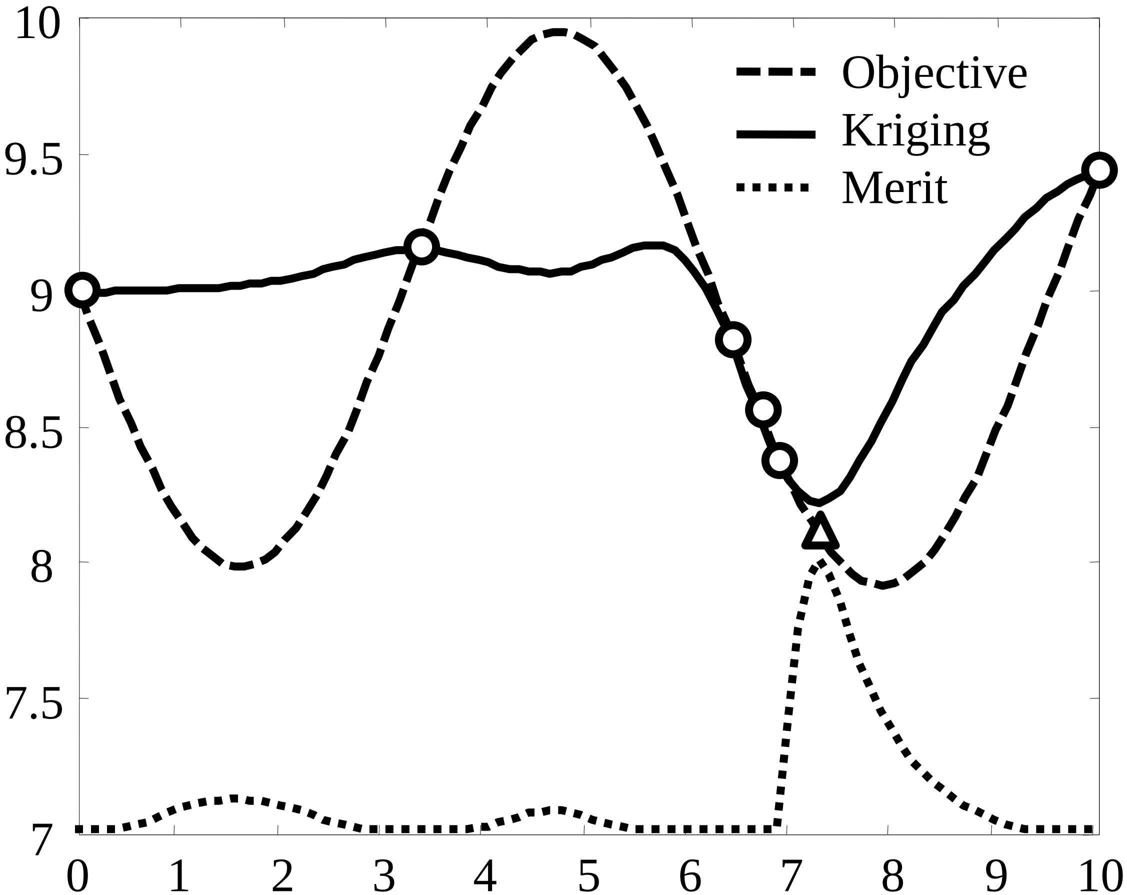
EGO - Example

Iteration #2



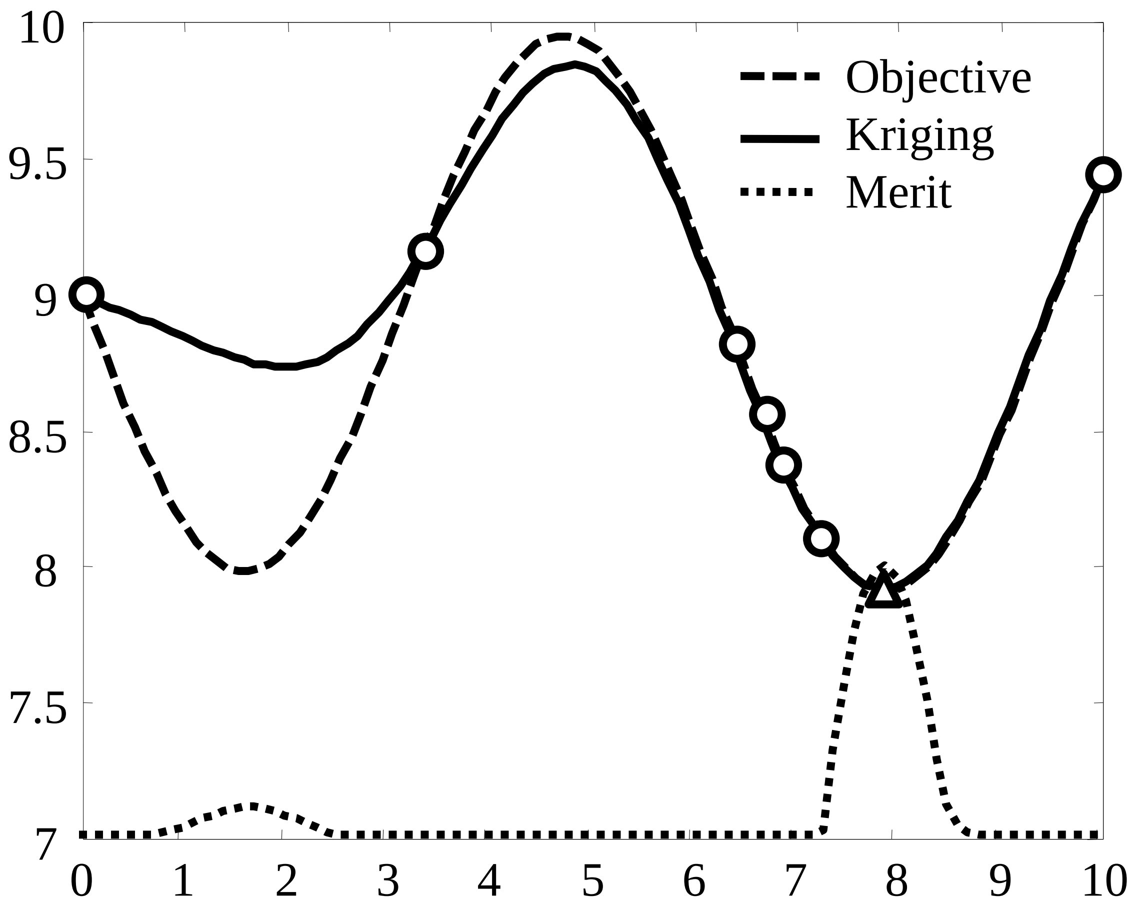
EGO - Example

Iteration #3



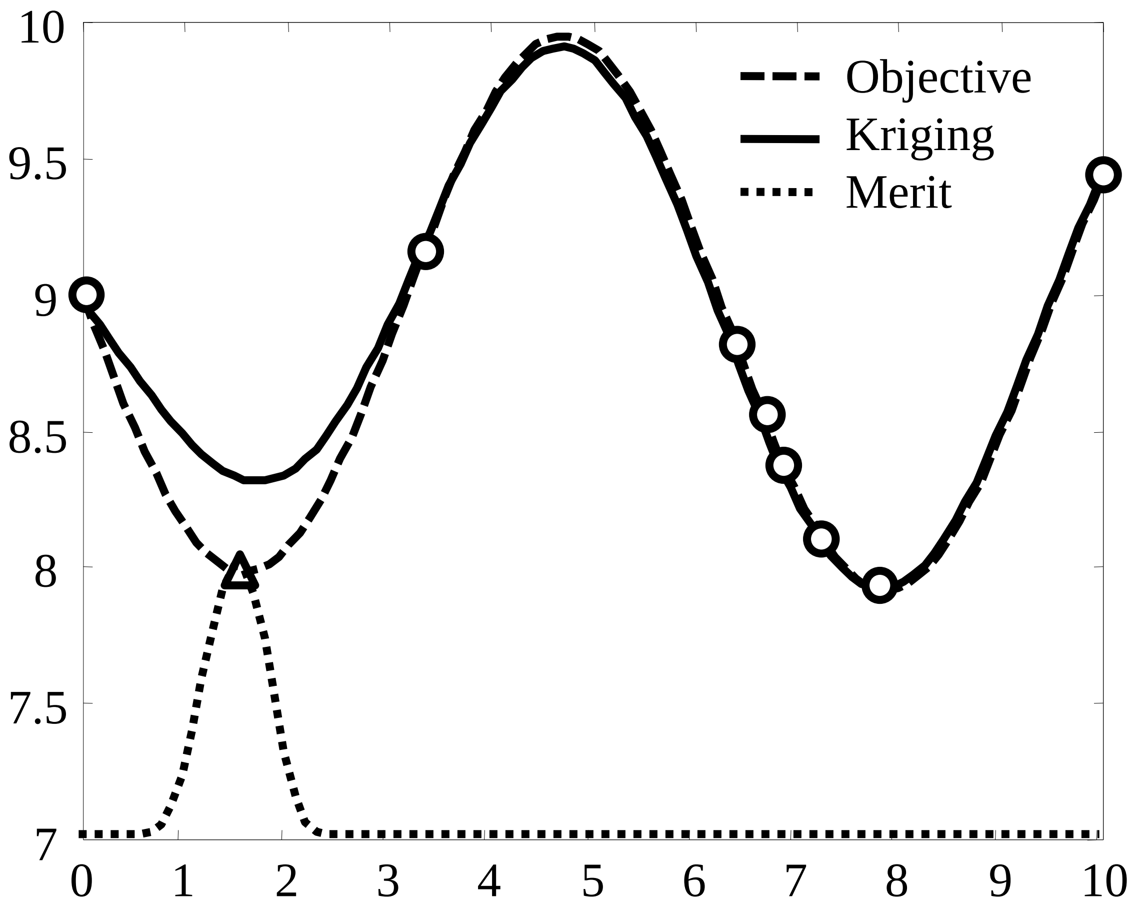
EGO - Example

Iteration #4



EGO - Example

Iteration #5



EGO – Pros/Cons

- **Advantages**
  - Creates surrogate model during search, which is advantageous for expensive functions
  - Surrogate model can smooth out noise and discontinuities
  - Balances global/local search, similar to DIRECT
- **Disadvantages**
  - Difficulty making surrogate model at high dimensions
  - Has to create surrogate model for each function, including constraints
  - Difficulty optimizing the merit function at high dimensions

Nelder-Mead Simplex

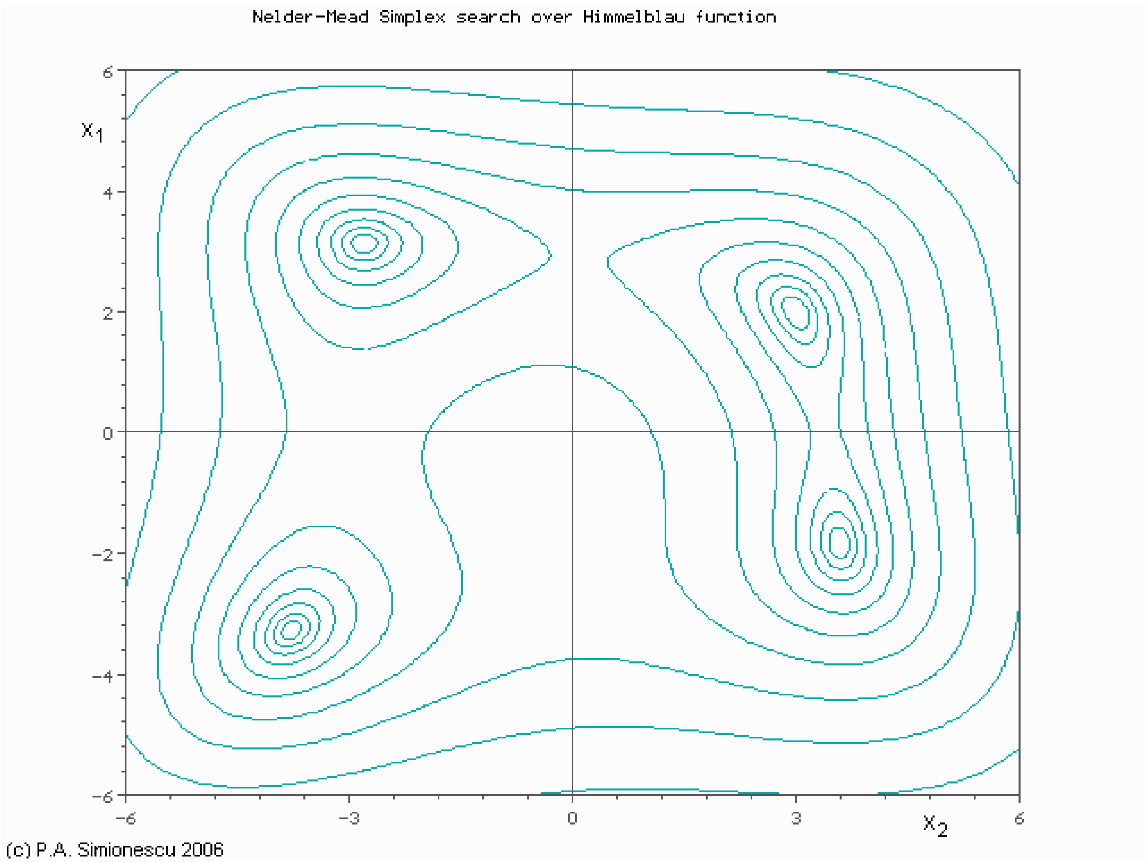
"Simplex" or "Polytope"
M+1 polygon

In this case, 3 vertices

$\mathbf{x}^{(1)} = (x_1^{(1)}, x_2^{(1)})$

$\mathbf{x}^{(2)} = (x_1^{(2)}, x_2^{(2)})$

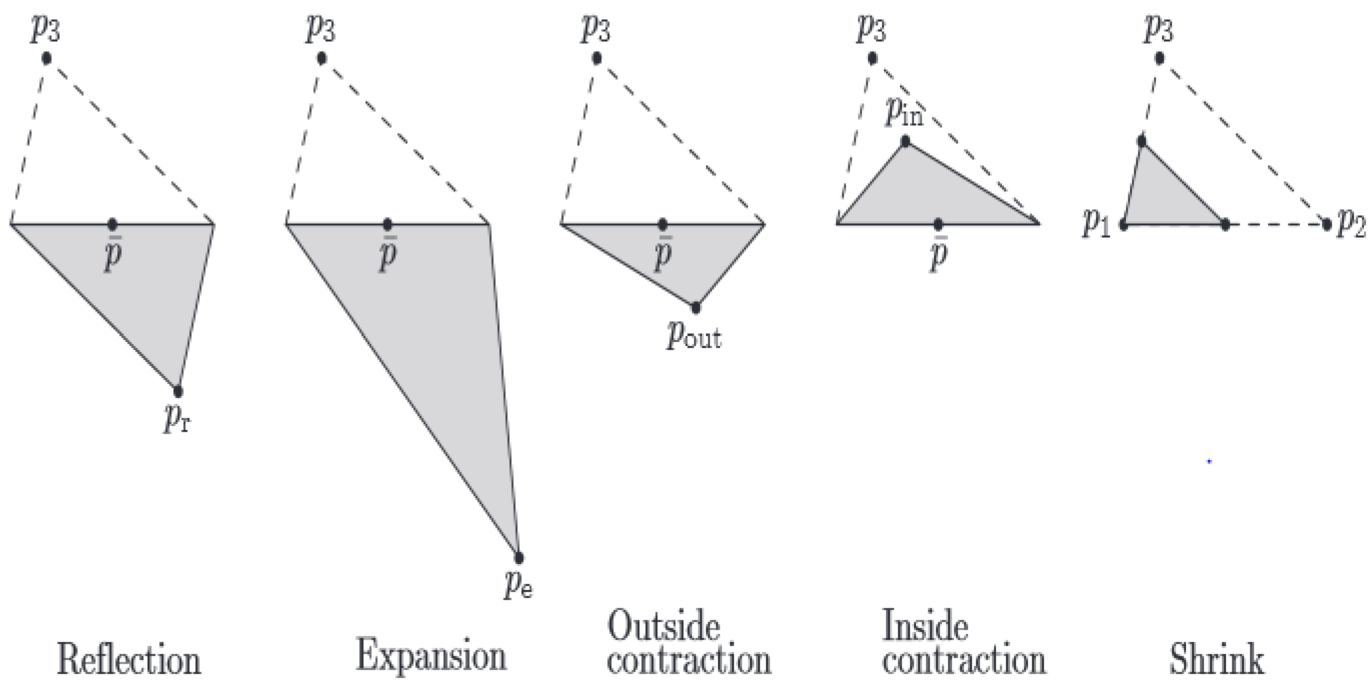
$\mathbf{x}^{(3)} = (x_1^{(3)}, x_2^{(3)})$



https://upload.wikimedia.org/wikipedia/commons/9/96/Nelder_Mead2.gif

The largest $f(\mathbf{x}^{(i)})$ of the $i = M + 1$ vertices changed to the centroid of the polytope

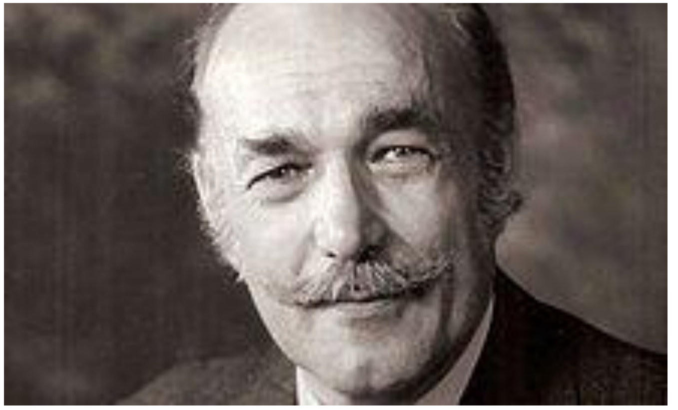
5 Possible Polytope Changes
(in order of when to try them)



http://www.math.uiuc.edu/documenta/vol-ismp/42_wright-margaret.pdf

Nelder-Mead:
Simple, intuitive, and effective

"Mathematicians hate it because you can't prove convergence; engineers seem to love it because it often works."



John Nelder
National Vegetable
Research Station

Over 2000 papers cited it in 2012 [1]

[1] Wright, Margaret H. "Nelder, Mead, and the other simplex method." *Documenta Mathematica* 7 (2010).

Pros/Cons of Pattern Search

Advantages

- Easy to implement
- Minimal parameter tuning

Disadvantages

- Can get stuck easily
- Very dependent on initialization point

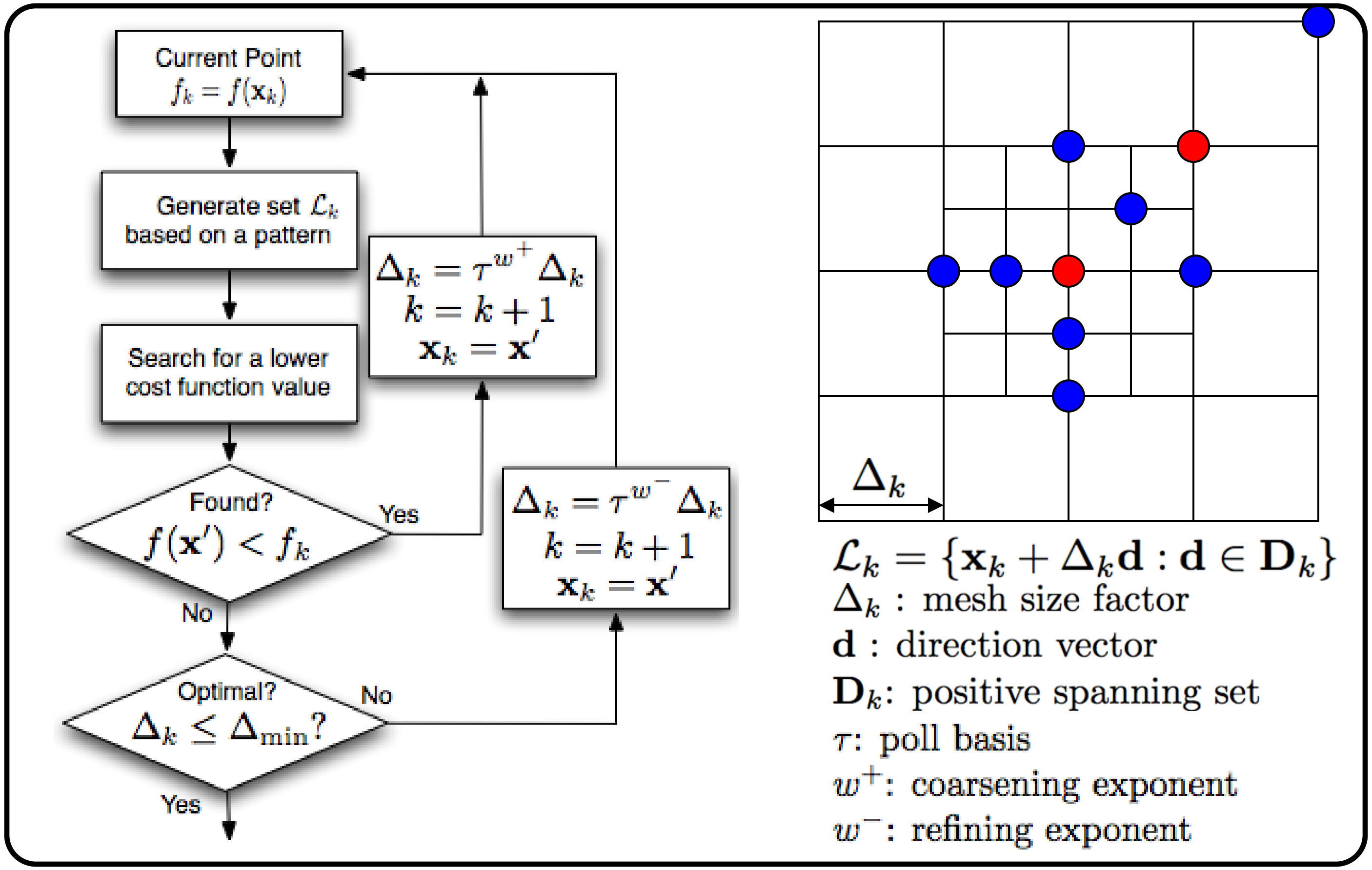
NOMAD – Overview

- Belongs to Pattern Search
- An implementation of the Mesh-Adaptive Direct Search (MADS) algorithm
- Pattern search method: creates mesh and samples along mesh

NOMAD – Pattern Search

Generalized Pattern Search (GPS)

- A number of points around the current point are evaluated
- Best point becomes center point for the next iteration.

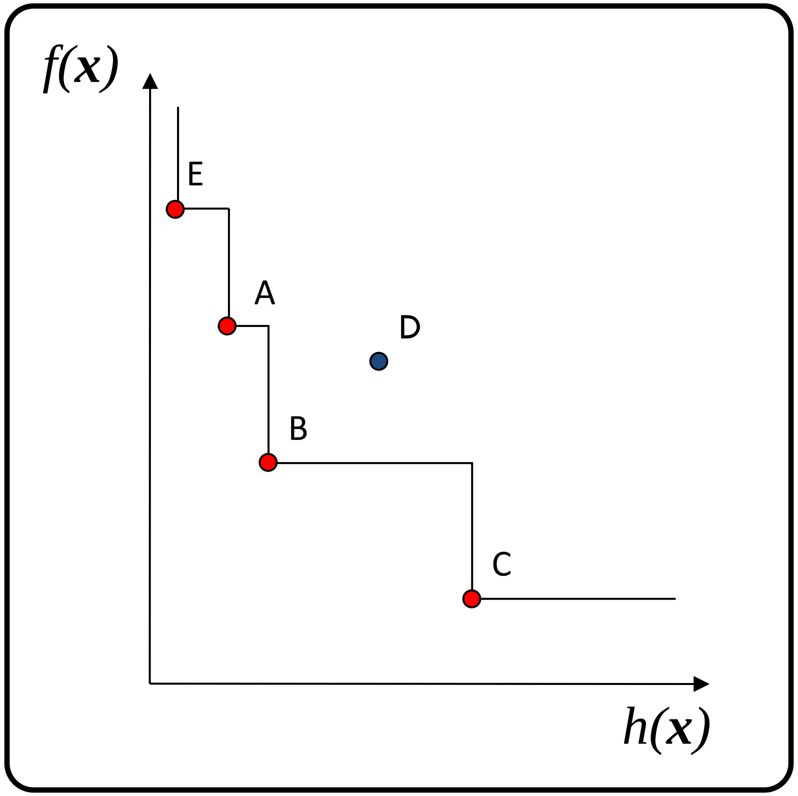


NOMAD – Constraint

- Bi-objective problem: minimize both the objective function, *f(x)*, and an aggregate constraint violation function:

$$h(\bar{x}) = \sum \max\{0, c_i(\bar{x})\}$$

- Chooses Pareto set of Best feasible/Least infeasible points



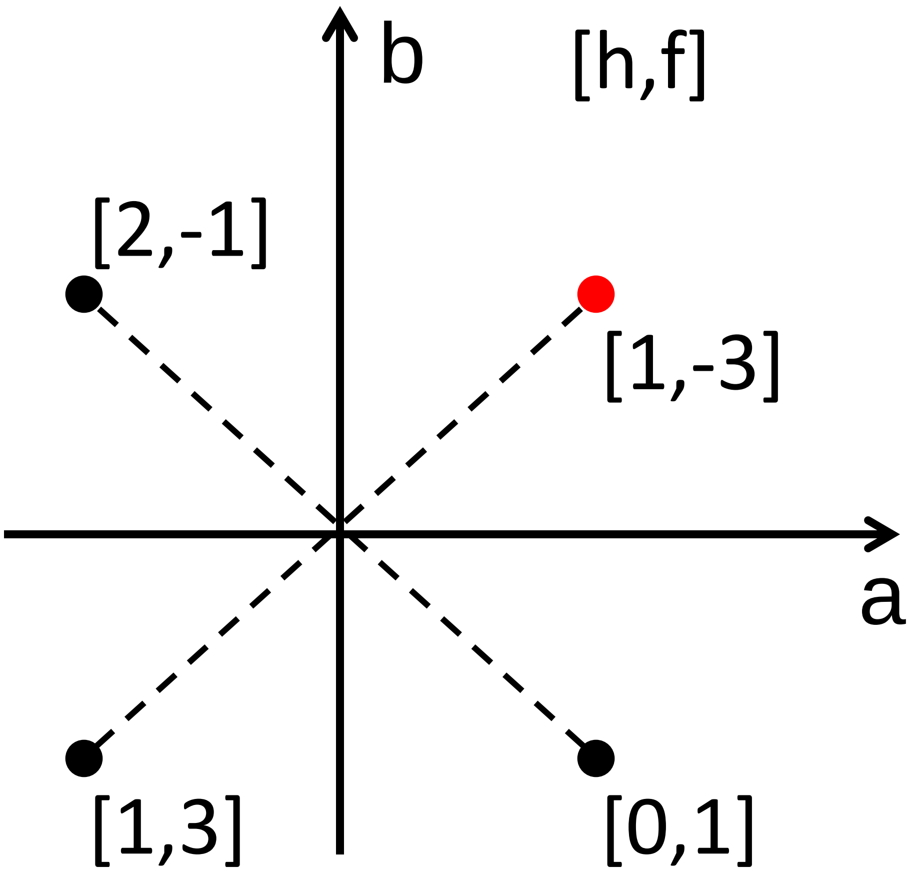
GPS– Example

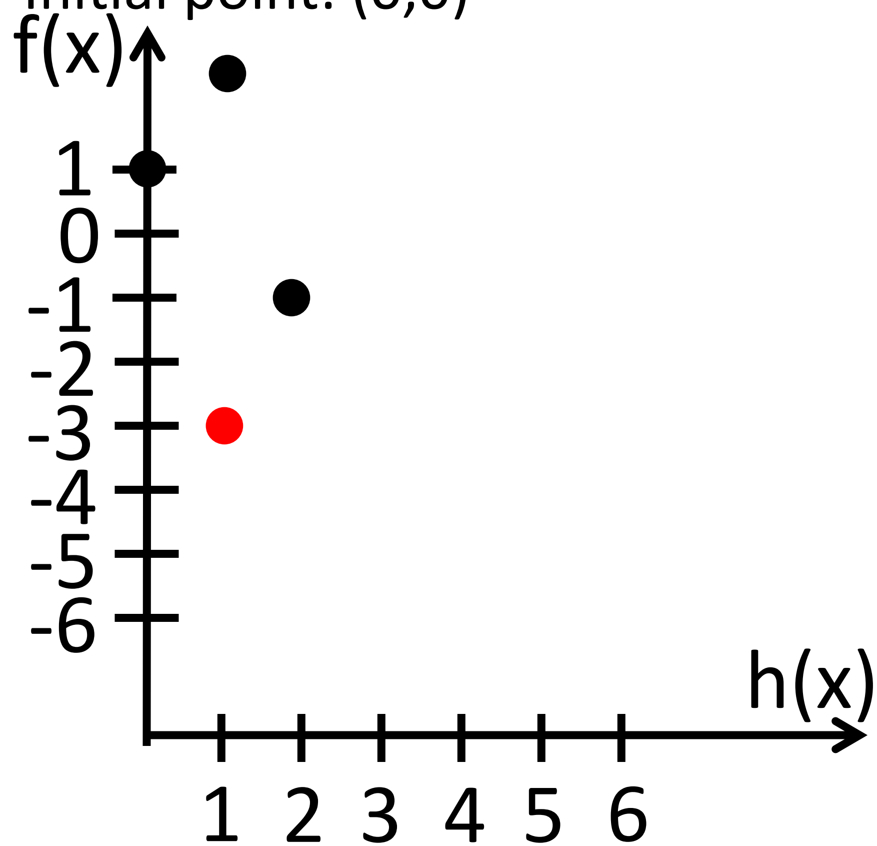
$$\min_{a,b} \quad -a-2b$$
$$\text{s.t.} \quad 0 \le a \le 1$$
$$b \le 0$$

GPS, Filter (least infeasible)

Directions: $\pm(1,1)^T, \pm(1,-1)^T$

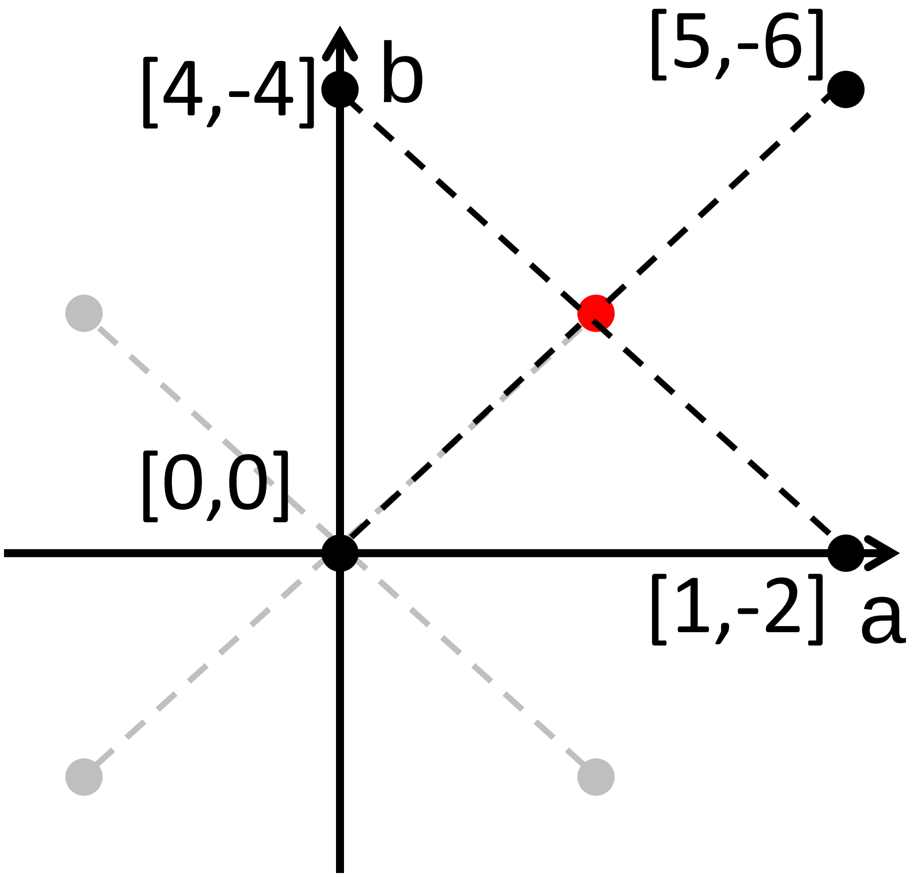
Initial point: $(0,0)^T$





GPS– Example

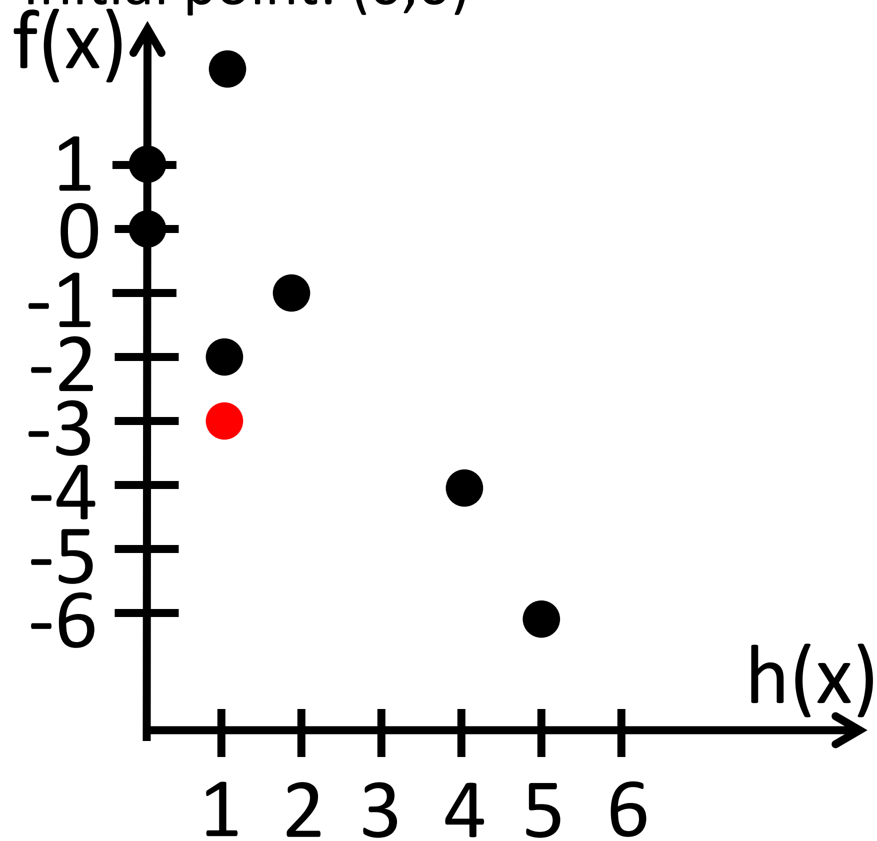
$$\min_{a,b} \quad -a-2b$$
$$\text{s.t.} \quad 0 \le a \le 1$$
$$b \le 0$$



GPS, Filter (least infeasible)

Directions: $\pm(1,1)^T, \pm(1,-1)^T$

Initial point: $(0,0)^T$



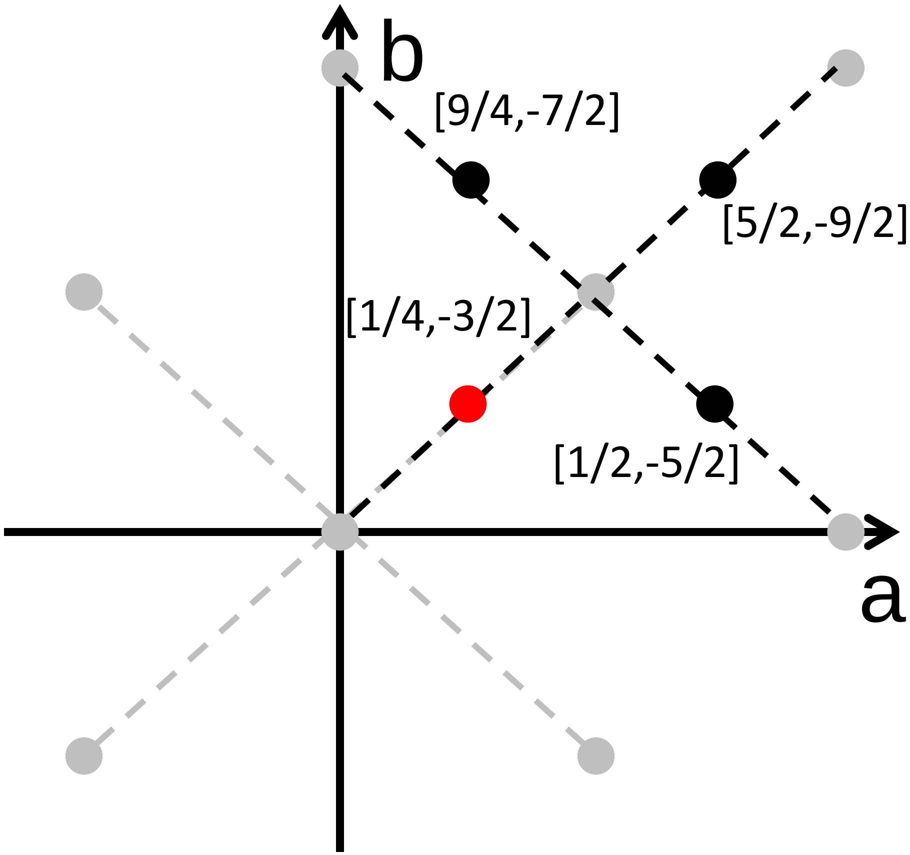
GPS– Example

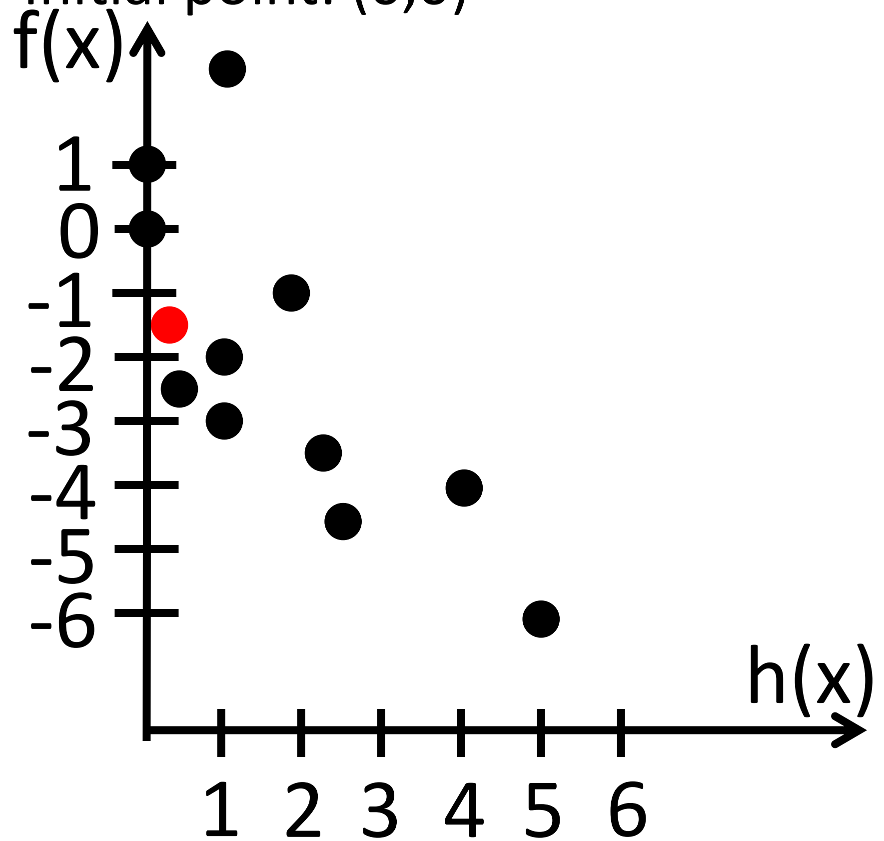
$$\min_{a,b} \quad -a-2b$$
$$\text{s.t.} \quad 0 \le a \le 1$$
$$b \le 0$$

GPS, Filter (least infeasible)

Directions: $\pm(1,1)^T, \pm(1,-1)^T$

Initial point: $(0,0)^T$





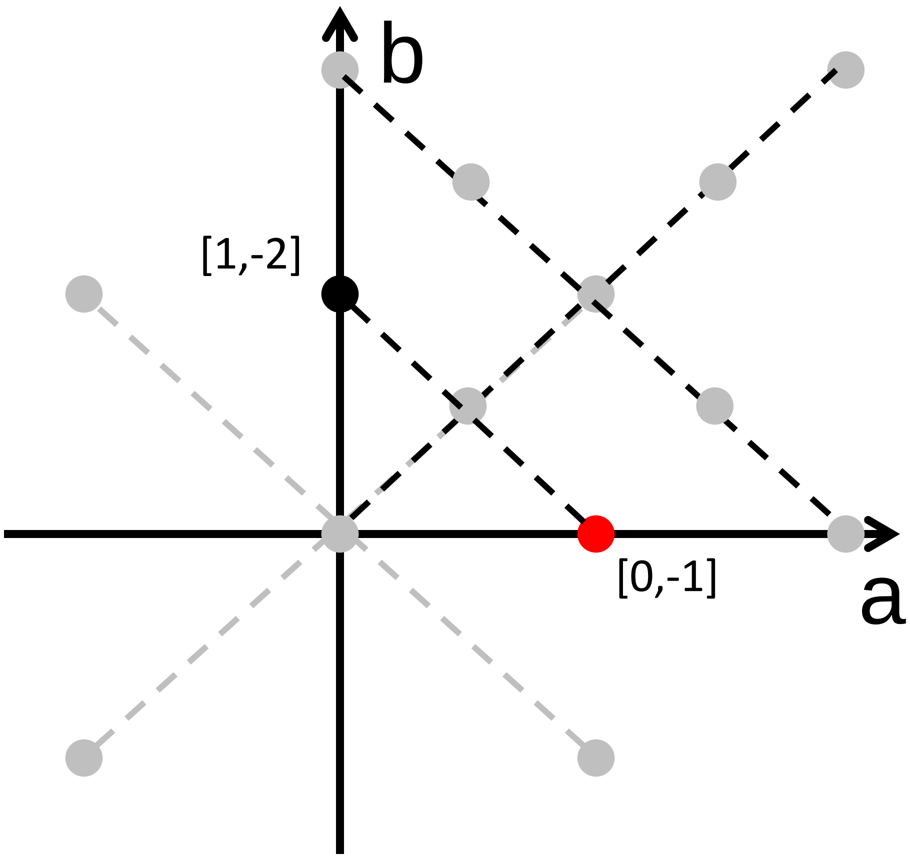
GPS– Example

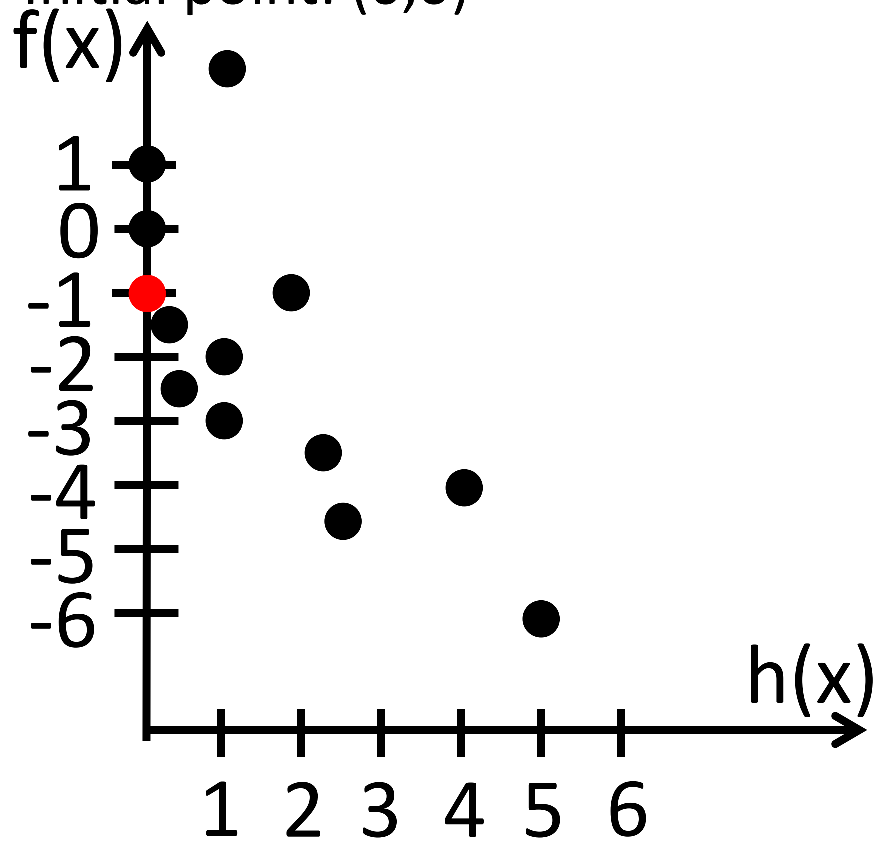
$$\min_{a,b} \quad -a - 2b$$
$$\text{s.t.} \quad 0 \le a \le 1$$
$$b \le 0$$

GPS, Filter (least infeasible)

Directions: $\pm(1,1)^T, \pm(1,-1)^T$

Initial point: $(0,0)^T$

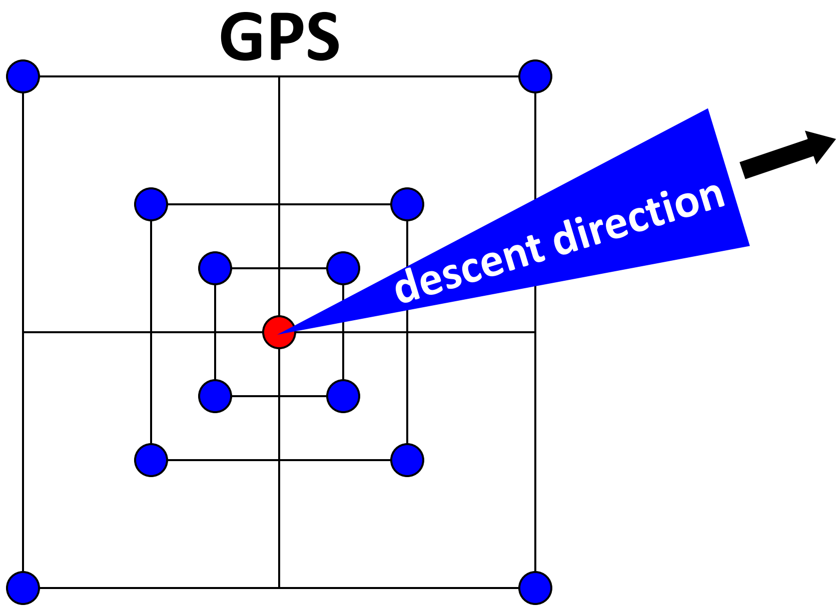


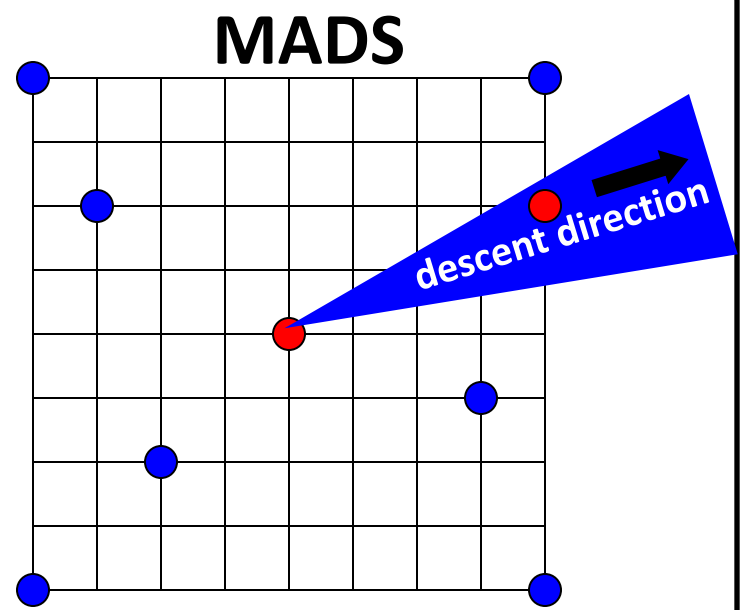


NOMAD – Pattern Search

- Mesh-Adaptive Direct Search (MADS)
  - GPS shows limitations due to the finite choices of directions
  - MADS removes the GPS restriction by allowing (nearly) infinitely many poll directions
  - Two parameters defining the frame size:
    mesh size Δ_k^m poll size Δ_k^p
  - mesh size ≤ poll size

NOMAD – Pattern Search





Can't find descent direction with finite poll directions

$$\Delta_1^m = \Delta_1^p = 1$$

$$\Delta_2^m = \Delta_2^p = 0.5$$

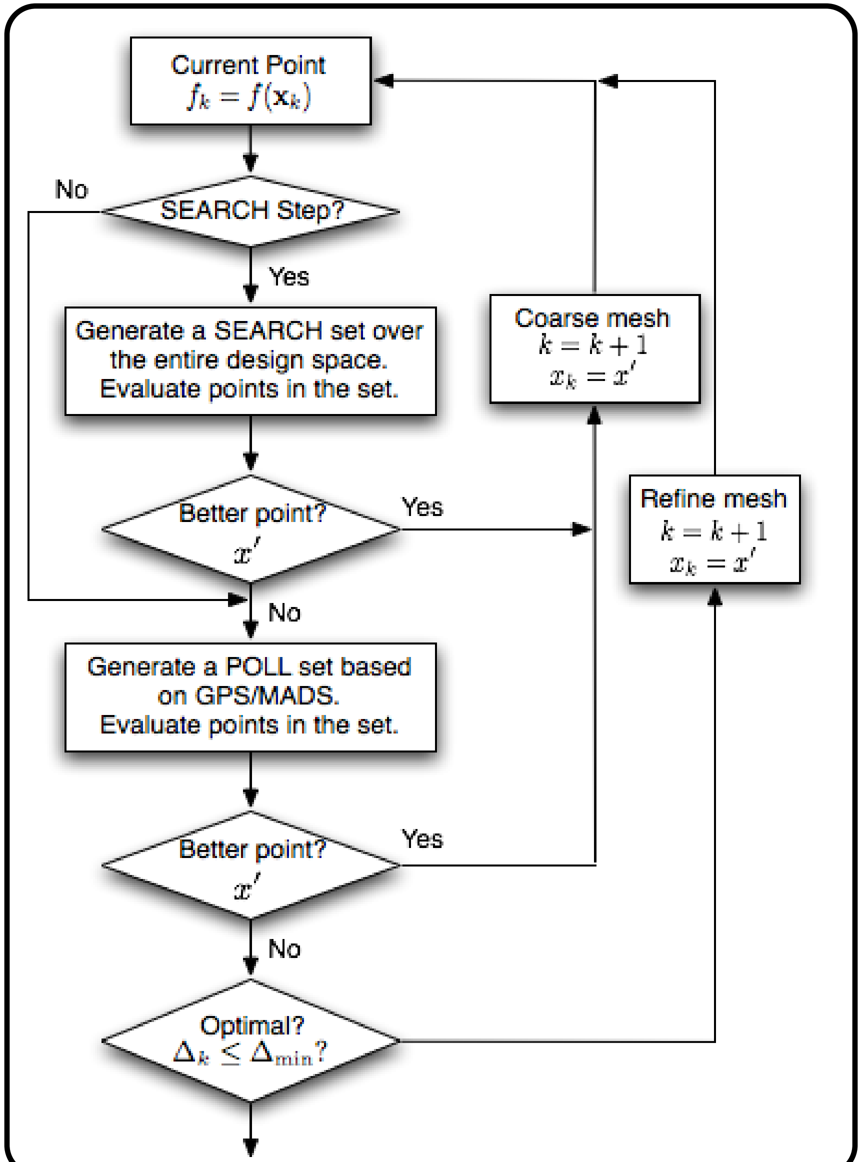
$$\Delta_3^m = \Delta_3^p = 0.25$$

Able to find descent direction due to infinitely many poll directions

$$\Delta_1^m = 1 \qquad \Delta_1^p = 1$$

$$\Delta_2^m = 0.25 \quad \Delta_2^p = 1$$

NOMAD



- Initial SEARCH step (optional)
  - Random search
  - Genetic algorithm
  - Latin hypercube
  - Orthogonal array
  - Etc.
- POLL step (MADS/GPS)
- Termination criteria based on mesh size

NOMAD – Pros/Cons

- **Advantages**
  - Can use discrete and categorical variables
  - Can integrate other algorithms (e.g. DIRECT) as part of search
  - Good combination of Global/Local searching
  - Can use gradient information, if available
- **Disadvantages**
  - Poll steps can require a large number of function evaluations in higher dimensions (though n+1 is no larger than finite differencing for a gradient algorithm)
  - Can terminate early if gets stuck in one area

Folk Wisdom – General Optimization

Always start with a gradient method

Try “multistart” with gradient method

Then try a number of *easy* to use nongradient methods…

…easy if software exists

…easy if not many parameters

Heuristic Name	Stochastic/ Deterministic	Constraint Handling	Termination Criteria	Discrete?	Availability
Simulated Annealing	Stochastic	Weighted Penalty	min. improvement tolerance	Y	Matlab, Optimus, iSight
Genetic Algorithm	Stochastic	Weighted Penalty	#generations/ fitness change	Y	Matlab, iSight
DIRECT	Deterministic	Weighted Penalty	#function calls	Y	Matlab, Tomlab
EGO	Stochastic or Deterministic	Response Surface	ask Optimus	N	Tomlab, Optimus
NOMAD	Stochastic or Deterministic	Pareto Set	min. mesh size #function calls	Y	Matlab

Credits

Some material adapted from:

A. Burnap, AE. Bayrak, N. Kang, University of Michigan

and

Prof. Kokkalaras, McGill University

Thank you