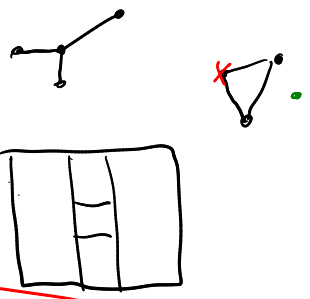
Last Time: Direct / Pattern Search

- Coordinate Search
- Generalized Patern Search
- Nelder Mead Simplex Method
- DIRECT (Lipschitz-Inspired)



Today: Stochastic + Population Methods

- Simulated Annealing
- Cross-Entropy

2 challenges

1. Finding local min within neighborhood ← so far
2. Finding neighborhood with lowest min

Simulated Annealing

temperature decay schedule → hyperparameters D, $t(k)$

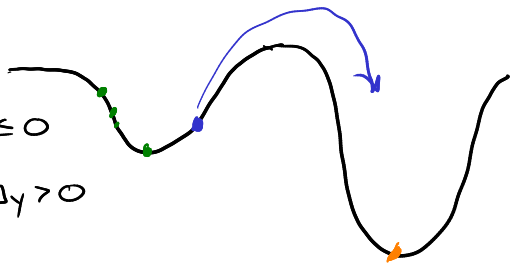
initialize x

loop

$x' \leftarrow x + \text{rand}(D)$

$\Delta y \leftarrow f(x') - f(x)$

accept $\leftarrow \text{rand}(U(0,1)) \leq \begin{cases} 1 & \text{if } \Delta y \leq 0 \\ e^{-\Delta y / t(k)} & \text{if } \Delta y > 0 \end{cases}$



if accept

$x \leftarrow x'$

Common temperature schedules

$t^k = t^1 \ln(2) / \ln(k+1)$ "logarithmic" asymptotically optimal, very slow

$t^{k+1} = \gamma t^k$ "exponential"

$t^k = \frac{t^1}{k}$ "fast annealing"

Cross-Entropy Method

Parameterized Distribution $D(\theta)$

initialize θ ← Gaussian: μ, Σ hyperparameters: D, m, m_{elite}

loop

population ← sample m from $D(\theta)$

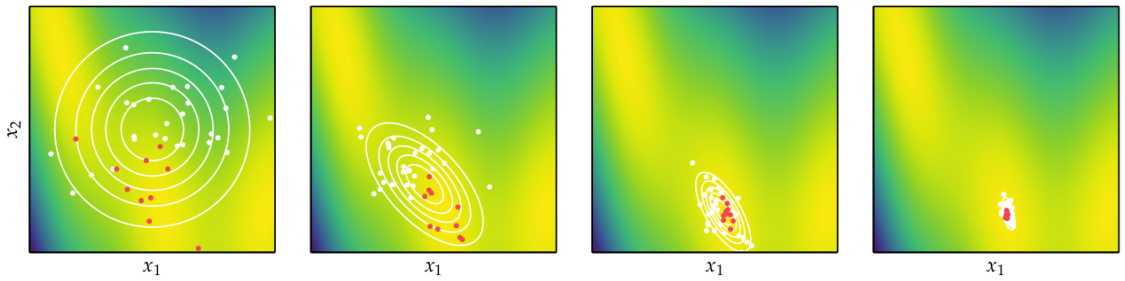
elite ← select m_{elite} best samples

$\theta \leftarrow$ fit to elite ← minimize "cross entropy"

fit: If D is Gaussian $\theta = (\mu, \Sigma)$

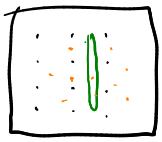
$$\vec{\mu}^{k+1} \leftarrow \frac{1}{m_{elite}} \sum_{i=1}^{m_{elite}} \vec{x}^i$$

$$\Sigma^{k+1} \leftarrow \frac{1}{m_{elite}} \sum_{i=1}^{m_{elite}} (\vec{x}^i - \vec{\mu}^{k+1})(\vec{x}^i - \vec{\mu}^{k+1})^T$$

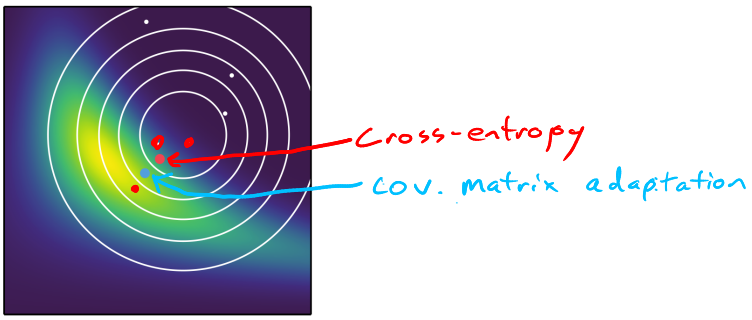


Other Stochastic Algorithms

- Mesh-adaptive direct search ← stochastic pattern search



- Natural Evolution Strategies
  estimate $\nabla_\theta E[f(\vec{x})]$ rather than fitting θ to elite
- Covariance Matrix Adaptation
  - similar to cross-entropy, but weight elite samples



Population Methods

Genetic Algorithms

Chromosome: vector of bits / real numbers

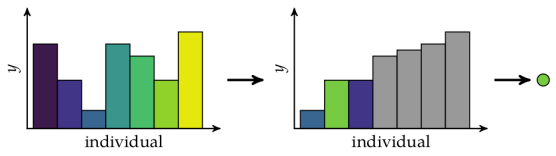
[0, 1, 1, 0 . . .]

[5.2, 6.3 . . .] $\vec{x}$ ← in practice

Loop
- Selection
- Crossover
- Mutation

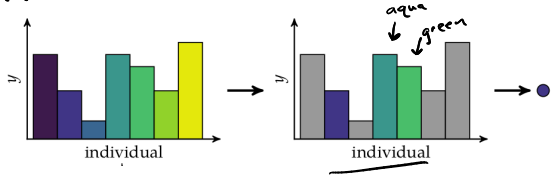
Selection

Truncation



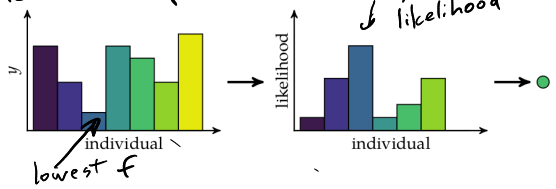
select best k individuals
randomly select from best k

Tournament



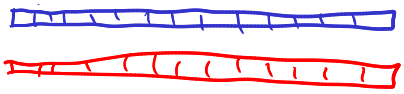
randomly select k
take best

Roulette Wheel



Crossover

Parent 1
Parent 2



Single Point
child

Two Point

Uniform

Mutation

Bitwise: each bit has an independent probability of flipping
Gaussian: each element has Gaussian noise added to it

Particle Swarm

initialize population $\{{}^i\vec{x}\}_{i=1}^m$

hyperparams: w, c_1, c_2

loop

for $i \in 1..m$

$${}^i\vec{x} \leftarrow {}^i\vec{x} + {}^i\vec{v}$$

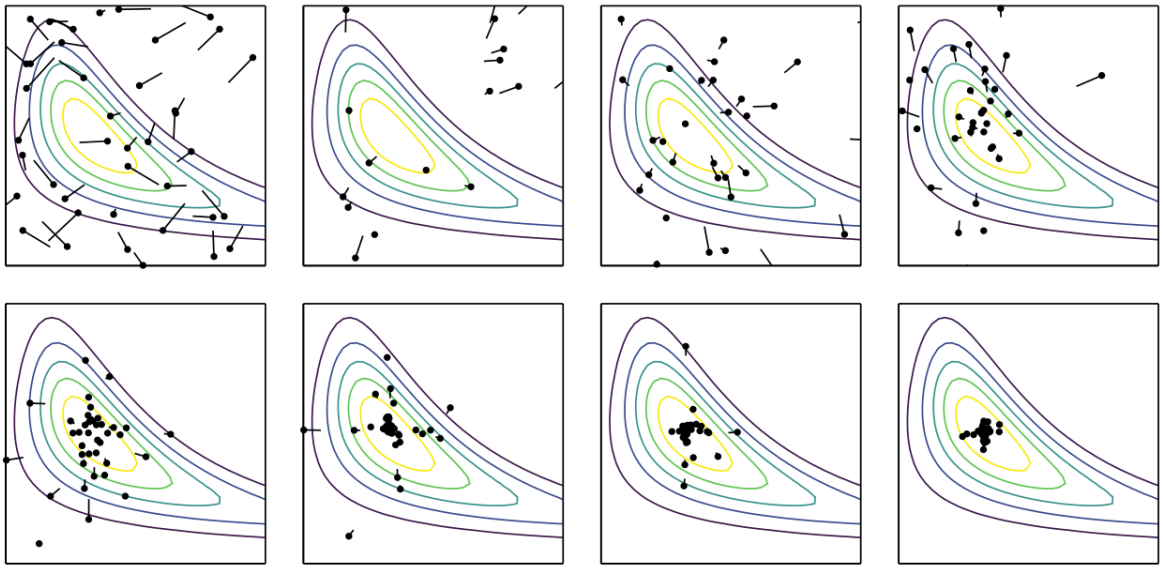
$${}^i v \leftarrow w\,{}^i v + c_1 r_1 \left({}^i\vec{x}_{best} - {}^i\vec{x}\right) + c_2 r_2 \left(\vec{x}_{best} - {}^i\vec{x}\right)$$

$r_1, r_2 \leftarrow \text{rand}(U(1,0))$

if $f({}^i\vec{x}) < f({}^i\vec{x}_{best})$

$${}^i\vec{x}_{best} \leftarrow {}^i\vec{x}$$

update $\vec{x}_{best}$



Firefly Algorithm

if $f(\vec{b}) < f(\vec{a})$

$\vec{a}$ moves toward $\vec{b}$

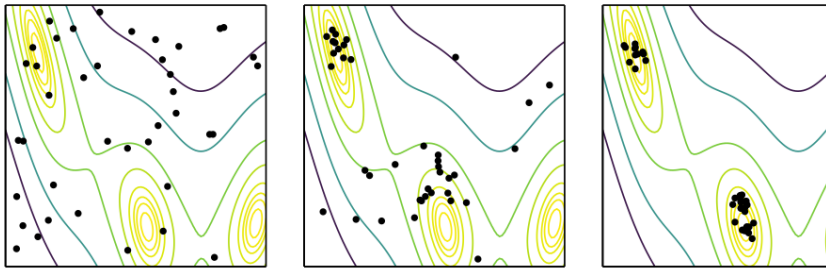
$$\vec{a} \leftarrow \vec{a} + \beta I\left(\|\vec{b}-\vec{a}\|\right)(\vec{b}-\vec{a}) + \alpha\varepsilon$$

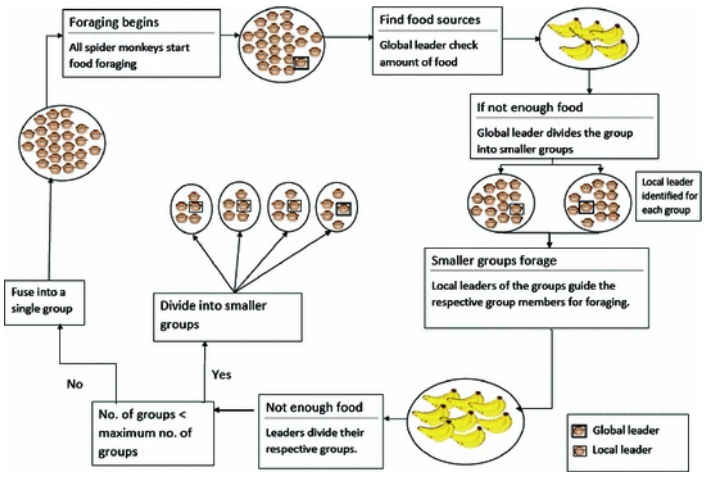
(I: intensity; β, α: hyperparameters; ε: Gaussian)

$$I(r) = \frac{1}{r^2}$$

$$I(r) = e^{-\gamma r}$$

$$I(r) = e^{-\gamma r^2}$$





§ These algorithms include the following: ant colony optimization, artificial bee colony algorithm, artificial fish swarm, artificial flora optimization algorithm, bacterial foraging optimization, bat algorithm, big bang–big crunch algorithm, biogeography-based optimization, bird mating optimizer, cat swarm optimization, cockroach swarm optimization, cuckoo search, design by shopping paradigm, dolphin echolocation algorithm, elephant herding optimization, firefly algorithm, flower pollination algorithm, fruit fly optimization algorithm, galactic swarm optimization, gray wolf optimizer, grenade explosion method, harmony search algorithm, hummingbird optimization algorithm, hybrid glowworm swarm optimization algorithm, imperialist competitive algorithm, intelligent water drops, invasive weed optimization, mine bomb algorithm, monarch butterfly optimization, moth-flame optimization algorithm, penguin search optimization algorithm, quantum-behaved particle swarm optimization, salp swarm algorithm, teaching–learning-based optimiza-

Why to use a meta heuristic alg. (esp. nature inspired)

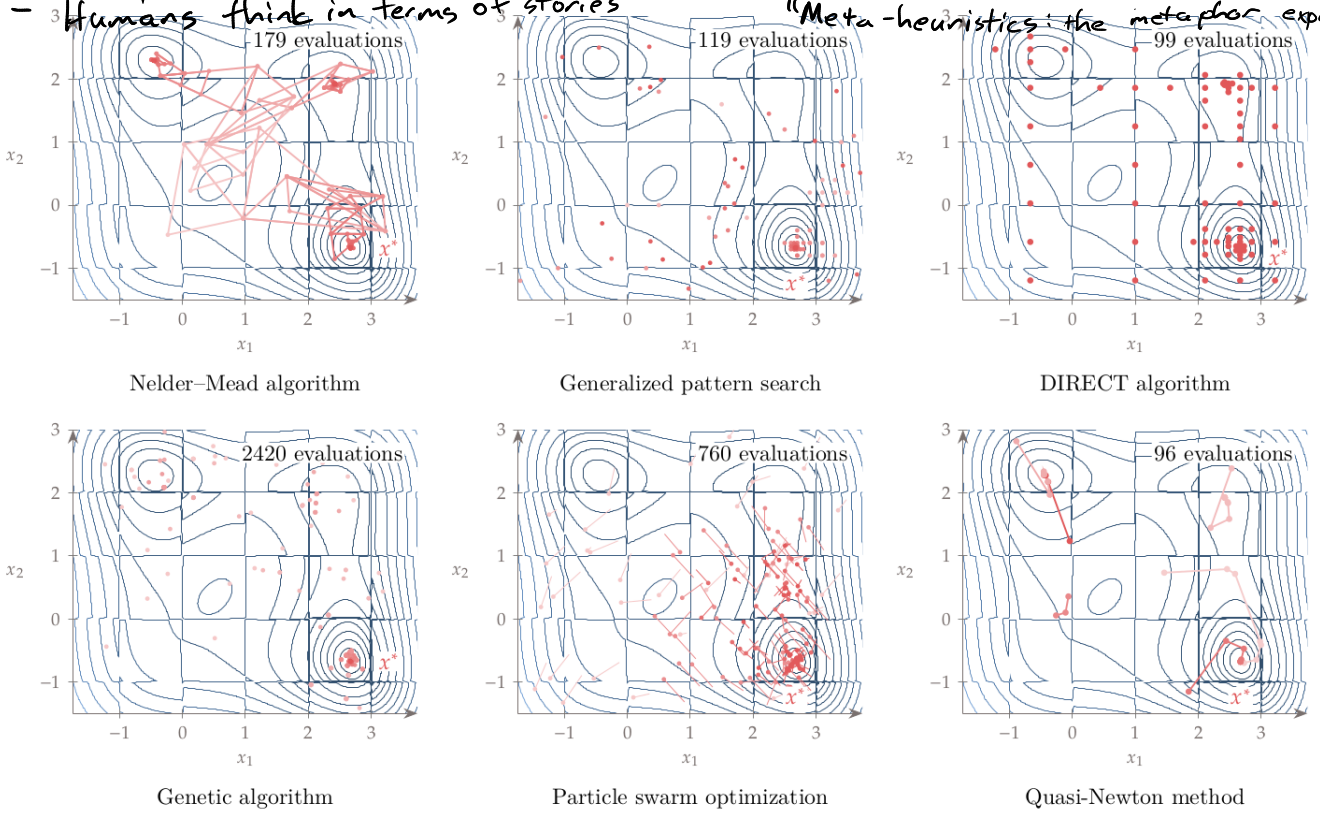
Pro

- There are NP-hard problems, usually involve many local minima
- Good for cheap function evaluation
- No need to be continuous/diff (useful for black-box)
- Highly Parallelizable
- Humans think in terms of stories

Con

- Few Guarantees
- Hard to know if a particular alg will work (hyperparameters)
- High population (bad if function eval is expensive)
- Poor scaling to high dimension (see below)
- Each new alg has own set of metaphors

"Meta-heuristics: the metaphor exposed"



Nelder–Mead algorithm | Generalized pattern search | DIRECT algorithm

Genetic algorithm | Particle swarm optimization | Quasi-Newton method

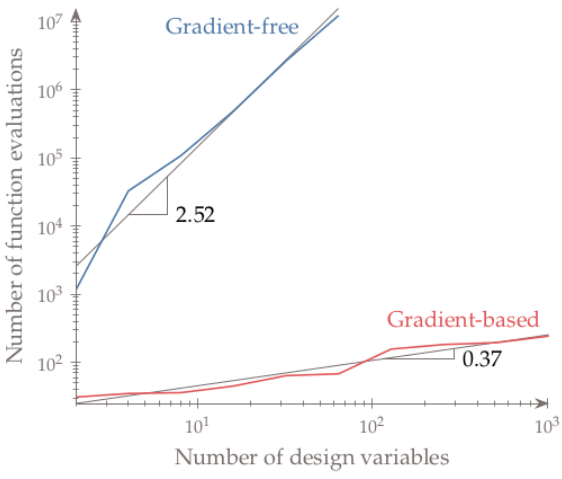


Fig. 7.1 Cost of optimization for increasing number of design variables in the n-dimensional Rosenbrock function. A gradient-free algorithm is compared with a gradient-based algorithm, with gradients computed analytically. The gradient-based algorithm has much better scalability.

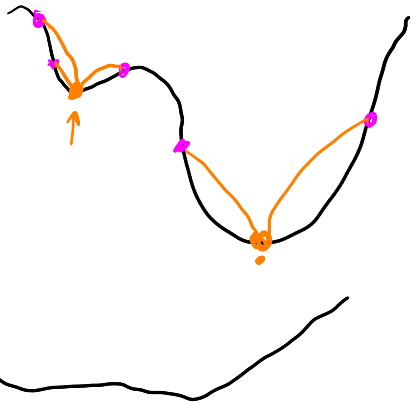
Hybrid - Approach

stochastic / population → explore new neighborhoods

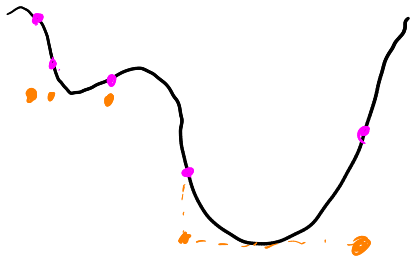
descent → find local minimum within neighborhood

$\hat{f}(x) = f(x^{*2})$

$\hat{f}(x) = f(x^{*1})$



Lamarckian



Baldwinian

Advantage: keeps points spread out