SWARM INTELLIGENCE METHODS FOR STATISTICAL REGRESSION

Lecture 2

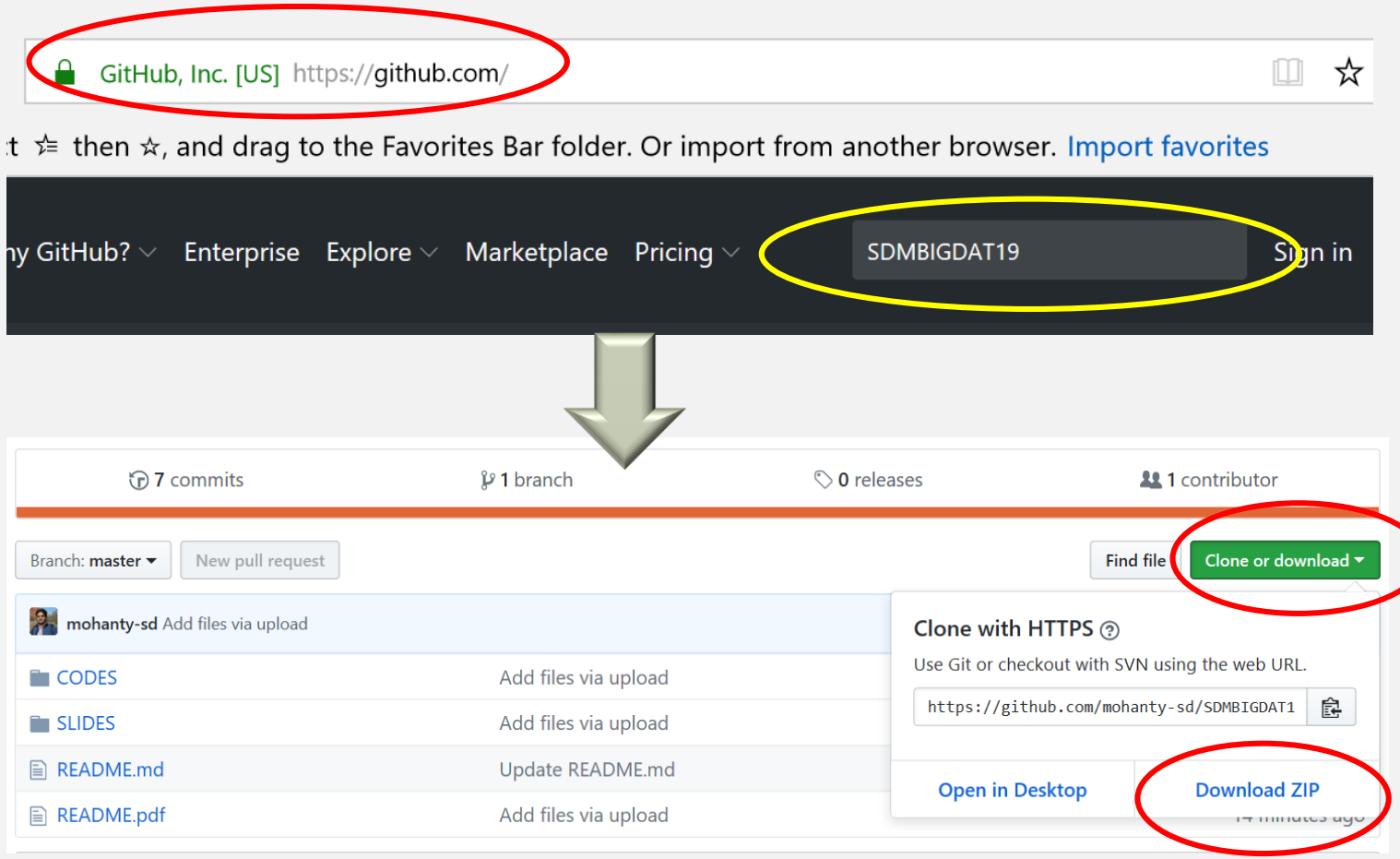
Soumya D. Mohanty

University of Texas Rio Grande Valley

ANNOUNCEMENT

- GitHub repository: SDMBIGDAT19
- Latest drafts of lecture slides
- Codes
  - Description and user manual to be added soon
- Repository will be deleted after end of course!

❖ Google drive shared folder remains primary outlet but slower update cycle

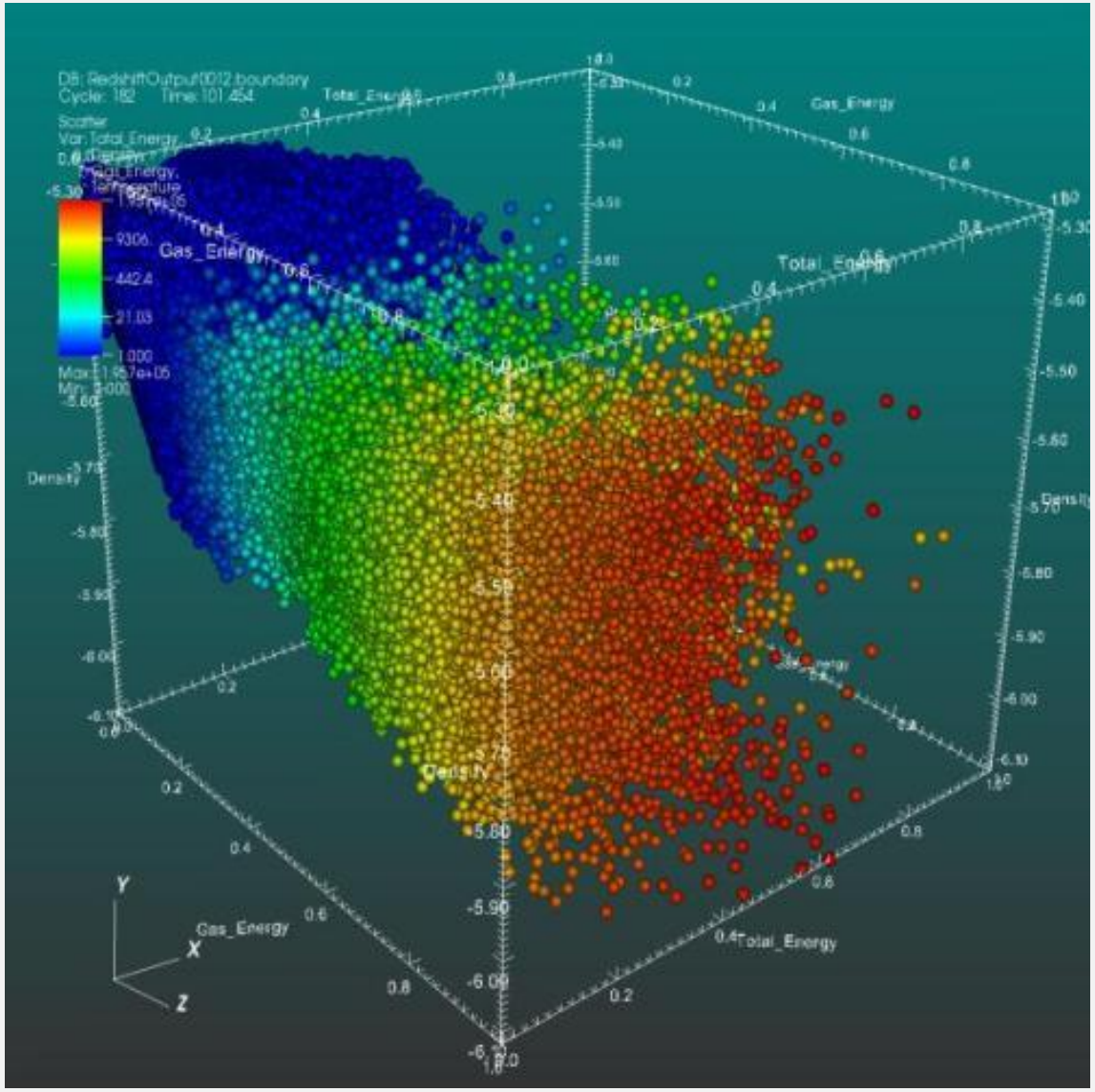


MOTIVATION

For large and complex data sets in the big data era, we need more flexible models

Flexibility may require models to be

- High-dimensional and/or
- Non-linear



Wikipedia: Data visualization

MOTIVATION

Global optimization of high-dimensional, non-linear statistical models is a challenging task

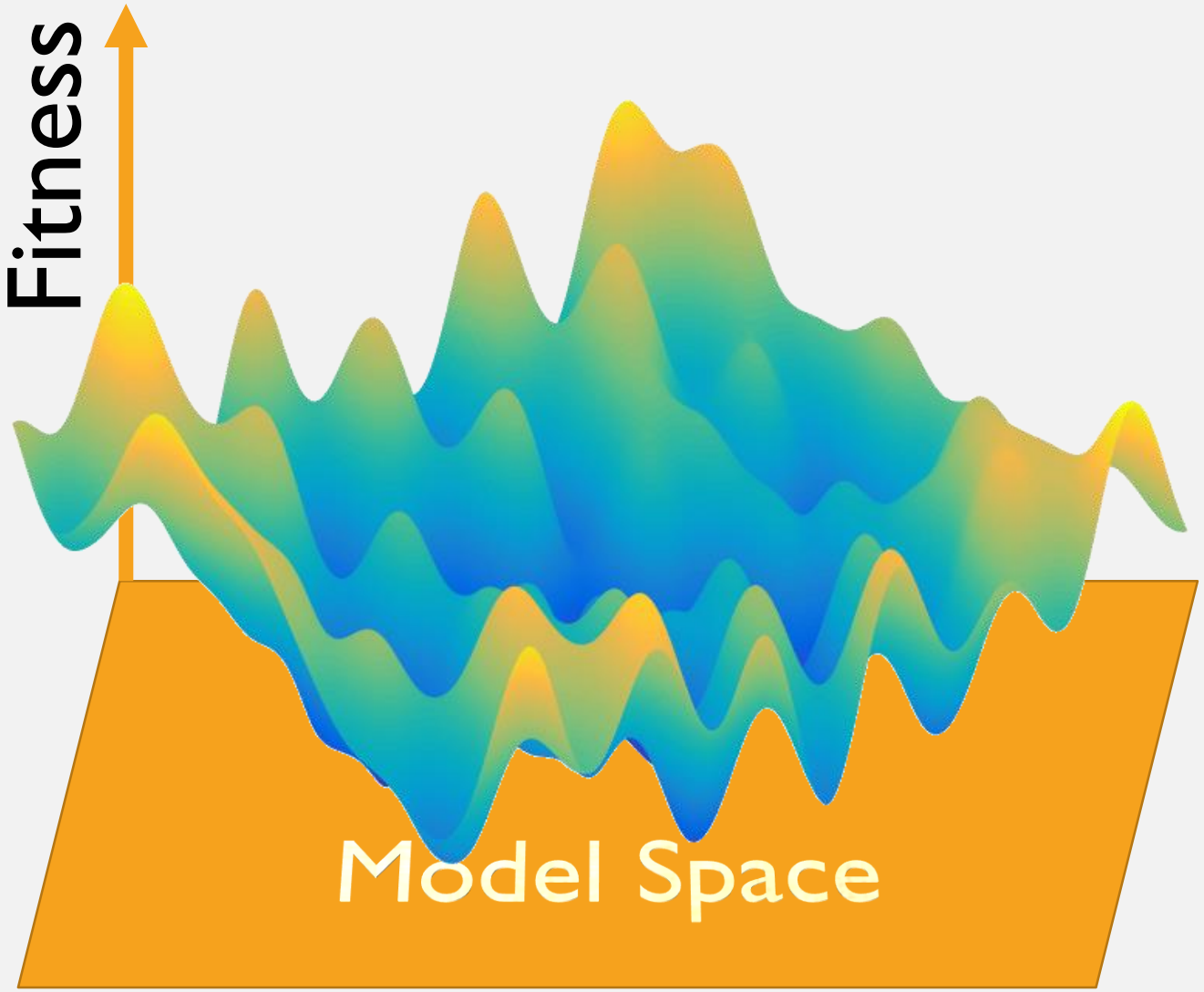


Image: https://users.ece.cmu.edu/~yuejiec/research.html

MOTIVATION

Success in breaking through the optimization barrier allows better modeling of data

Swarm intelligence (SI) methods can prove effective for optimization in statistical analysis

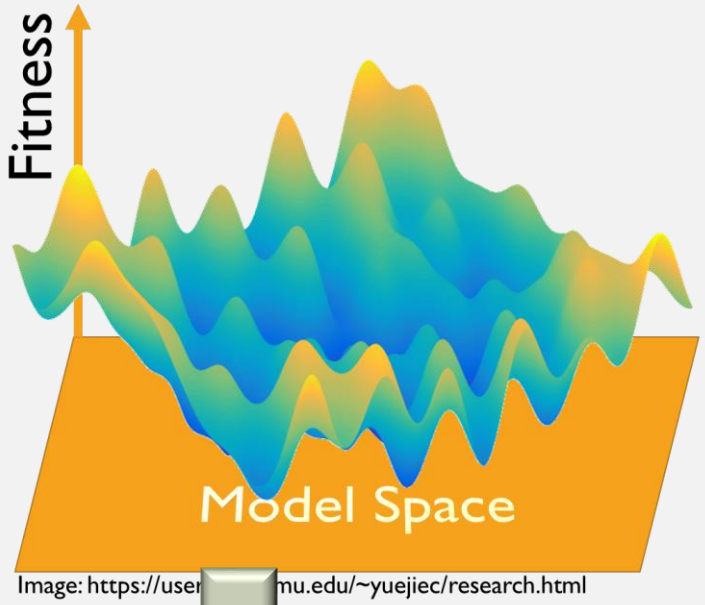
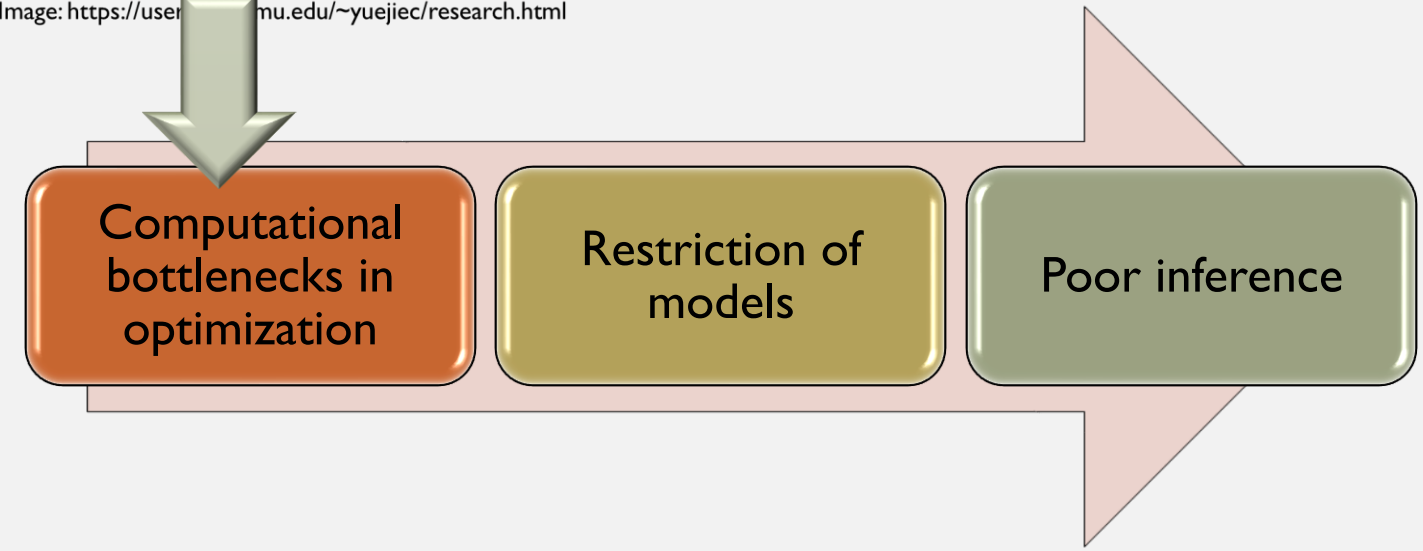


Image: https://user mu.edu/~yuejiec/research.html



LECTURE 2 OUTLINE

Stochastic optimization

- Fundamental results
  - Convergence
  - Exploration and exploitation
  - No free lunch
- Benchmarking and tuning
- Best-of-M-runs strategy for performance improvement

EC and SI methods

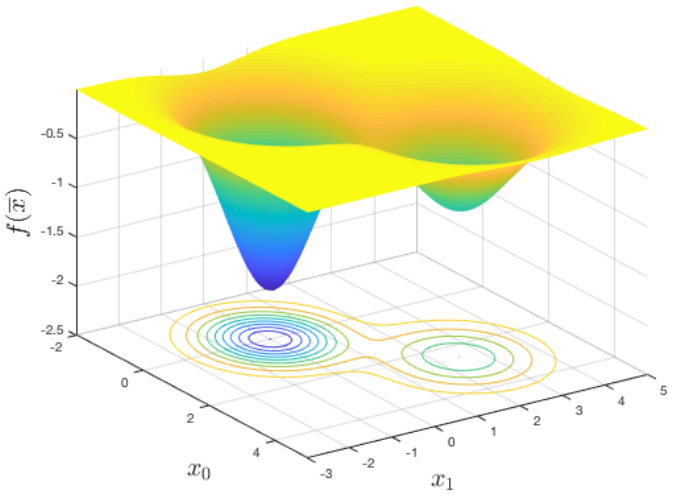
- Review
  - (EC) Evolutionary computation
  - (SI) Swarm intelligence

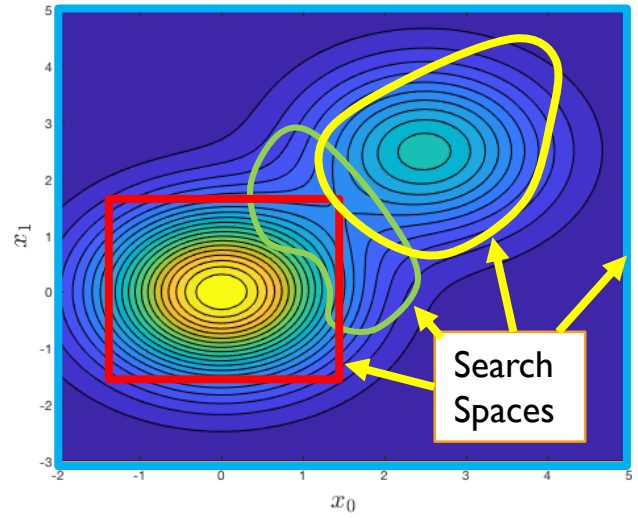
Particle Swarm Optimization

- Basic ($gbest$) PSO
- Kinematics
- Dynamics

BASIC OPTIMIZATION THEORY

Common terminology





OPTIMIZATION: OBJECTIVE

- Continuous optimization problem: Find the minimum value of a function $f(\bar{x})$ in a specified domain $\bar{x} \in \mathbb{D} \subseteq \mathbb{R}^D$
- Minimizer: The location, $\bar{x}^*$, of the minimum

$$f(\bar{x}^*) \leq f(\bar{x}), \bar{x}^* \in \mathbb{D}, \bar{x} \in \mathbb{D}$$

- Alternatively

$$f(\bar{x}^*) = \min_{\bar{x} \in \mathbb{D}} f(\bar{x})$$
$$\bar{x}^* = \operatorname{argmin}_{\bar{x} \in \mathbb{D}} f(\bar{x})$$

- $f(\bar{x})$ is called the fitness function (also objective function)
- $\mathbb{D}$ is called the search space (or constraint set)
- *Maximization of $f(\bar{x})$ is equivalent to minimization of $-f(\bar{x})$

GLOBAL AND LOCAL MINIMIZER

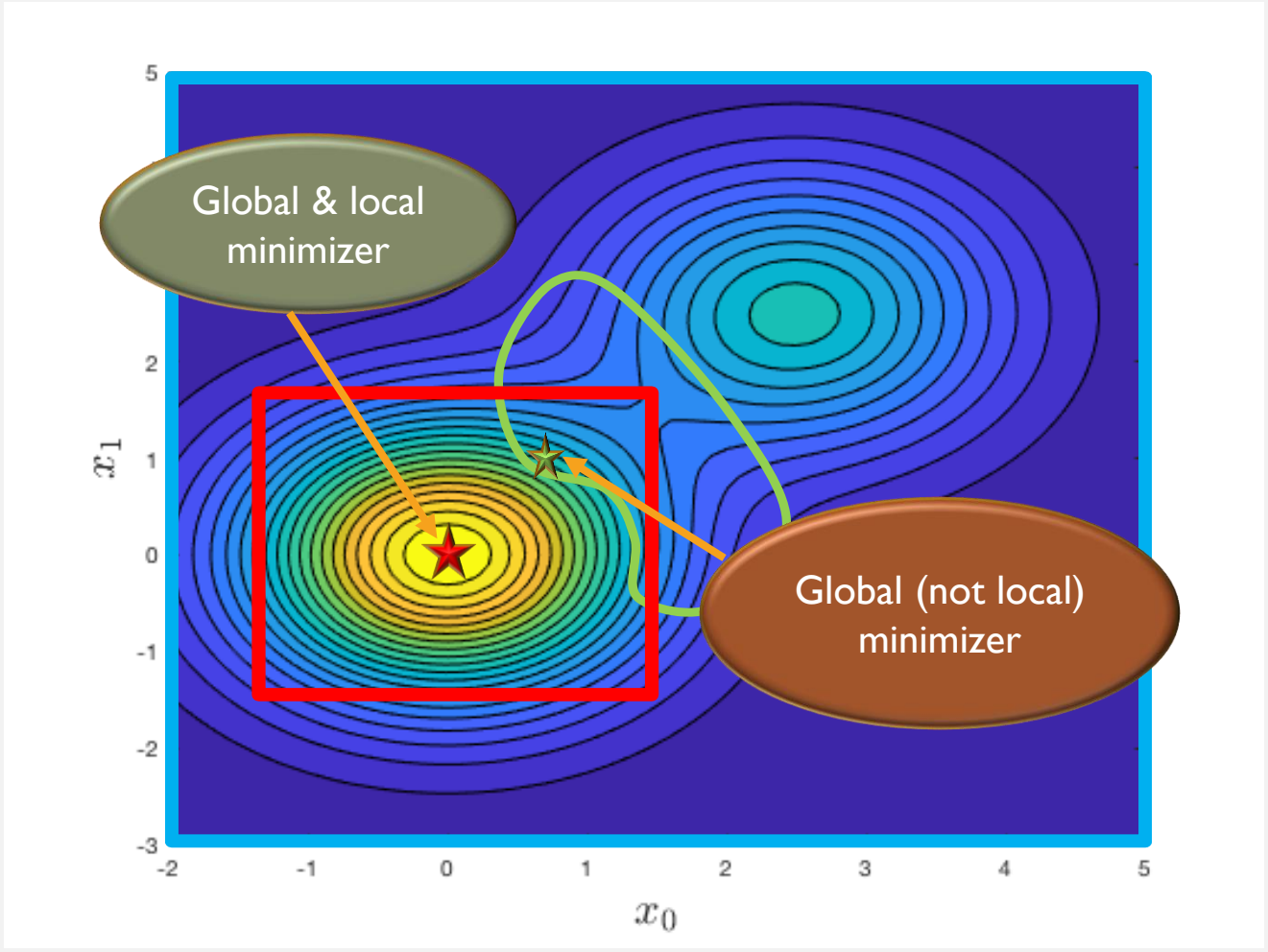
- Global minimizer:

$$f(\bar{x}^*) \leq f(\bar{x}), \bar{x}^* \in \mathbb{D}, \bar{x} \in \mathbb{D}$$

- Local minimizer: For differentiable functions, the gradient vanishes

$$\left.\frac{\partial f}{\partial x_i}\right|_{\bar{x}^*} = 0, \forall i$$

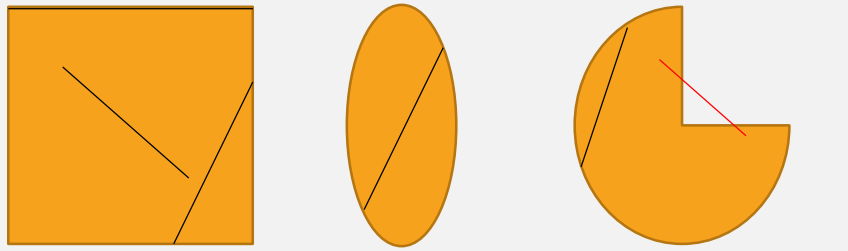
- Regression requires global minimizer of cost function



CONVEX SET AND FUNCTION

CONVEX SET

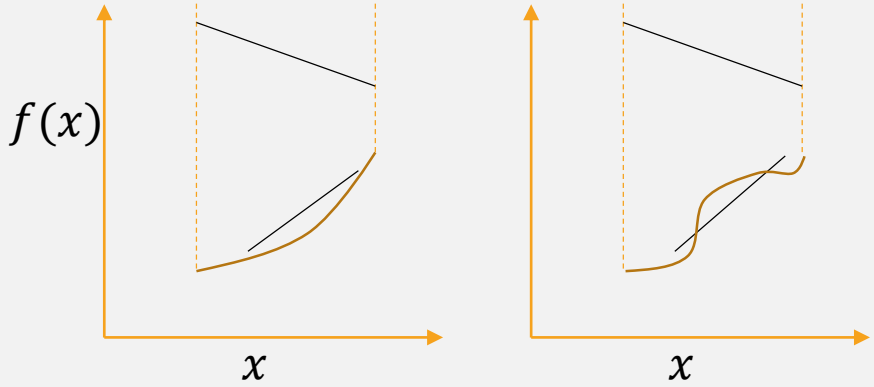
(Volume in $\mathbb{R}^D$) All the points along a straight line joining any two points in the set also belong to the set



Convex | Convex | Non-convex

CONVEX FUNCTION

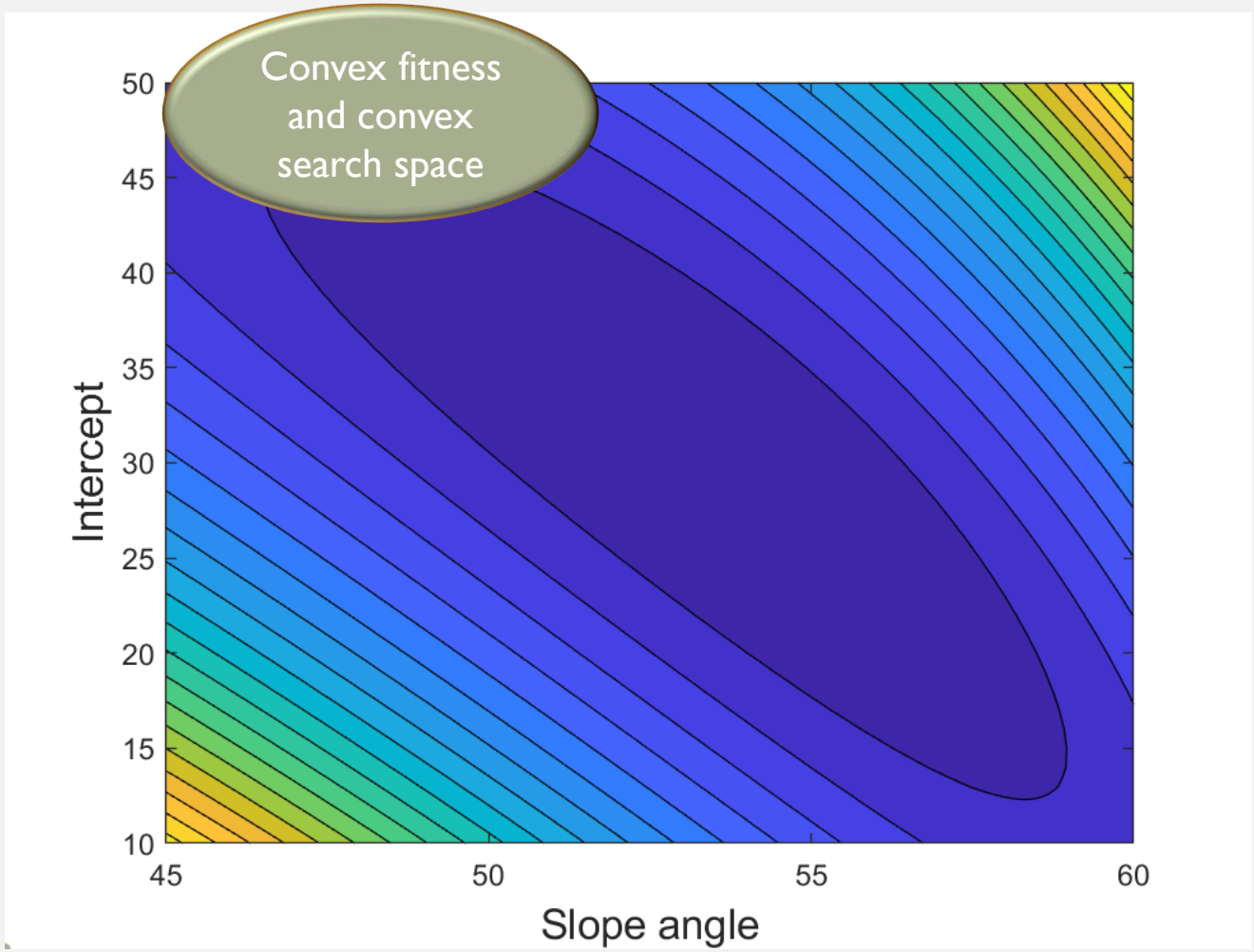
(Volume in $\mathbb{R}^{D+1}$) The "volume" bounded by the function is a convex set



Convex | Non-convex

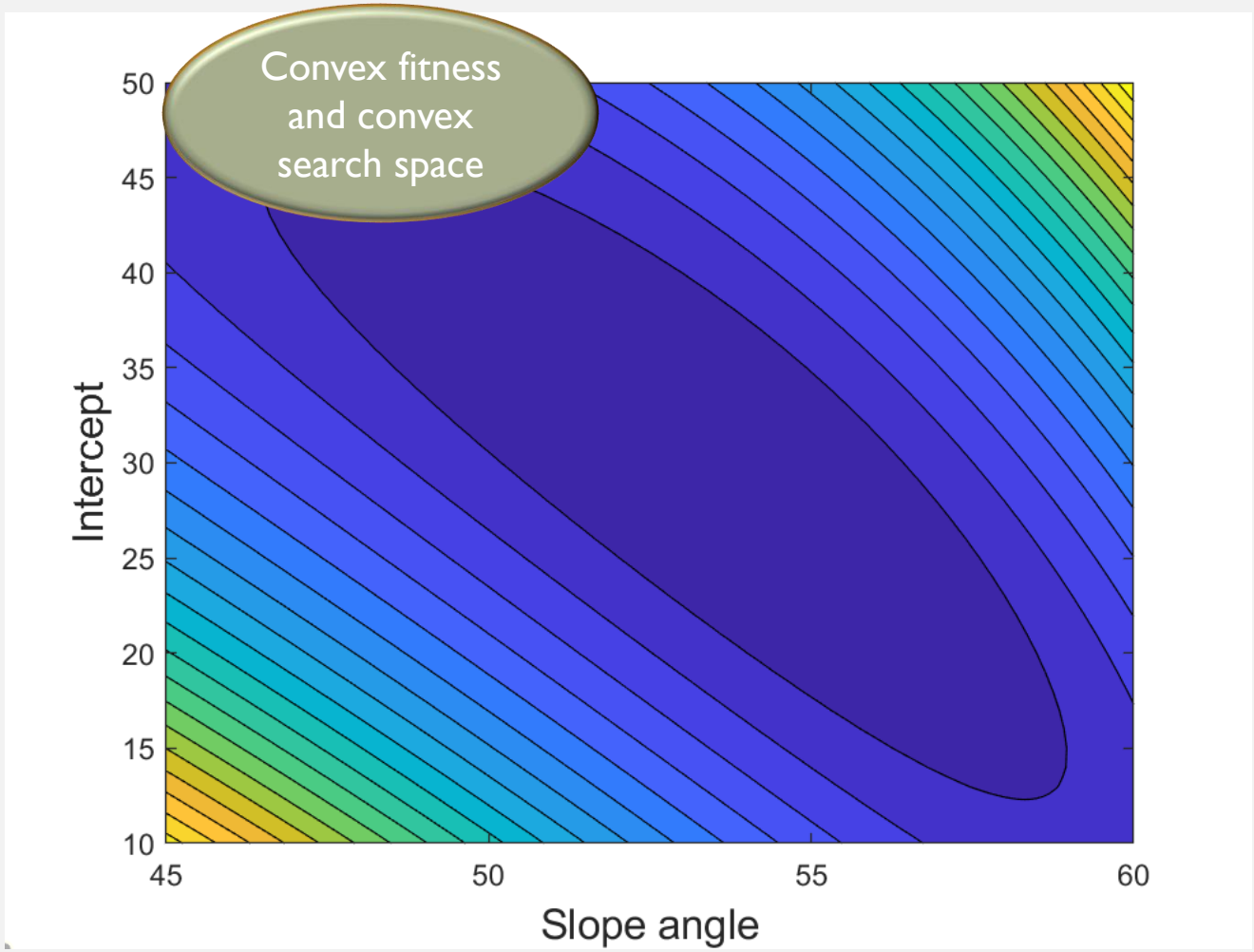
CONVEX OPTIMIZATION

- Convex search space and convex fitness ⇒ Global optimization is "easy"
  - Unique global minimum
  - If local minimum exists, it is the global minimum



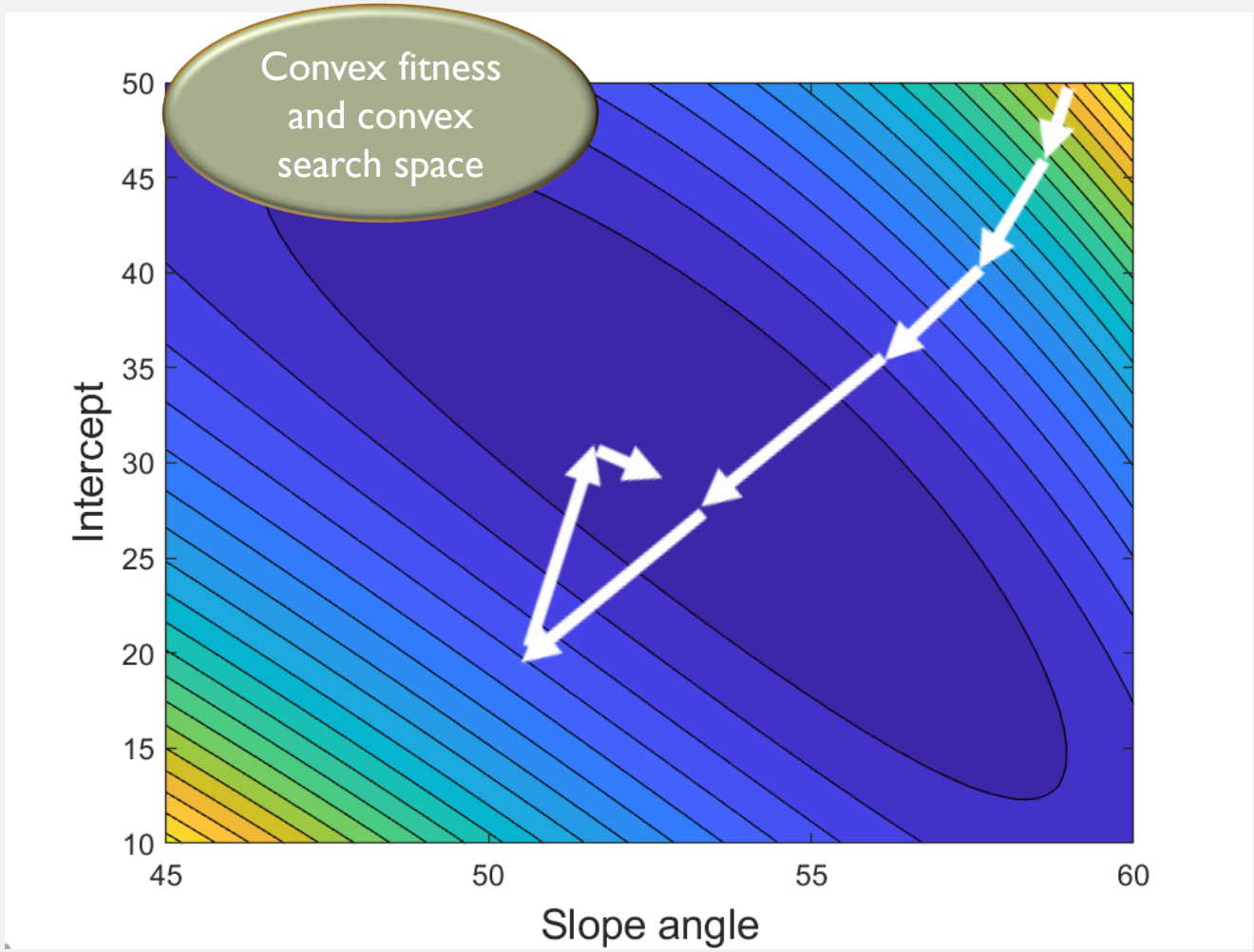
CONVEX OPTIMIZATION

- Convex search space and convex fitness ⇒ Global optimization is "easy"
- Local optimizations methods serve as global optimization methods



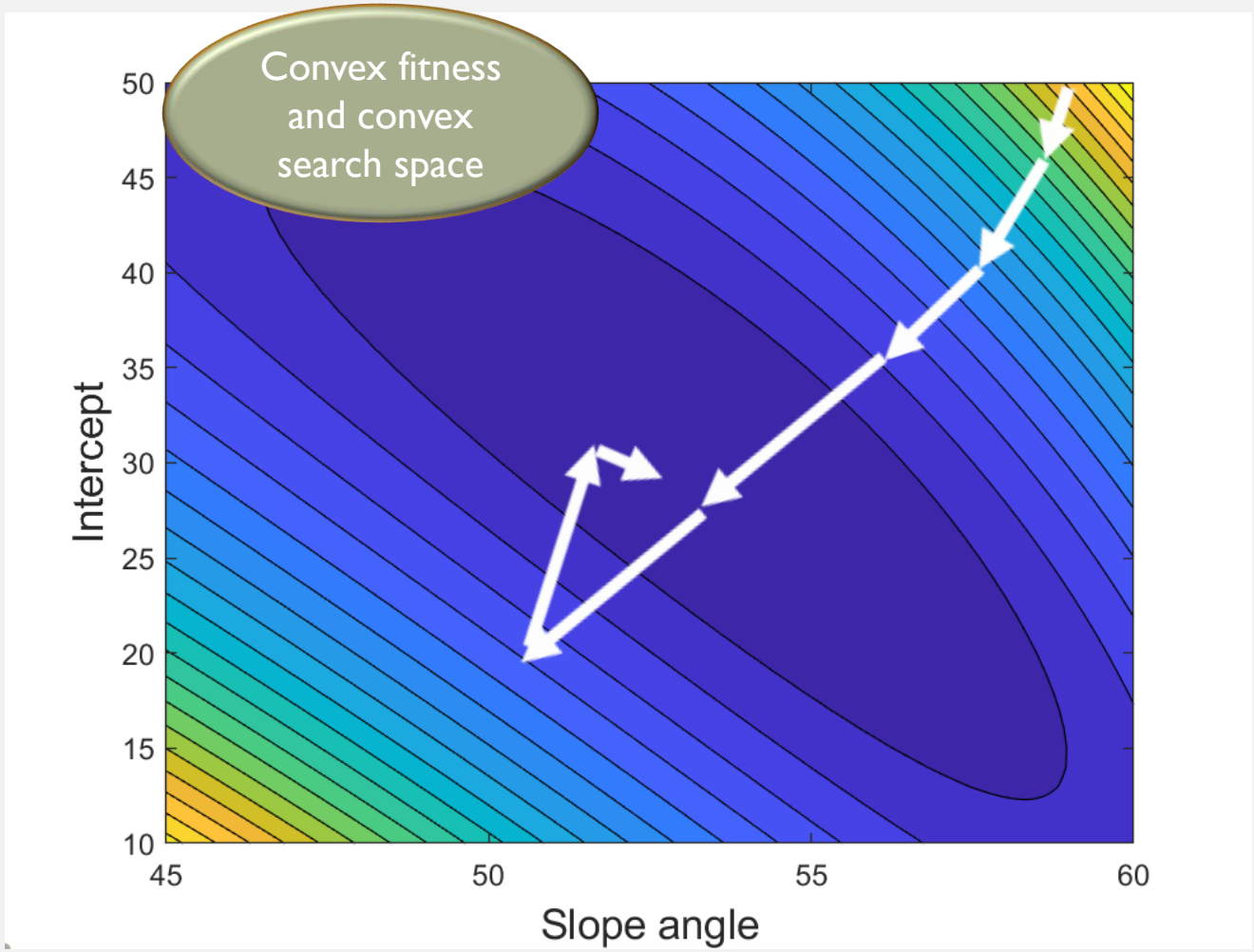
LOCAL OPTIMIZATION METHODS

- Greedy method: Always seek a lower fitness value
- Example: Method of steepest descent
  - Follow the direction of the negative gradient at each point



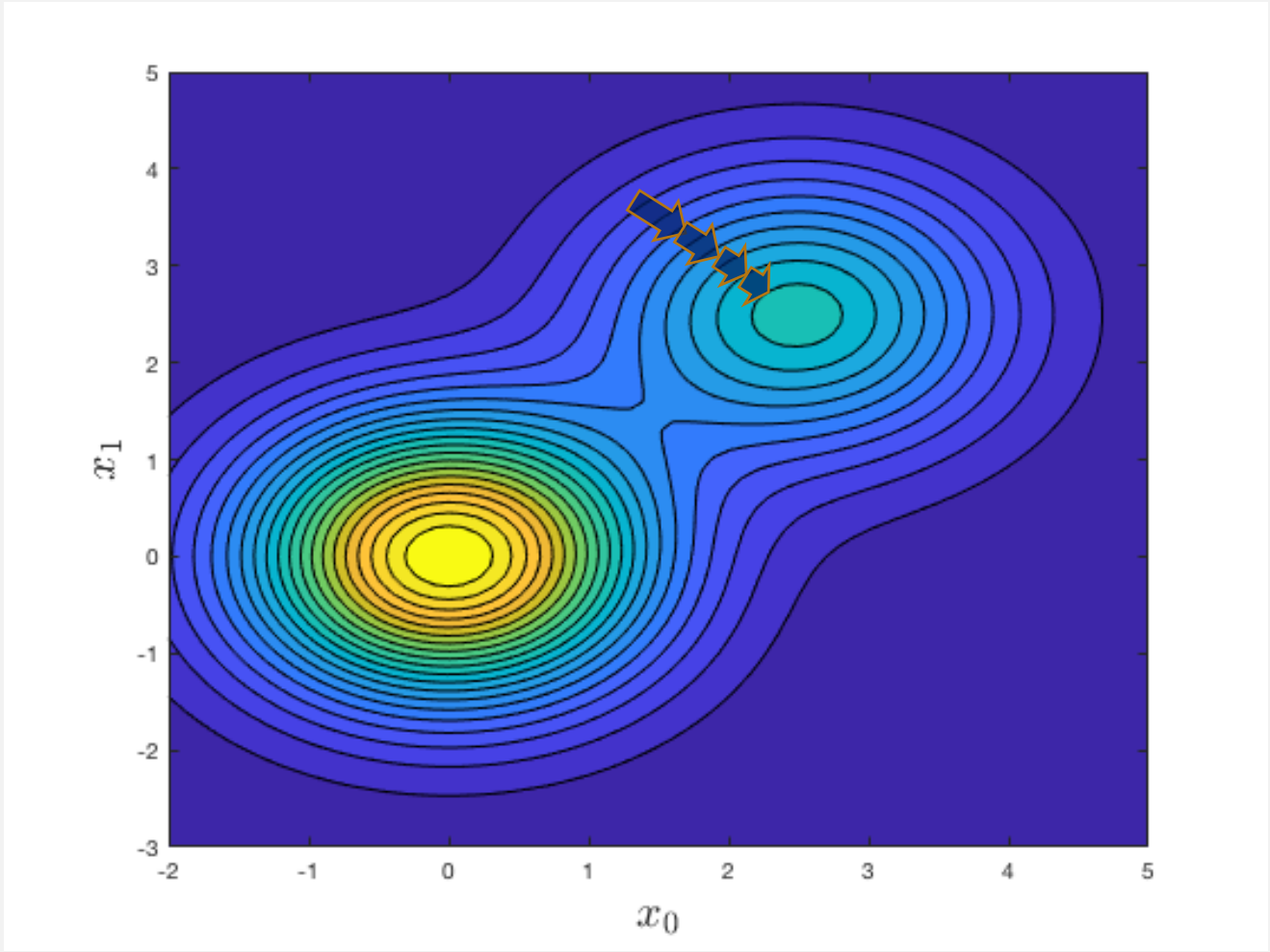
LOCAL OPTIMIZATION METHODS

- Greedy method: Always seek a lower fitness value
- Guaranteed to converge to a local minimum
  - Deterministic: Same initial point leads to same final point
  - *Caution: oversimplification of convergence analysis



NON-CONVEX OPTIMIZATION

- Non-convex fitness and/or non-convex search space
  - Example: Multiple local minima in the search space (Multi-modal fitness function)
- Local optimization methods are not guaranteed to find the global minimum



OPTIMIZATION IN STATISTICAL REGRESSION

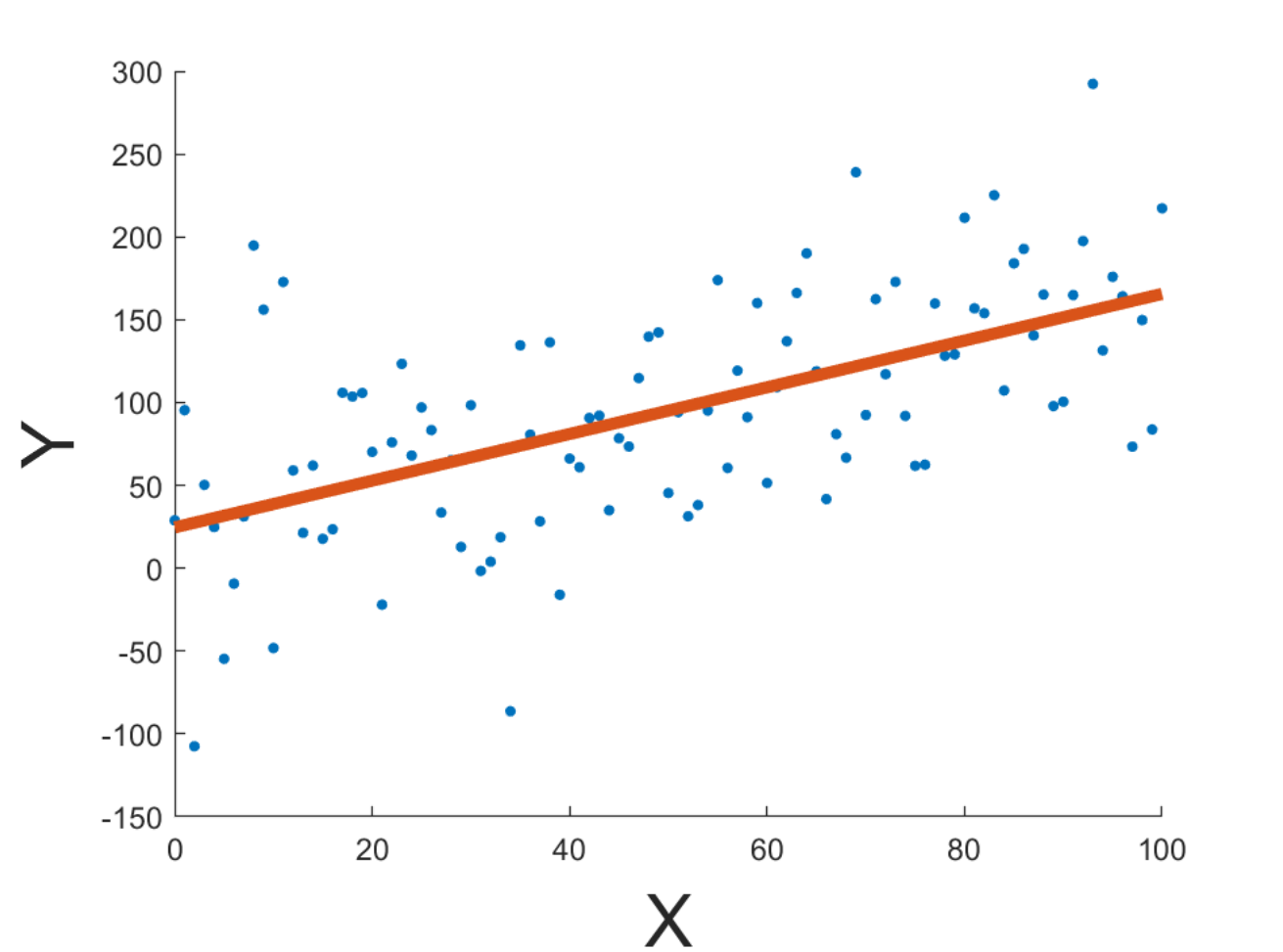
LEAST SQUARES: LINEAR MODEL

- Example: $f(X)$ belongs to the family of straight lines

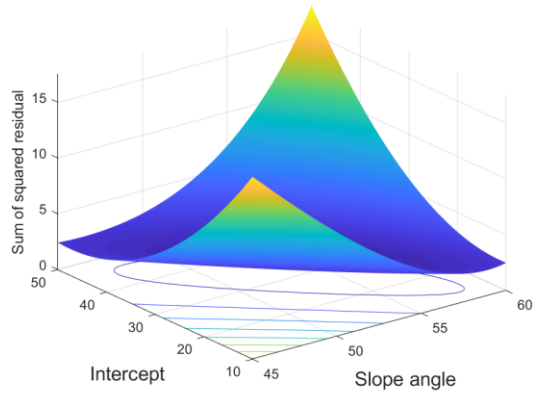
$$f(X) = aX + b$$

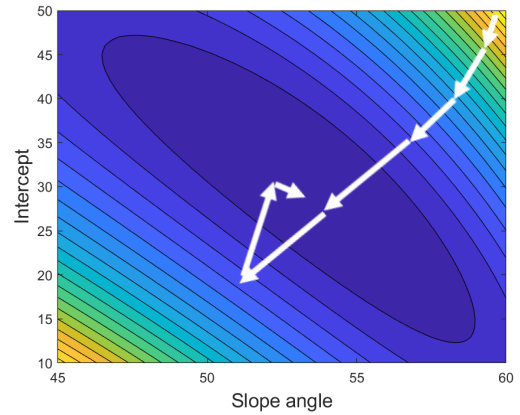
Least-squares fit

$$\min_{a,b} \sum_{i=0}^{N-1} (y_i - ax_i - b)^2$$



Straight line fit: Sum of squared residuals





LEAST SQUARES: LINEAR MODEL

- Unconstrained search: Convex optimization problem
- Local minimum = global minimum ⇒ easy (in principle) optimization problem

LEAST SQUARES: NON-LINEAR MODEL

- Example: Quadratic chirp

$$f\left(x;\bar{\theta}\right) = A\sin(2\pi\Phi(x))$$

Instantaneous phase:

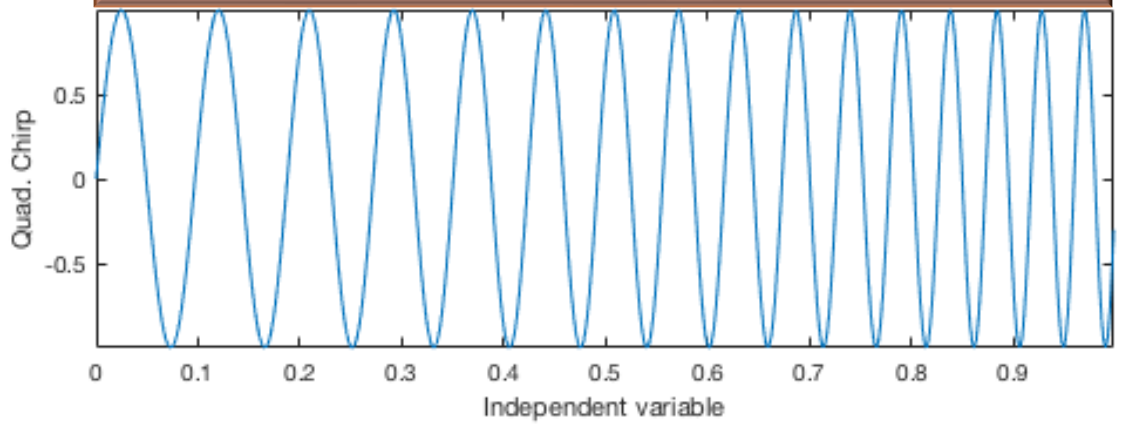
$$\Phi(x) = a_1 x + a_2 x^2 + a_3 x^3$$

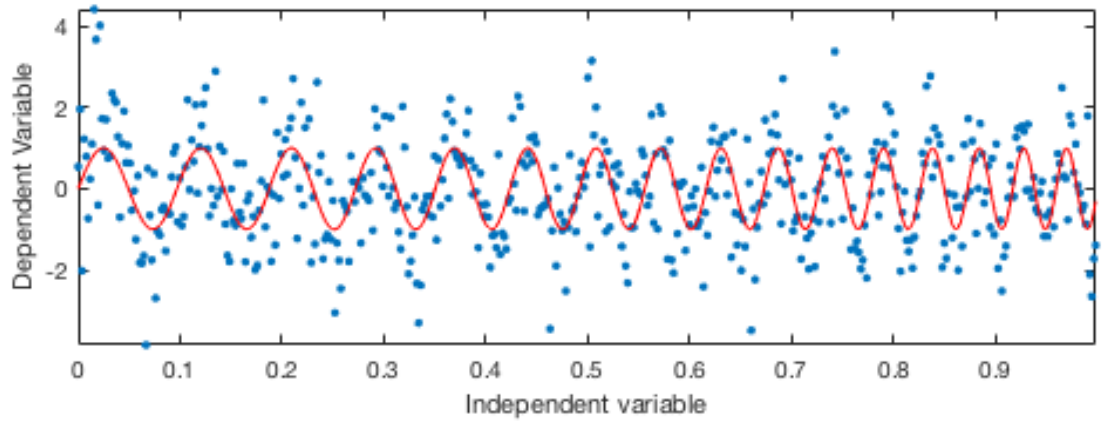
Instantaneous frequency:

$$f(x) = \frac{d\Phi}{dx} = a_1 + 2a_2 x + 3a_3 x^2$$

(We can think of x as time t)

$f(x)$ increases with x
$1/f(x)$ (Instantaneous period) decreases with x

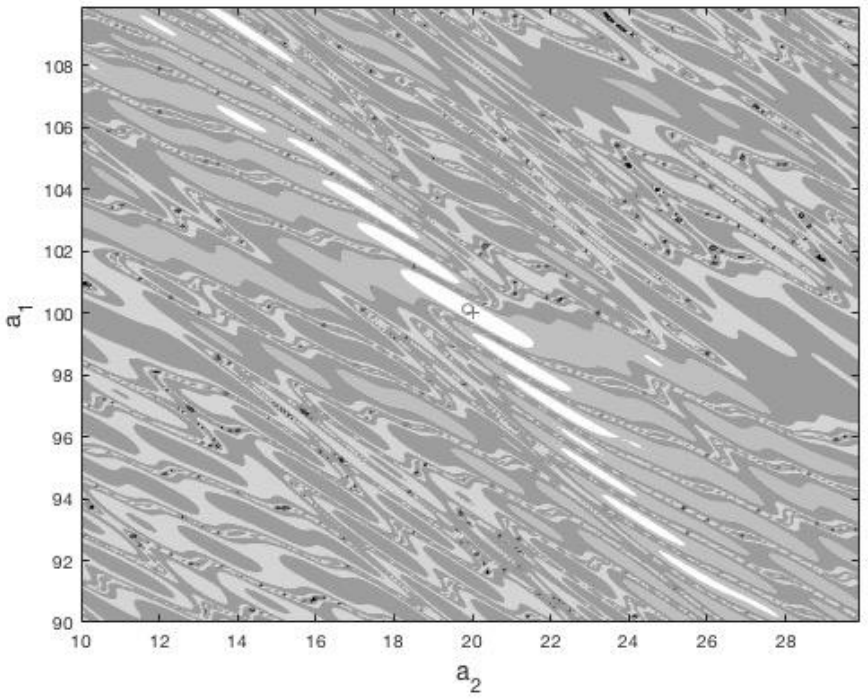




Data realization

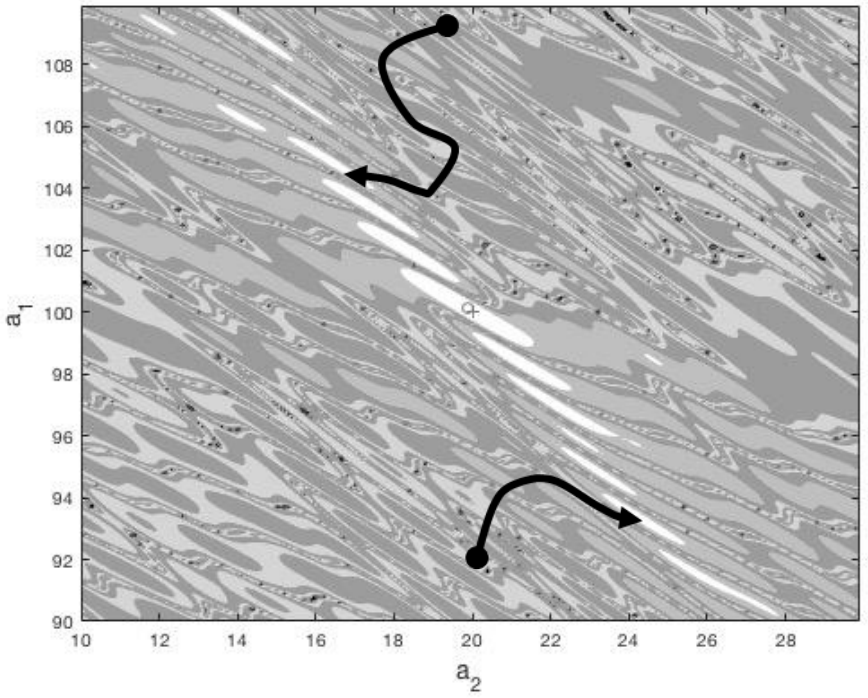
LEAST SQUARES: NON-LINEAR MODEL

- Regression with non-linear models typically leads to Multi-modal fitness functions ⇒ Non-convex optimization problems



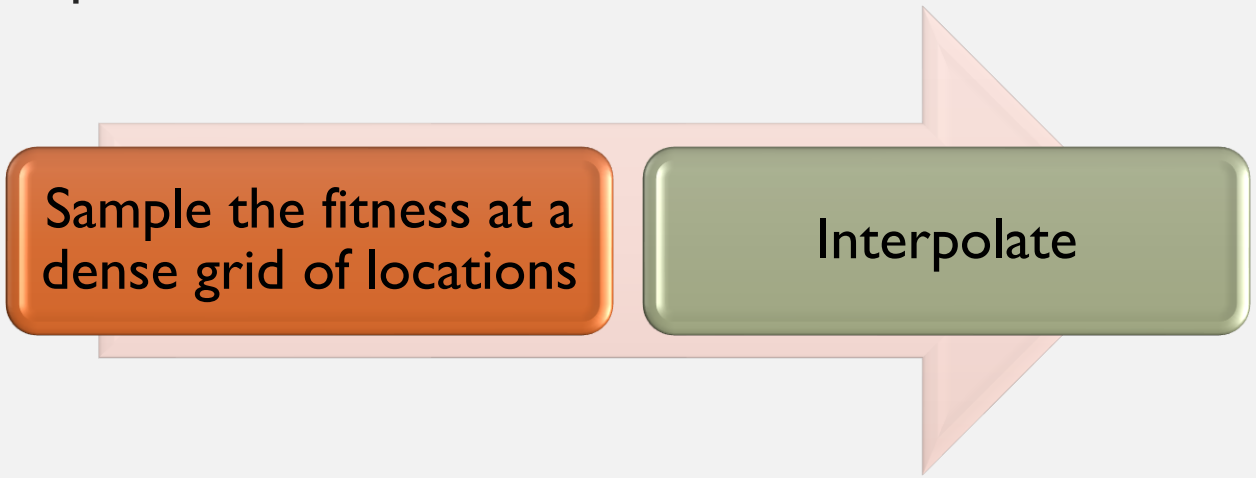
NON-CONVEX OPTIMIZATION

- Local optimization methods are not viable for non-linear regression
- High dimensional search space ⇒ the difficulty of global optimization increases manyfold



DETERMINISTIC NON-CONVEX OPTIMIZATION

- Grid-based optimization: The only guaranteed deterministic global optimization method for non-convex fitness functions



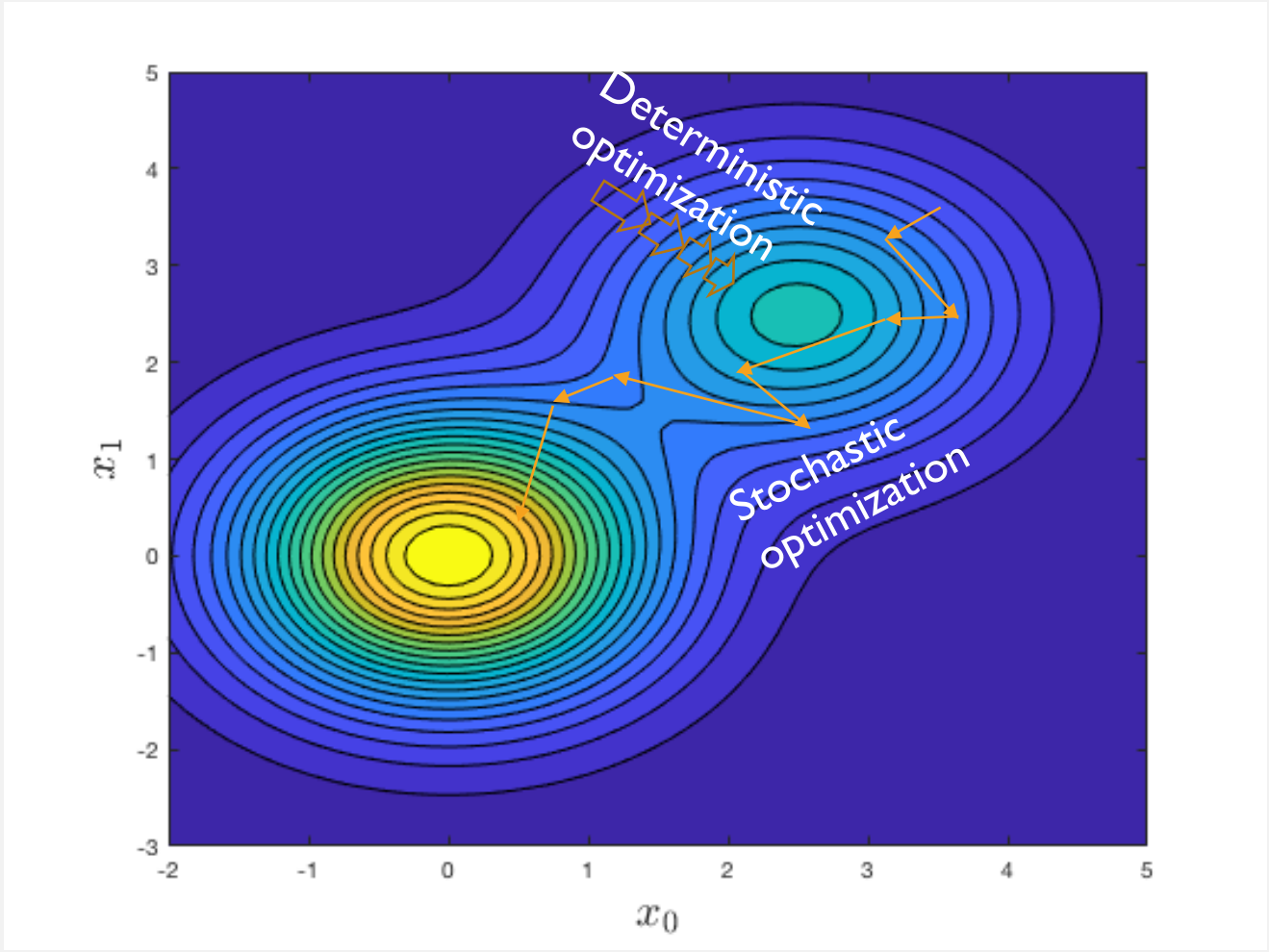
- Exponential growth in the number of grid points
  - N_G^D where N_G is the number of grid points per dimension and D is the number of dimensions

STOCHASTIC OPTIMIZATION

General formulation

STOCHASTIC OPTIMIZATION

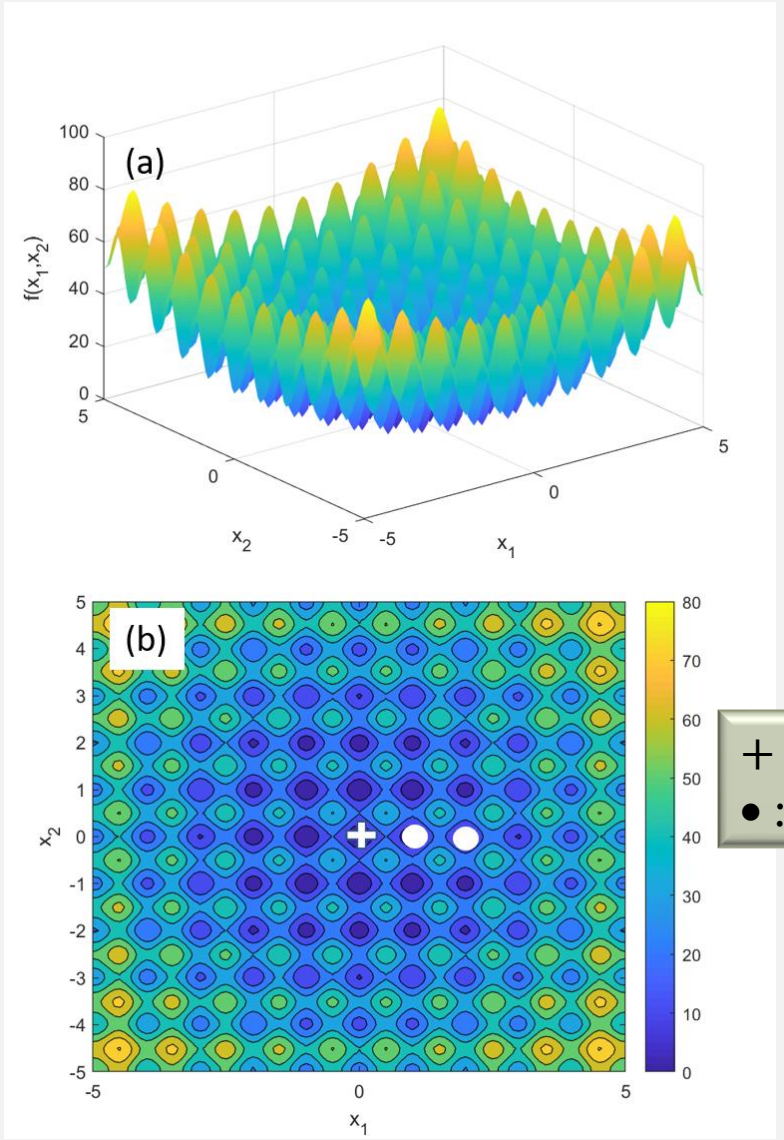
- Escape local minimum by random moves in search space
- Avoid completely random motion by imposing some goal or guidance



BENCHMARK FITNESS FUNCTIONS

- Non-convex optimization is studied using a standard set of fitness functions that can be evaluated rapidly
- Example: Generalized Rastrigin function

$$f(\bar{x}) = \sum_{i=0}^{D-1} \left[x_i^2 - 10\cos(2\pi x_i) + 10 \right]$$

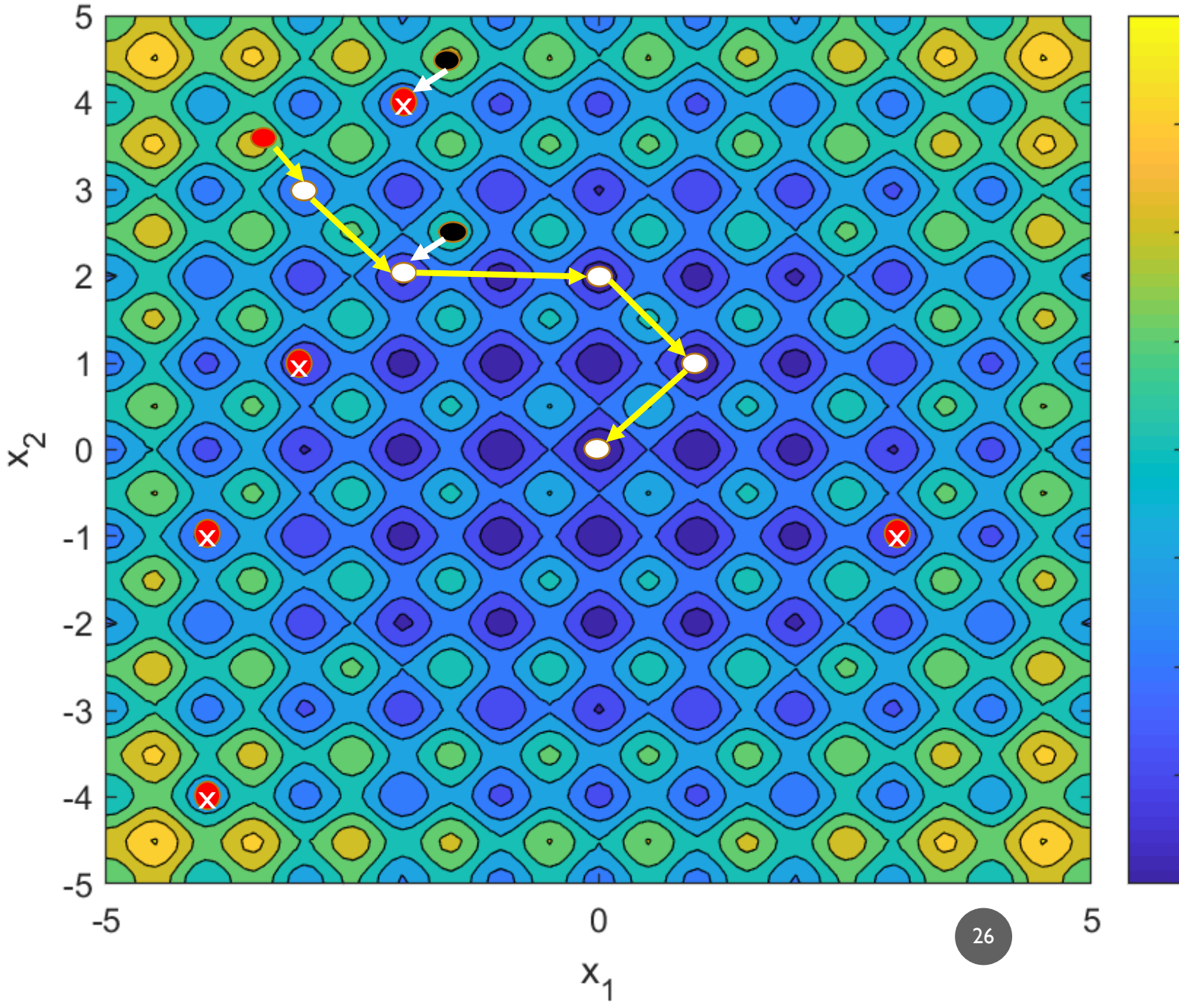


+ : Global minimum
• : Local minimum

STOCHASTIC OPTIMIZATION EXAMPLE

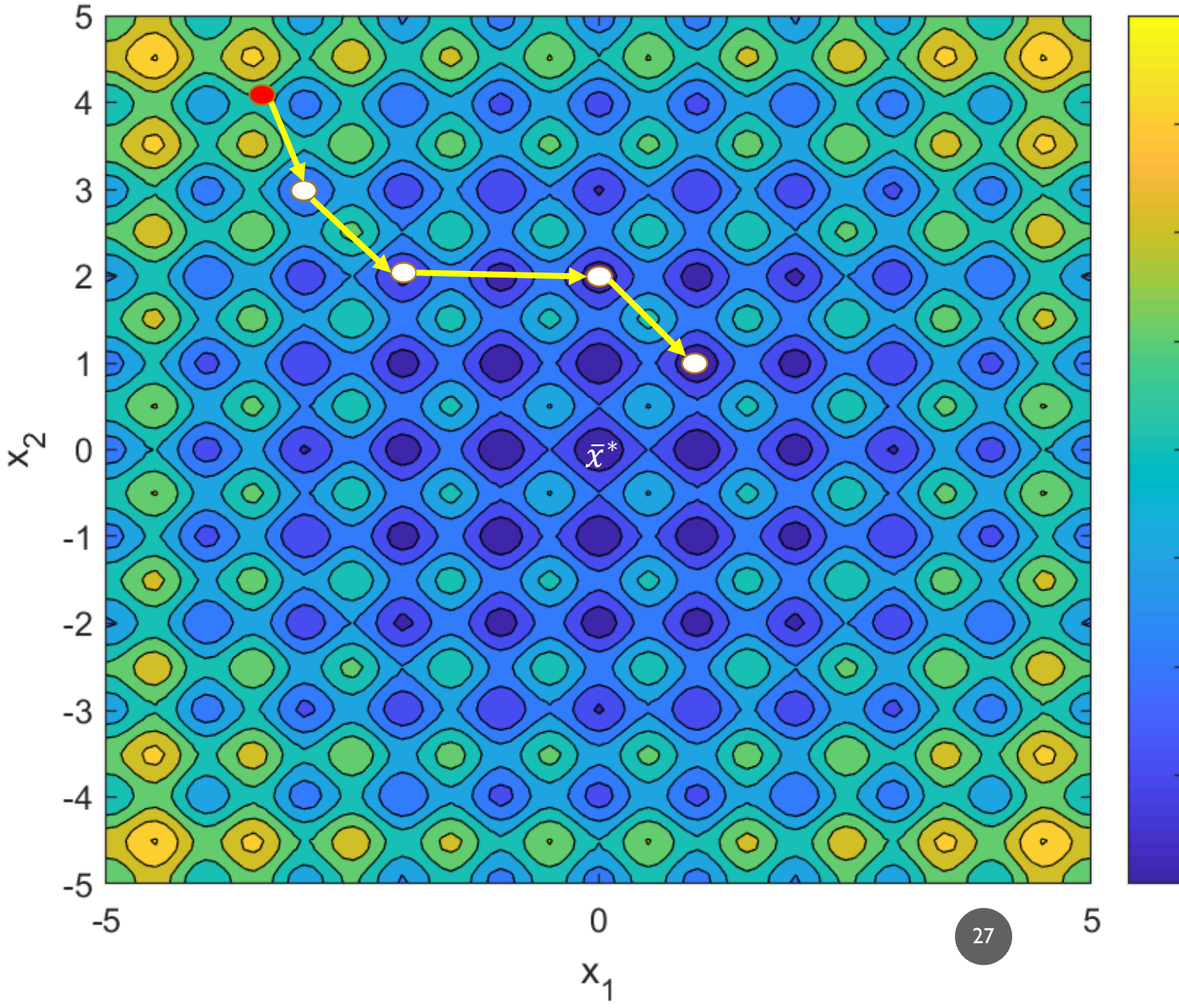
- Solis and Wets (1981): General form of stochastic optimization methods
- "Algorithm 3":

1. Initialization: Start from a random point in search space
2. Repeat …
   1. Randomization: Pick a new point randomly; Find the nearest local minimizer
   2. Position update: Move to this point if it is a better local minimum
3. Until termination condition is true



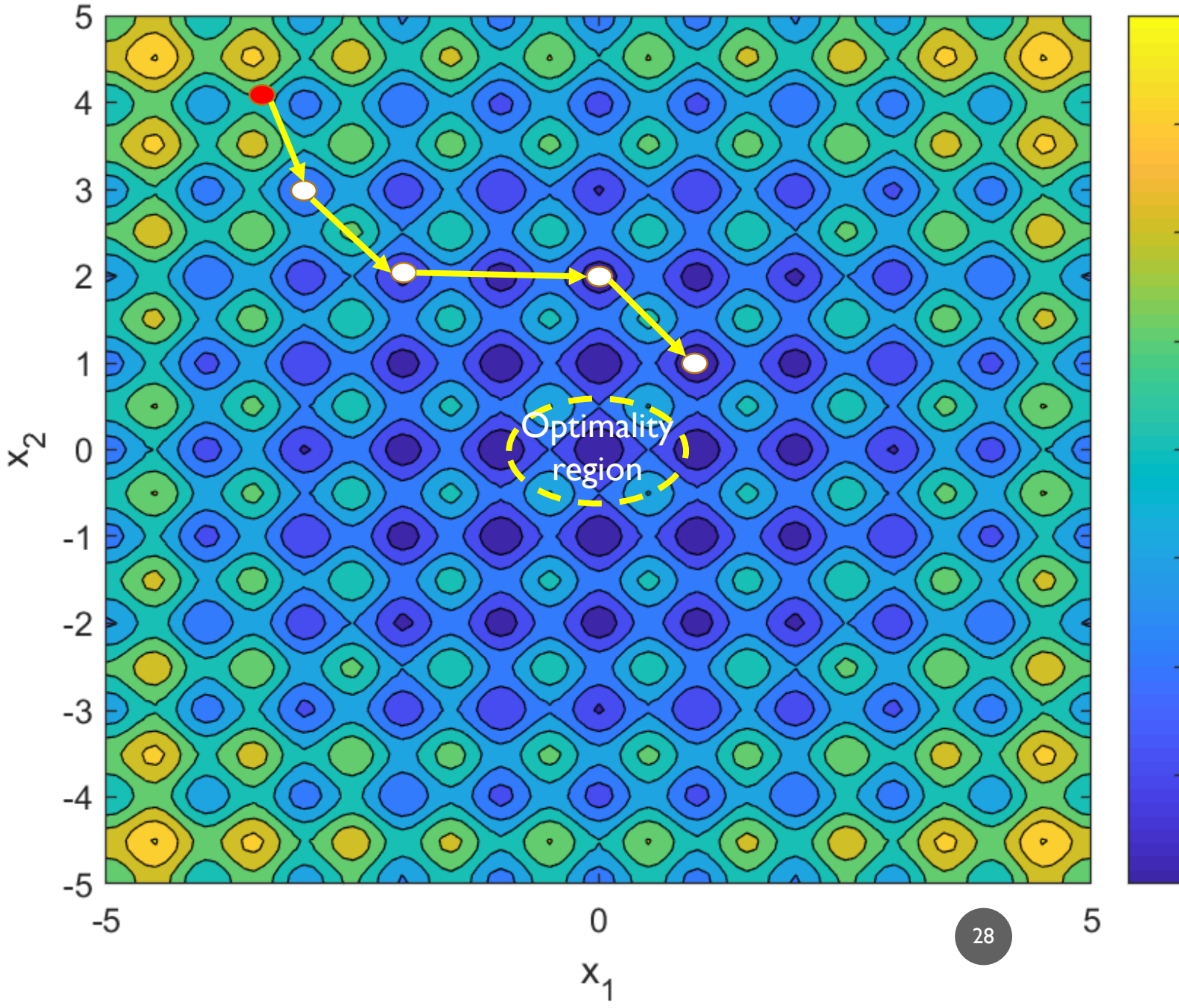
PERFORMANCE

- Stochastic ⇒ unpredictability
- Stochastic optimization ⇒ unpredictable terminal location
- Depends on the interaction of the random moves with the fitness values



CONVERGENCE

- Termination at exactly $\bar{x}^*$ is not a viable criterion
- Convergence must be measured probabilistically
- ⇒ Measure probability of terminating in a region around $\bar{x}^*$



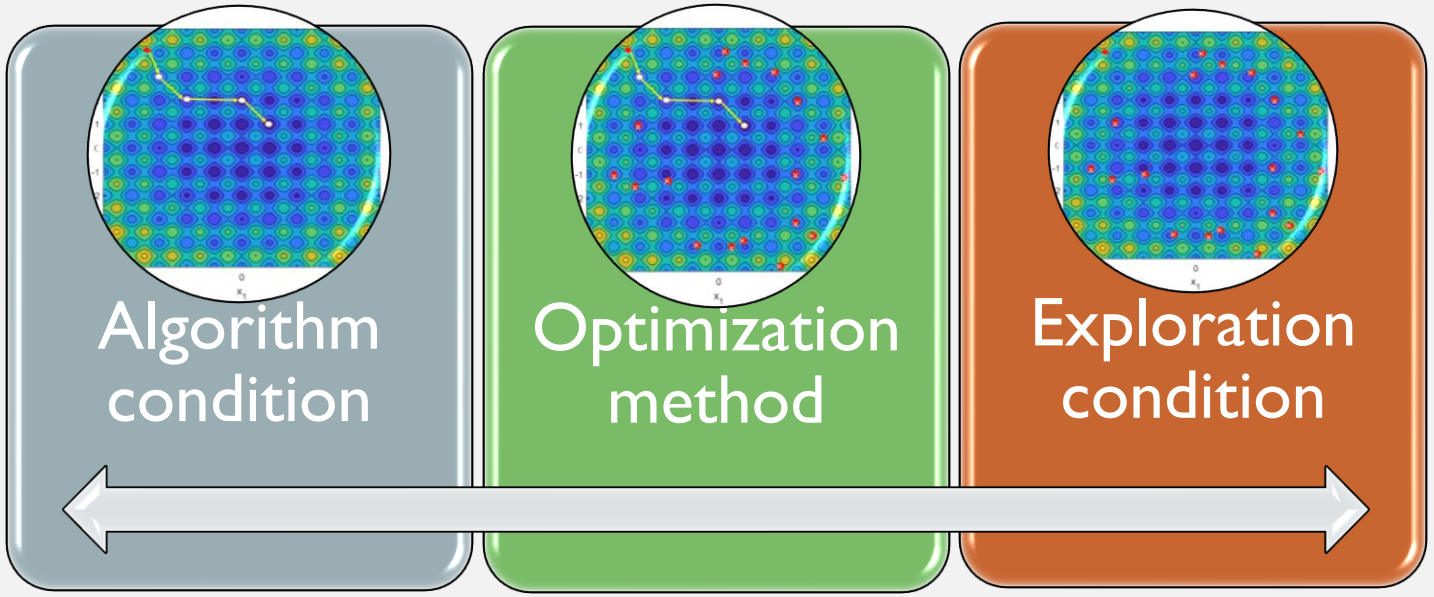
CONVERGENCE CONDITIONS

- Convergence criterion for stochastic optimization :

$$\mathrm{Lim}_{k\to\infty}\ \Pr(\bar{x}[k] \in \text{Optimality region}) = 1$$

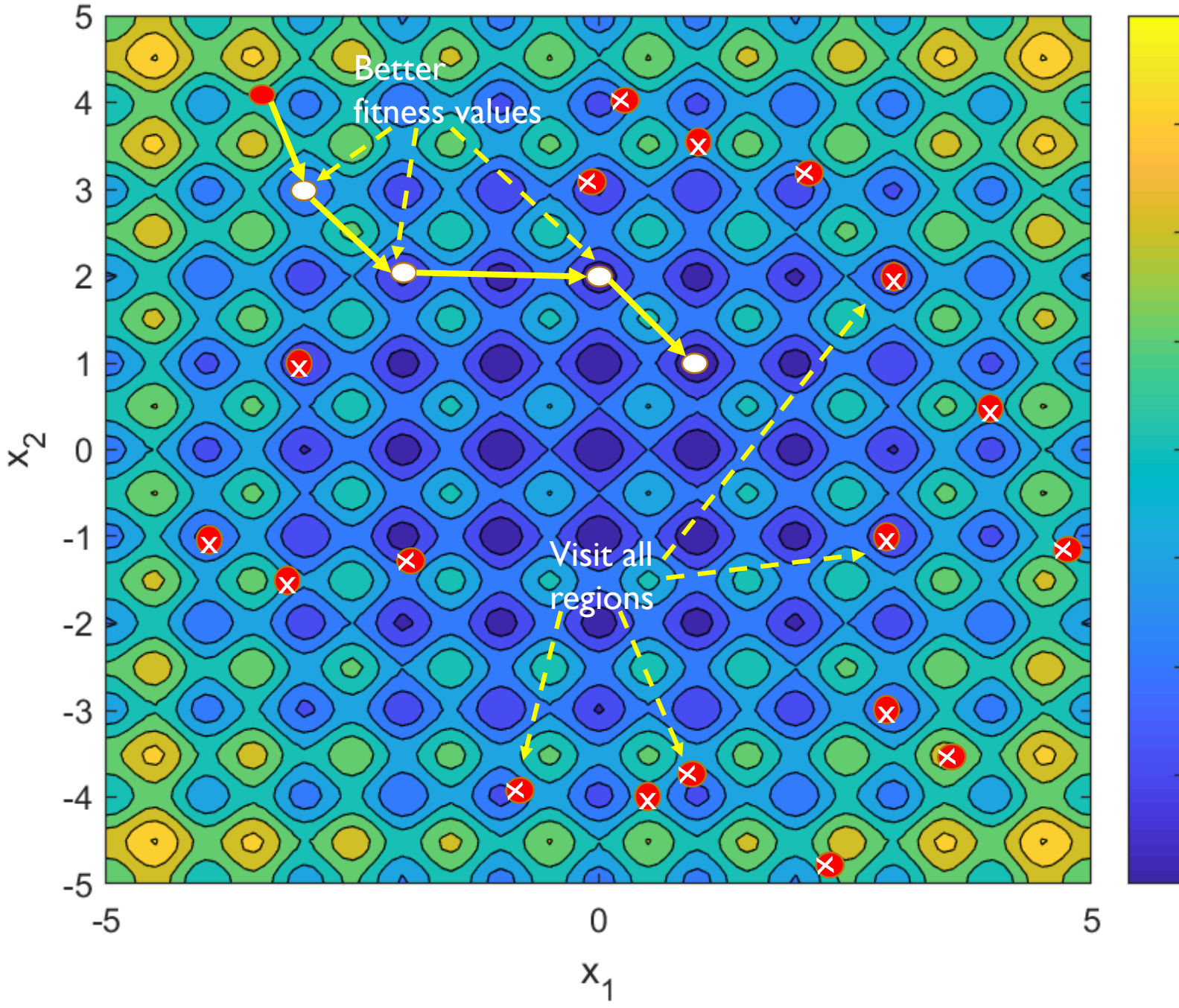
- Sufficient conditions (Solis and Wets, 1981):
  - Algorithm condition (H1): The fitness values always improves as the algorithm iterations progress
  - Exploration condition (H2): There is no region of the search space that is not visited at least once

CONVERGENCE CONDITIONS



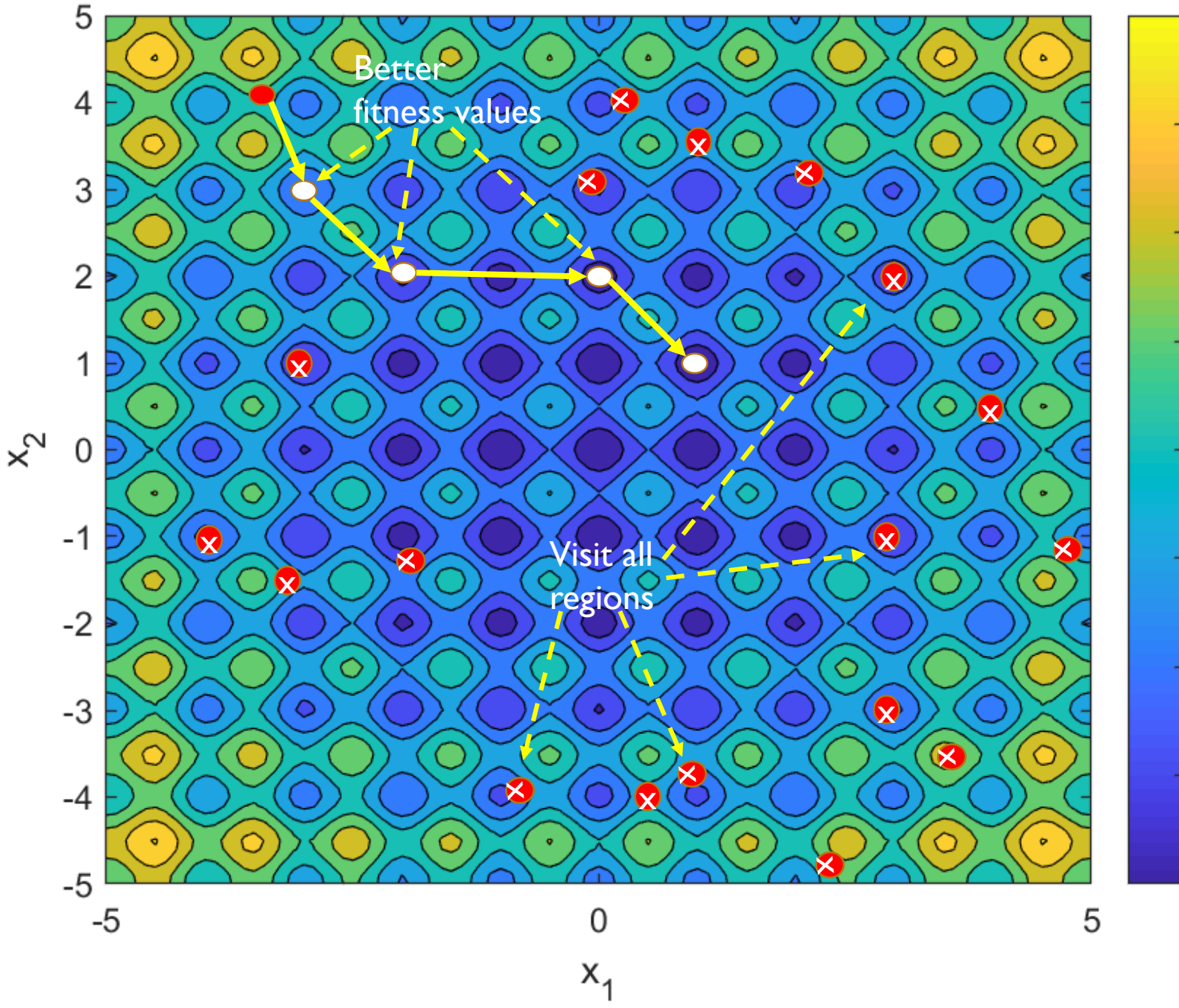
The sufficiency conditions pull a global optimization method in opposite directions

Algorithm 3 of Solis and Wets provably satisfies the convergence conditions

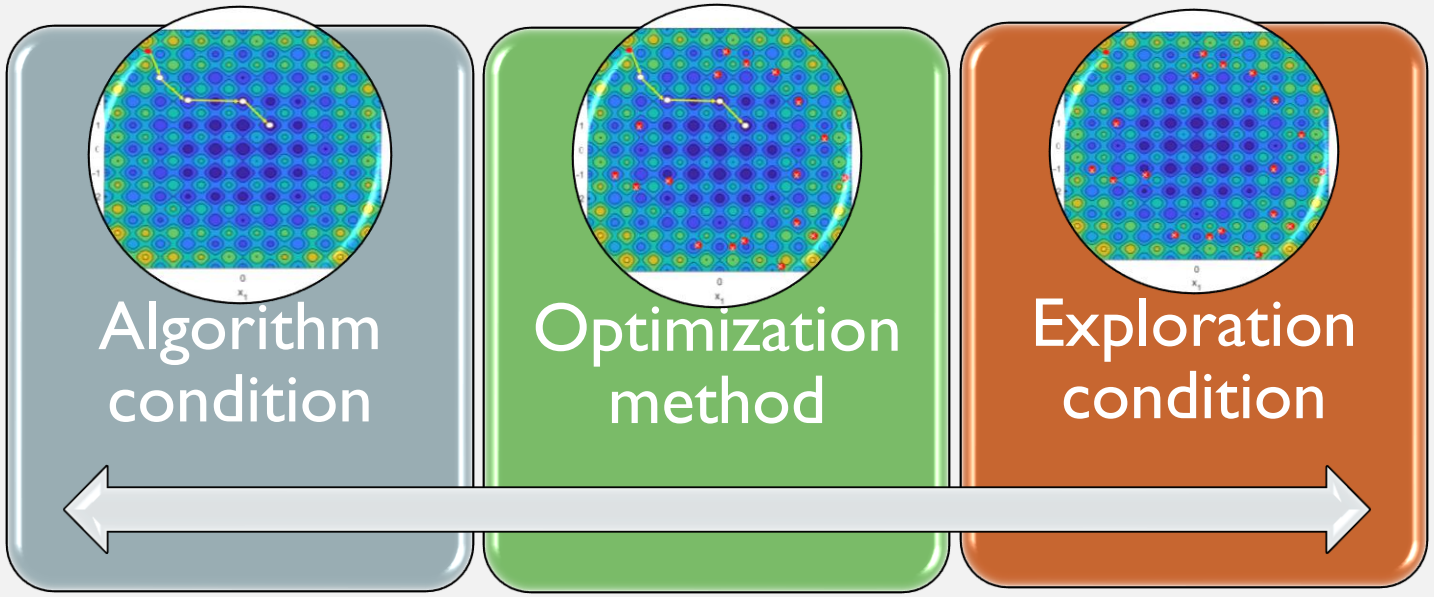


Asymptotic convergence does not quantify the rate of convergence

Finite number of iterations ⇒ Convergence not guaranteed



CONVERGENCE CONDITIONS

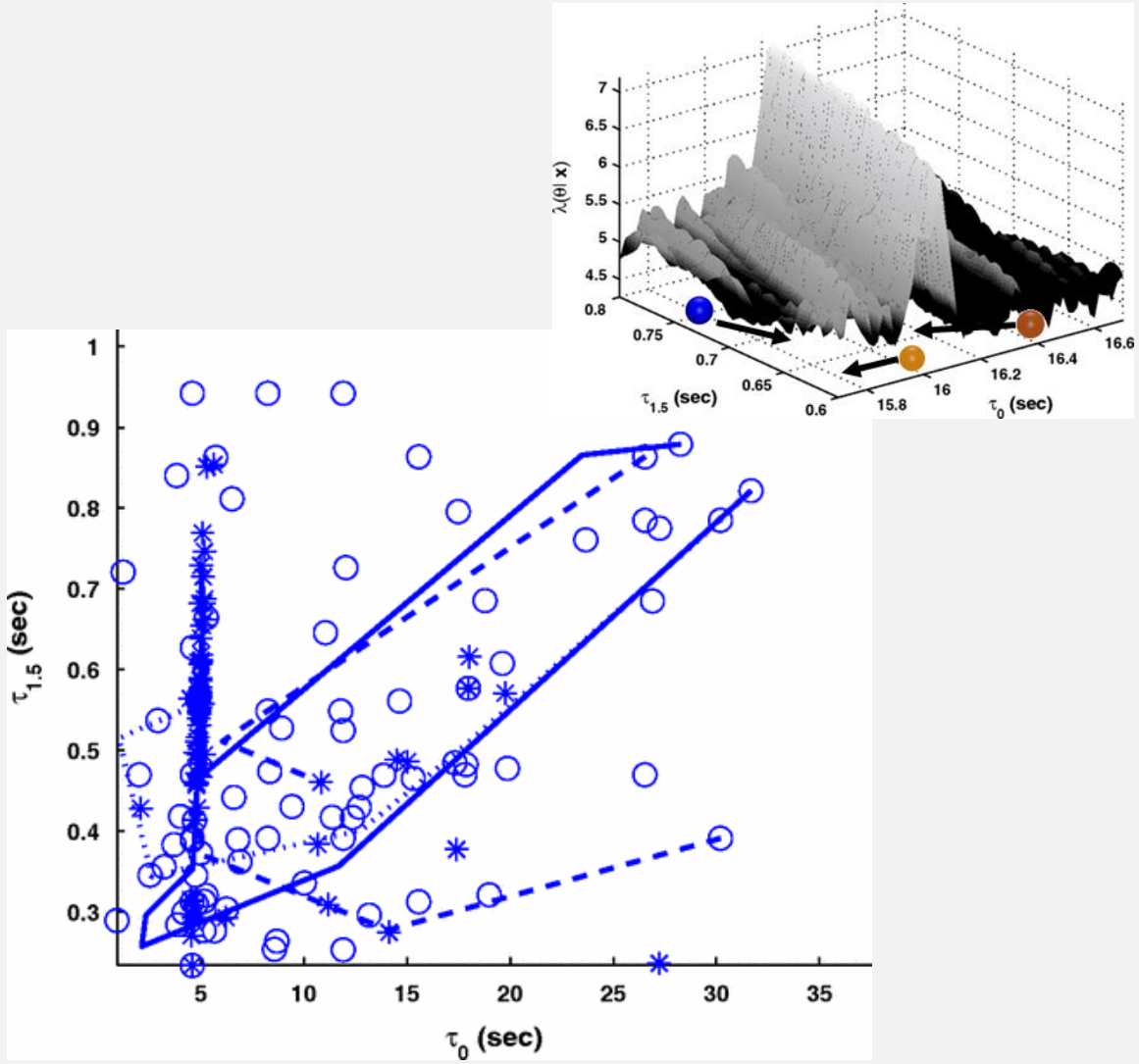


- The sufficiency conditions pull a global optimization method in opposite directions
- Finite number of iterations ⇒ Full exploration of search space not possible while trying to converge to a better values

EXPLORATION AND EXPLOITATION

- The opposite pulls of the sufficiency conditions ⇒ practical stochastic optimization methods have two phases
- Exploration: explore the search space
- Exploitation: Identify a promising region and focus on improving the fitness

❖Fitness must improve in both phases

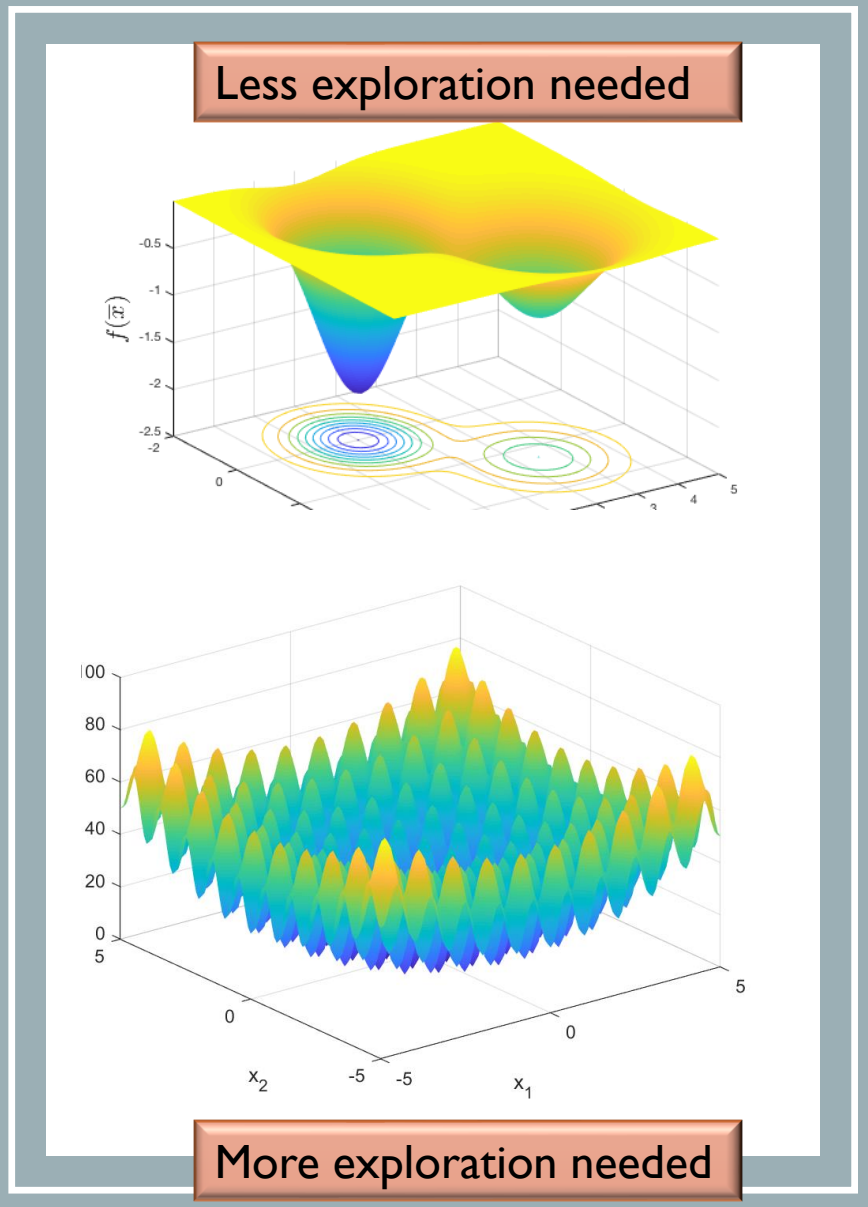


Lecture 1: From Wang, Mohanty, Physical Review D (2010)

EXPLORATION AND EXPLOITATION

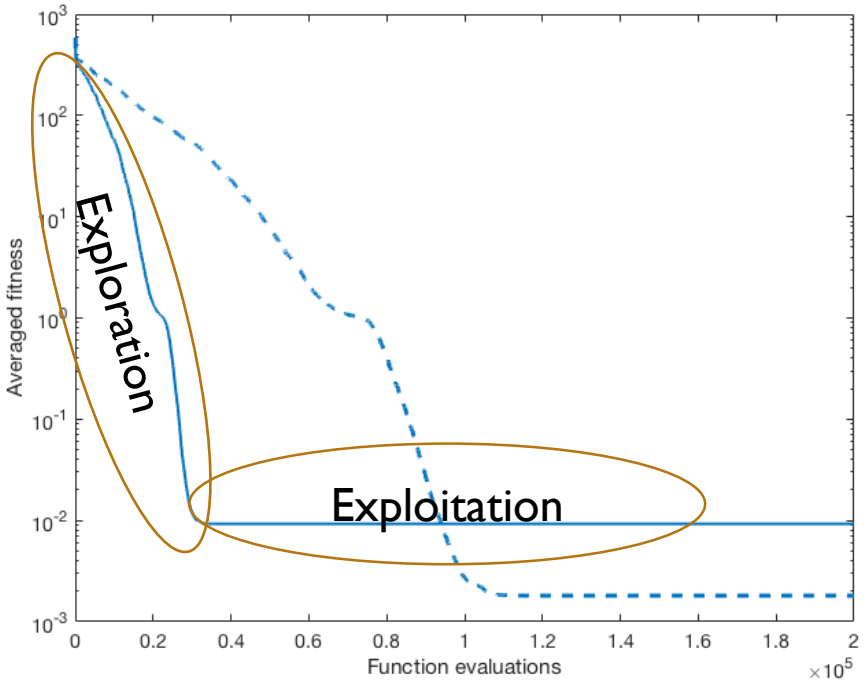
EXPLORATION-EXPLOITATION TRADE-OFF

- Exploration and exploitation oppose each other: Judicious trade-off required
- Higher fitness function ruggedness ⇒ More iterations spent in exploration
- Expensive fitness evaluation & limited computing resources ⇒ rapid exploitation
- Trade-off also depends on the dimensionality of the search space



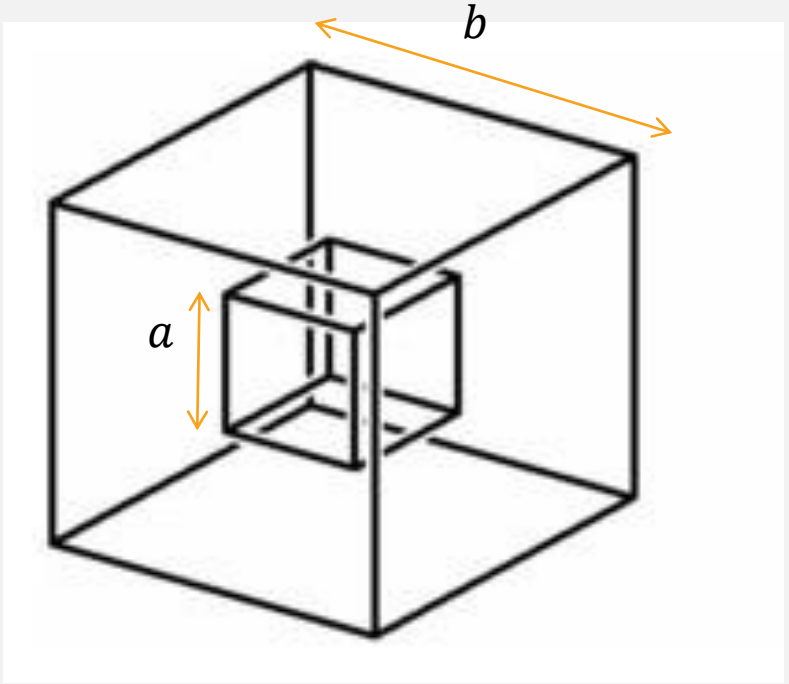
EXPLORATION-EXPLOITATION TRADE-OFF

- 30-dimensional generalized Griewank function
- Fitness value averaged over independent trials for two different stochastic optimization methods.
- PSO: gbest and lbest variants (*Lecture 3)



CURSE OF DIMENSIONALITY

- Non-intuitive behavior of optimization methods with change in dimensionality of search space
  - Example: As dimensionality grows, a larger fraction of volume occurs near the boundaries of the search space
  - ⇒ Randomization from a "uniform" distribution will put a higher fraction of points near the boundary as dimensionality increases
- The relative performance of methods can change with search space dimensionality



- Volume exterior to the inner hypercube

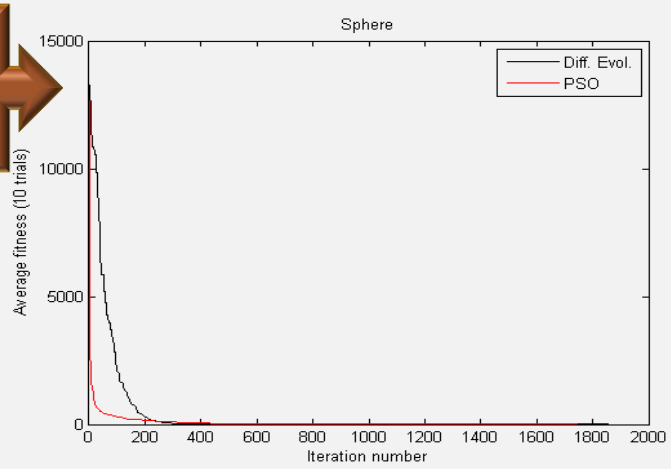
$$b^n - a^n$$

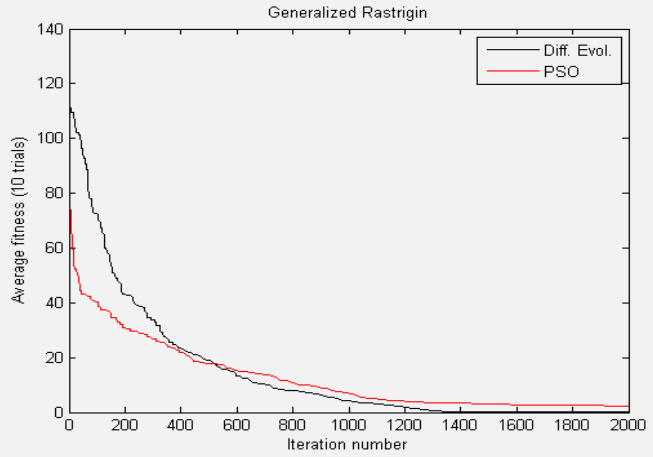
- Relative to the total volume

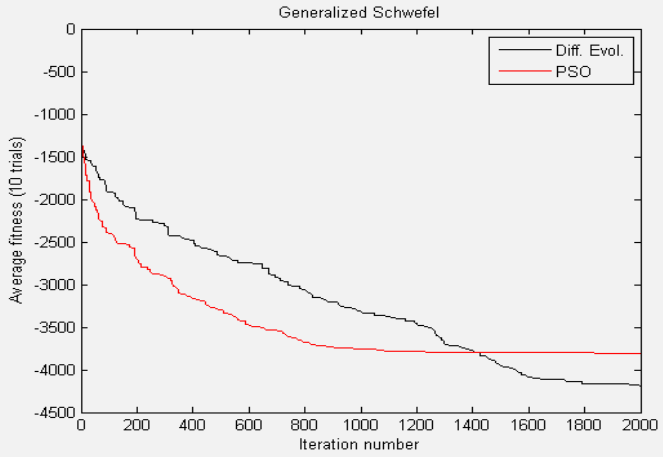
$$1 - \left(\frac{a}{b}\right)^n$$

DIFFERENTIAL EVOLUTION VS. PSO

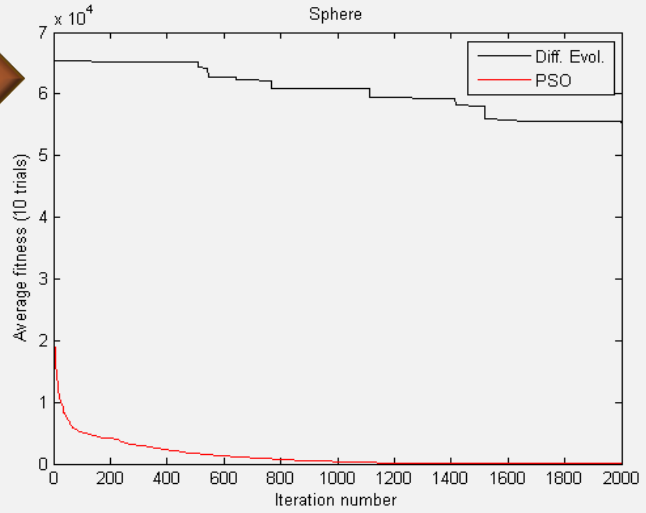
10 dimensions

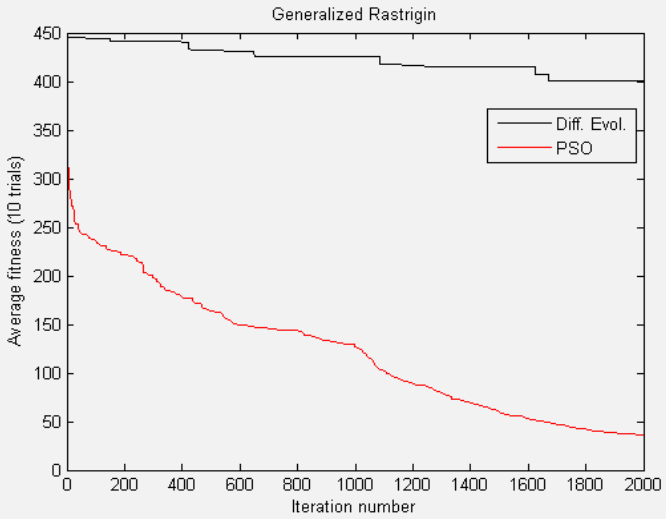


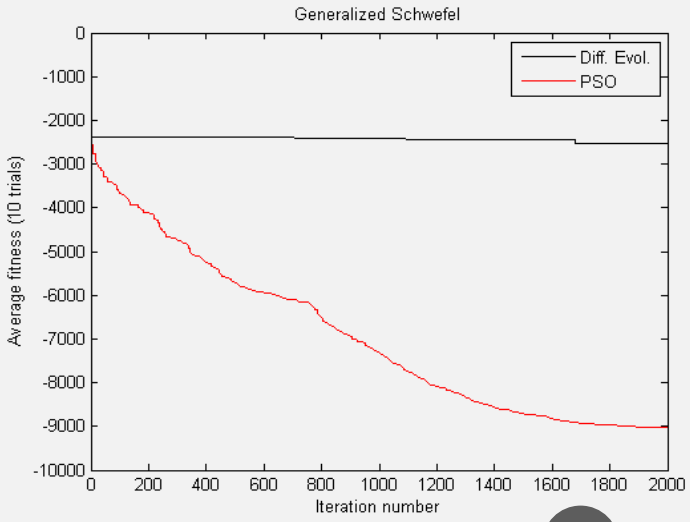




30 dimensions



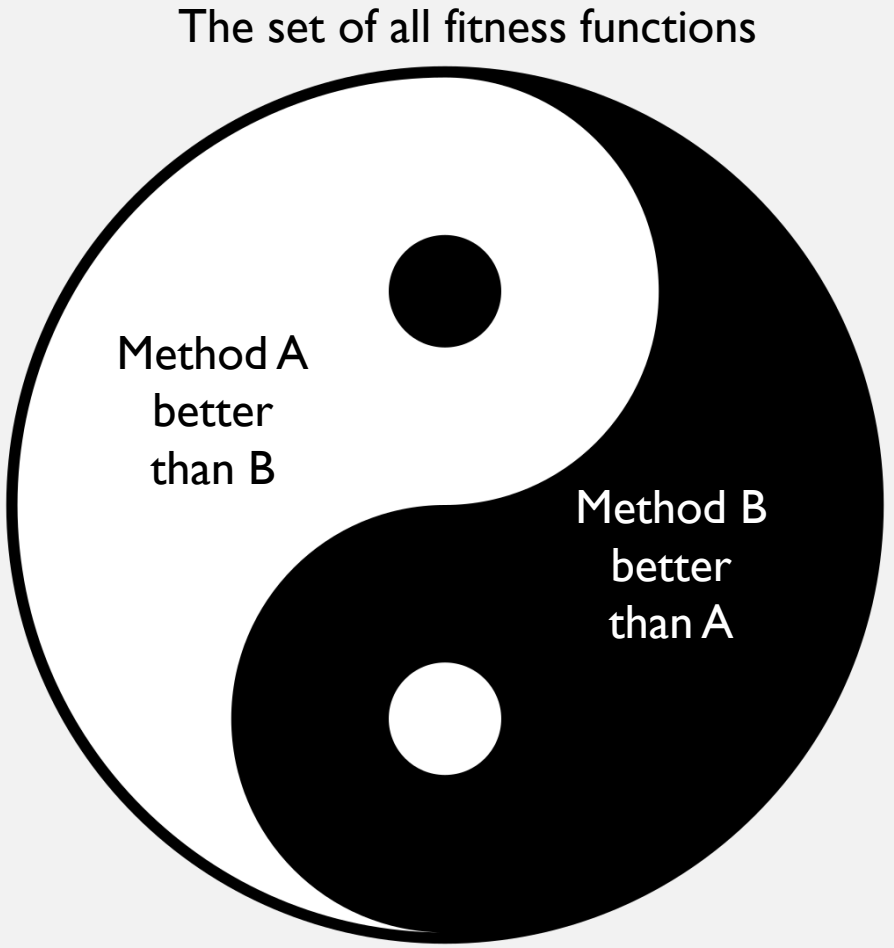




NO FREE LUNCH AND BENCHMARKING

NO FREE LUNCH

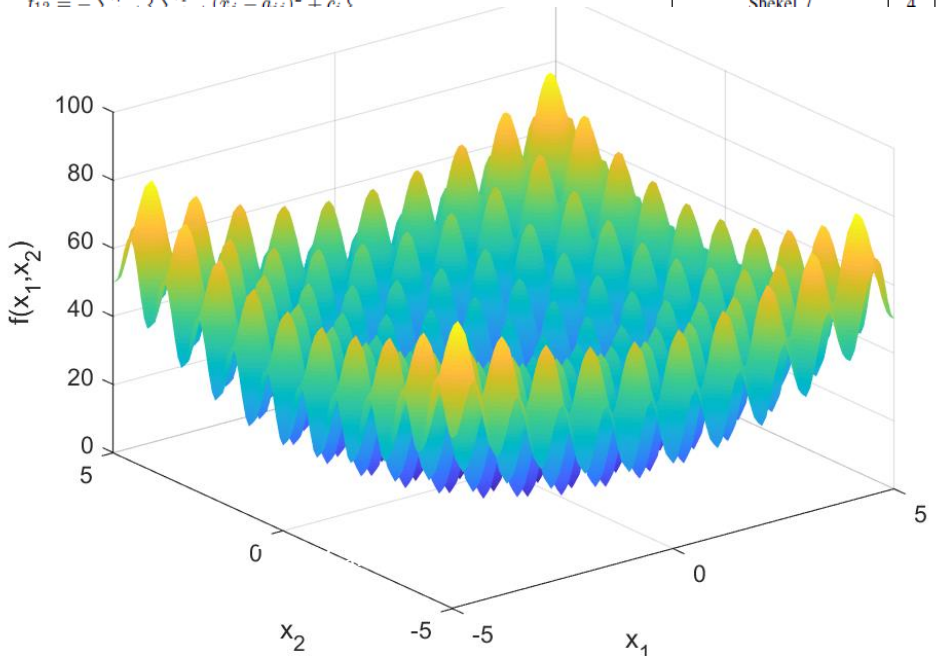
- Wolpert and McReady (1997): No free lunch theorems in optimization
- No practical stochastic optimization algorithm is better than any other when performance is averaged over all fitness functions



Bratton, Kennedy, 2007

TABLE I
BENCHMARK FUNCTIONS

Equation	Name	D	Feasible Bounds
$f_1 = \sum_{i=1}^{D} x_i^2$	Sphere/Parabola	30	$(-100, 100)^D$
$f_2 = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	Schwefel 1.2	30	$(-100, 100)^D$
$f_3 = \sum_{i=1}^{D-1} \left\{ 100 \left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2 \right\}$	Generalized Rosenbrock	30	$(-30, 30)^D$
$f_4 = -\sum_{i=1}^{D} x_i \sin\left(\sqrt{x_i}\right)$	Generalized Schwefel 2.6	30	$(-500, 500)^D$
$f_5 = \sum_{i=1}^{D} \left\{ x_i^2 - 10\cos(2\pi x_i) + 10 \right\}$	Generalized Rastrigin	30	$(-5.12, 5.12)^D$
$f_6 = -20\exp\left\{ -0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2} \right\} - \exp\left\{ \frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i) \right\} + 20 + e$	Ackley	30	$(-32, 32)^D$
$f_7 = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	Generalized Griewank	30	$(-600, 600)^D$
$f_8 = \frac{\pi}{D}\left\{ 10\sin^2(\pi y_i) + \sum_{i=1}^{D-1} (y_i - 1)^2 \left\{ 1 + 10\sin^2(\pi y_{i+1}) \right\} + (y_D - 1)^2 \right\}$ $+ \sum_{i=1}^{D} \mu(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $\mu(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	Penalized Function P8	30	$(-50, 50)^D$
$f_9 = 0.1\left\{ \sin^2(3\pi x_i) + \sum_{i=1}^{D-1} (x_i - 1)^2 \left\{ 1 + \sin^2(3\pi x_{i+1}) \right\} + (x_D - 1)^2 \times \left\{ 1 + \sin^2(2\pi x_D) \right\} \right\} + \sum_{i=1}^{D} \mu(x_i, 5, 100, 4)$	Penalized Function P16	30	$(-50, 50)^D$
$f_{10} = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	Six-hump Camel-back	2	$(-5, 5)^D$
$f_{11} = \left\{ 1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right) \right\} \times \left\{ 30 + (2x_1 - 3x_2)^2 \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right) \right\}$	Goldstein-Price	2	$(-2, 2)^D$
$f_{12} = -\sum_{i=1}^{5} \left\{ \sum_{j=1}^{4} (x_j - a_{ij})^2 + c_i \right\}^{-1}$	Shekel 5	4	$(0, 10)^D$
$f_{13} = -\sum^{7} \left\{ \sum^{4} (x_j - a_{ij})^2 + c_i \right\}^{-1}$	Shekel 7	4	$(0, 10)^D$



BENCHMARKING

NFL ⇒ It is important to test performance:

1. On a variety of benchmark fitness functions
2. For high dimensional search spaces

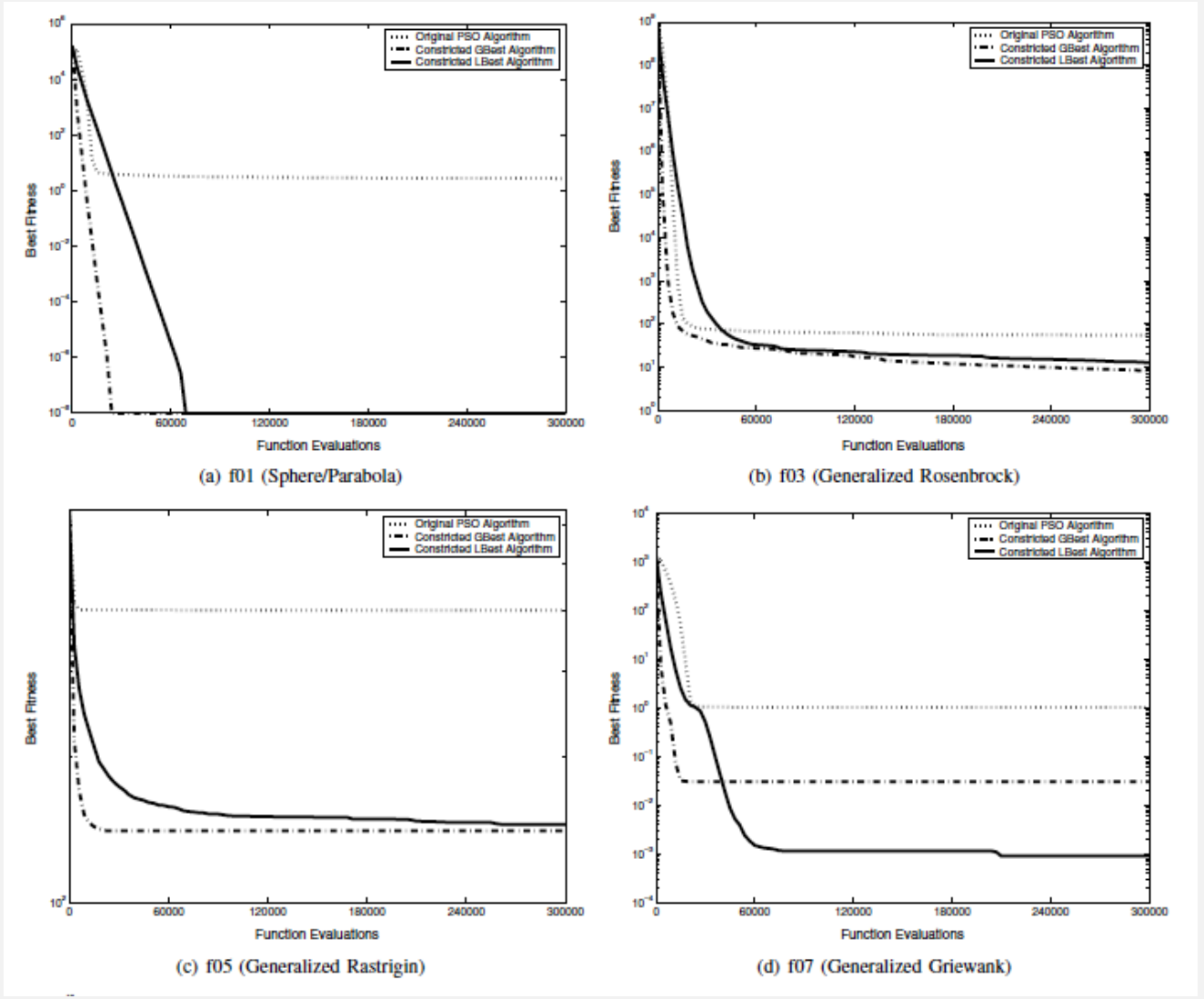
Comparison procedures should use rigorous statistical methodology

- See Bratton, Kennedy, 2007

BENCHMARKING METRICS

- Average the best fitness over multiple runs with independent initializations
- Study exploration vs exploitation behavior
- Statistical analysis: Two-sample tests of final best fitness values
  - Examples: Student's t-test, Kolmogorov-Smirnov test

Bratton, Kennedy, 2007



(a) f01 (Sphere/Parabola)

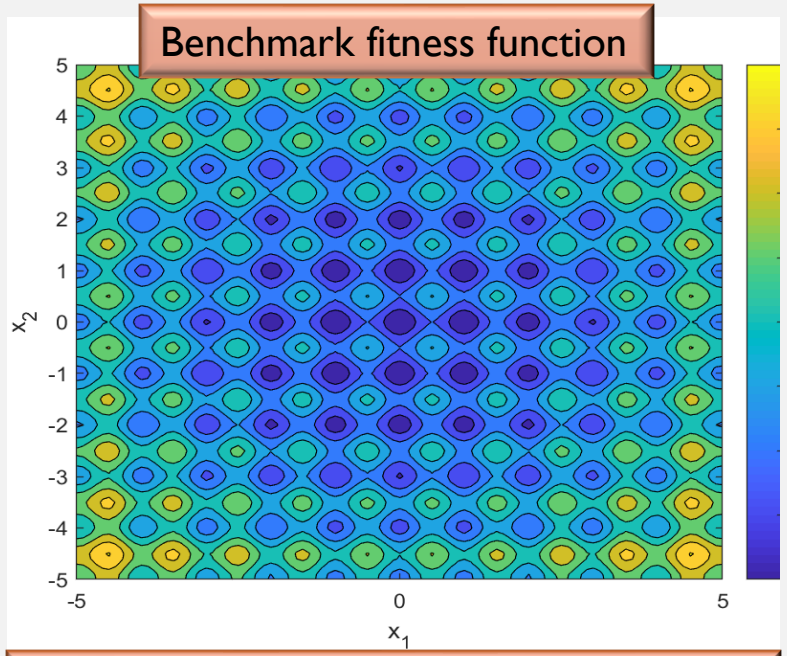
(b) f03 (Generalized Rosenbrock)

(c) f05 (Generalized Rastrigin)

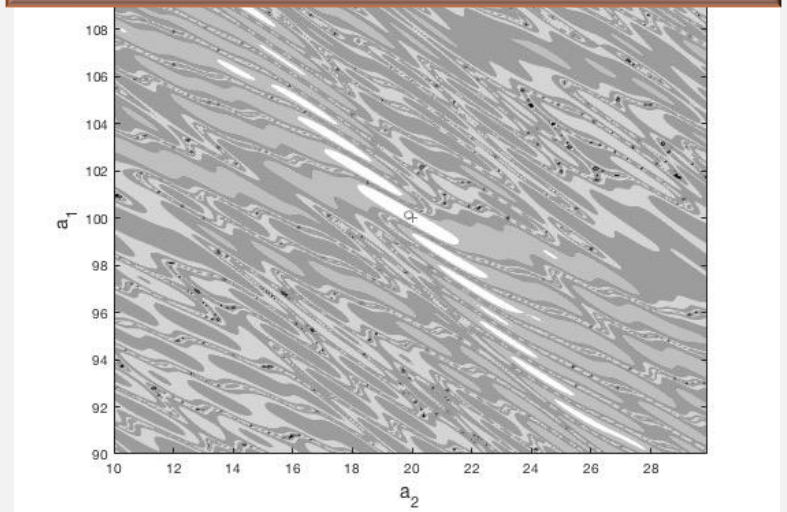
(d) f07 (Generalized Griewank)

TUNING

Benchmark fitness function



Fitness function for problem of interest

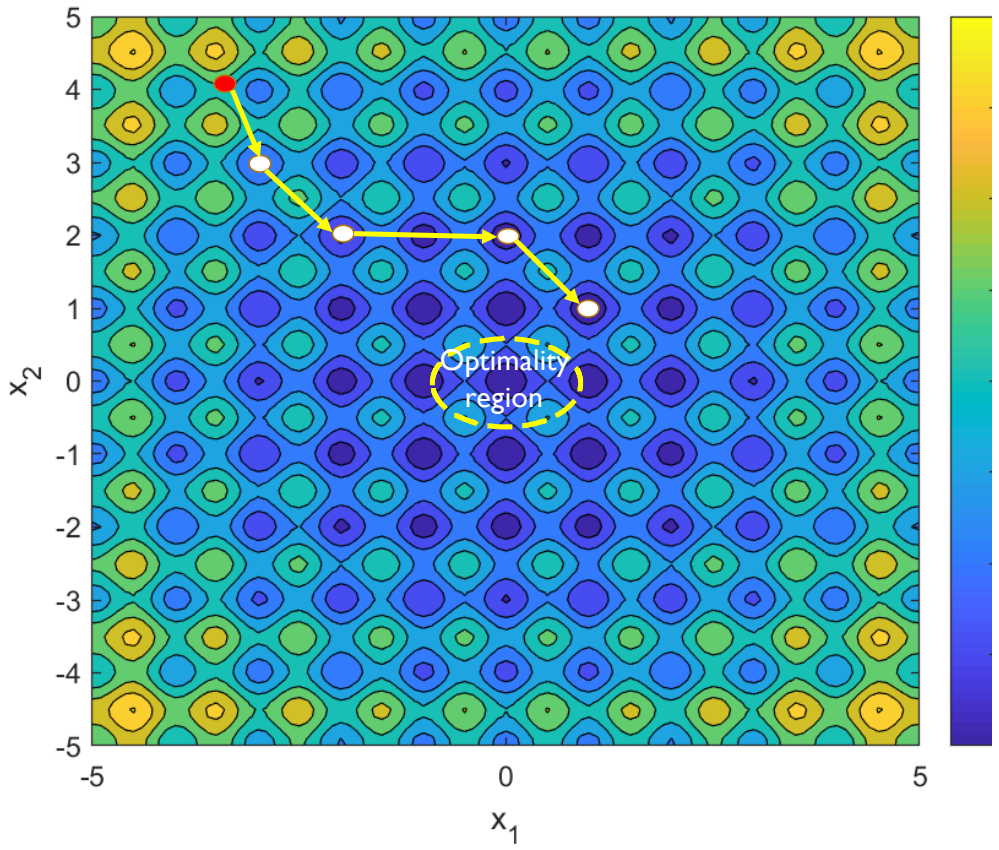


TUNING

- NFL ⇒ Good performance on benchmark fitness functions does not mean good performance on problem of interest
- The qualitative nature of benchmark functions maybe quite different
- Statistical regression: Noise induces many minima with comparable fitness values
- **Tuning**: Obtaining satisfactory performance on the class of problems of interest to the user

TUNING GOAL

- Stochastic optimization: No guaranteed convergence
- "Success": High probability of termination in the optimality region
- The goal of tuning is to increase the probability of success for a given class of optimization problems



TUNING AND BENCHMARKING

Benchmarking

- Requires fitness functions with low evaluation costs
- Goal: apples-to-apples comparison of optimization methods
- Benchmark functions represent different "classes" of fitness functions

Tuning

- Real world fitness evaluation: computational costs can be high
- Goal: Improving the performance of one optimization method
- Tuning should also use a variety of fitness functions but from the same class (*Lecture 3)

TUNING FOR REGRESSION PROBLEMS

- Different data realizations → different fitness functions but same "class"
- NFL ⇒ Over-tuning on one data realization ⇒ Worse performance on other realizations

"BEST OF M RUNS" STRATEGY

probability of "success" in one run:

p

Probability of failure over M runs:

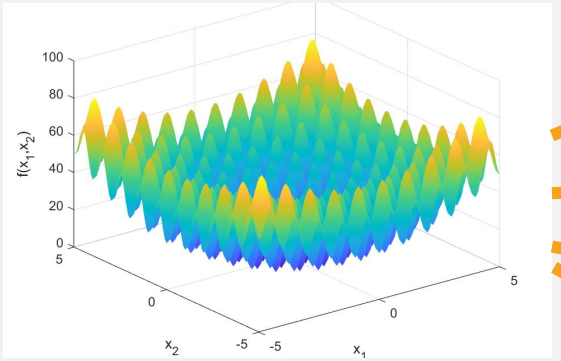
$(1-p)^M$

- Example: If $p = 0.5$, failure probability over $M = 10$ runs is $\simeq 0.001$

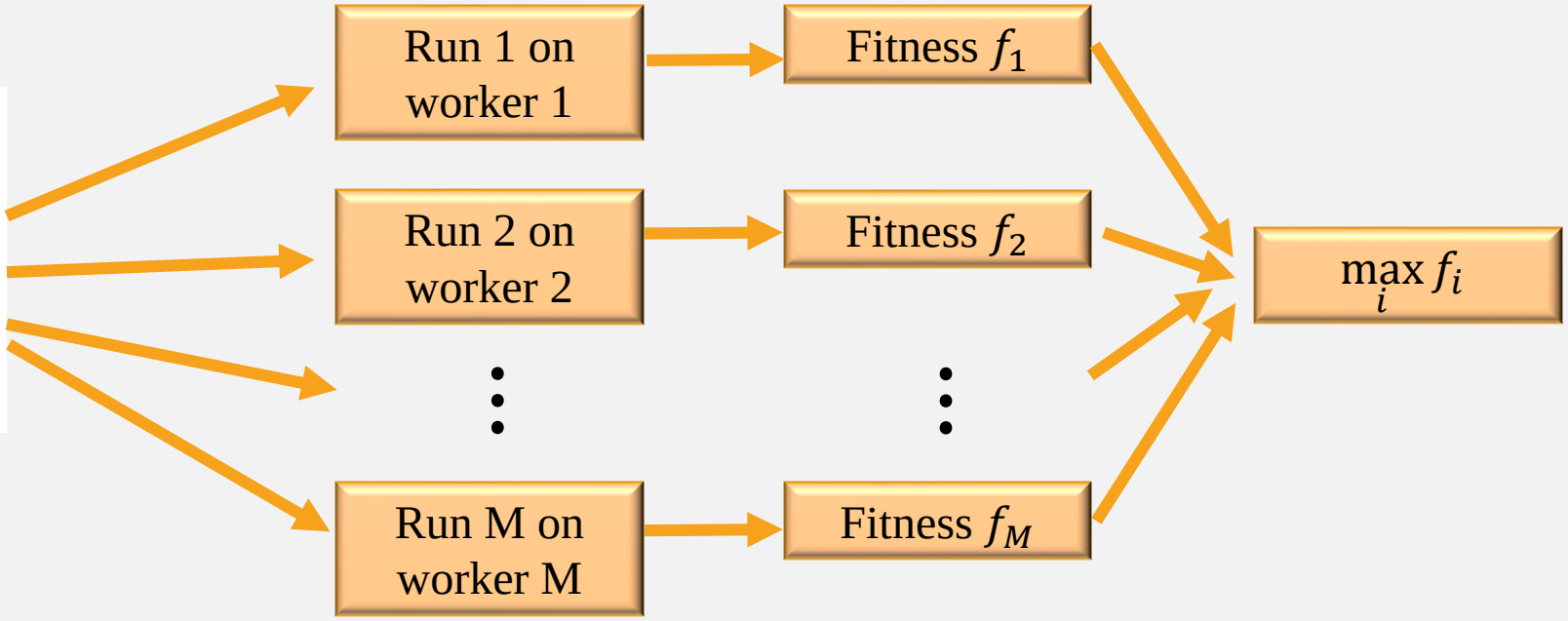
Tuning strategy: Target a moderately high p and pick best fitness from M runs

- Moderate tuning reduces the danger of over-tuning
- Reduces the effort needed to achieve good tuning

PRALLELIZATION IN BMR STRATEGY



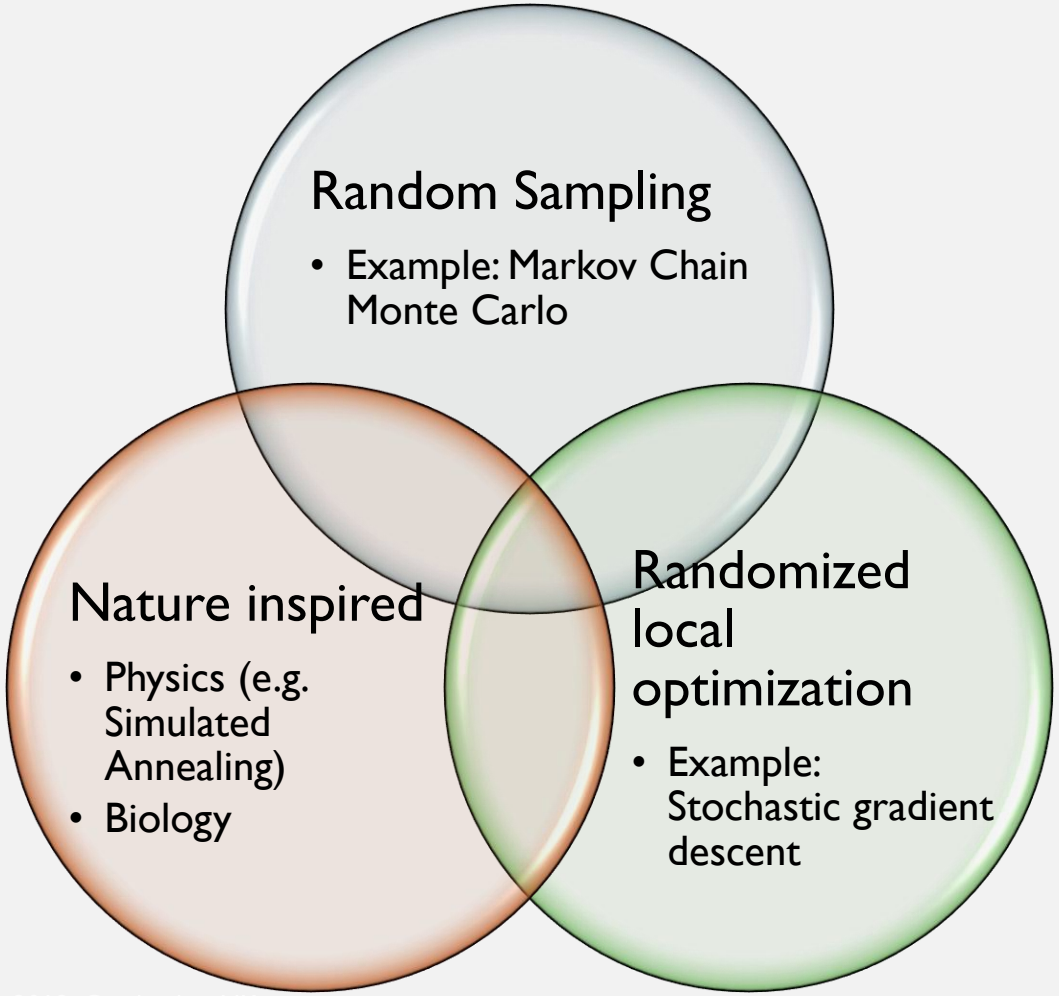
Fitness function



STOCHASTIC OPTIMIZATION METHODS

Brief overview

METAHEURISTICS

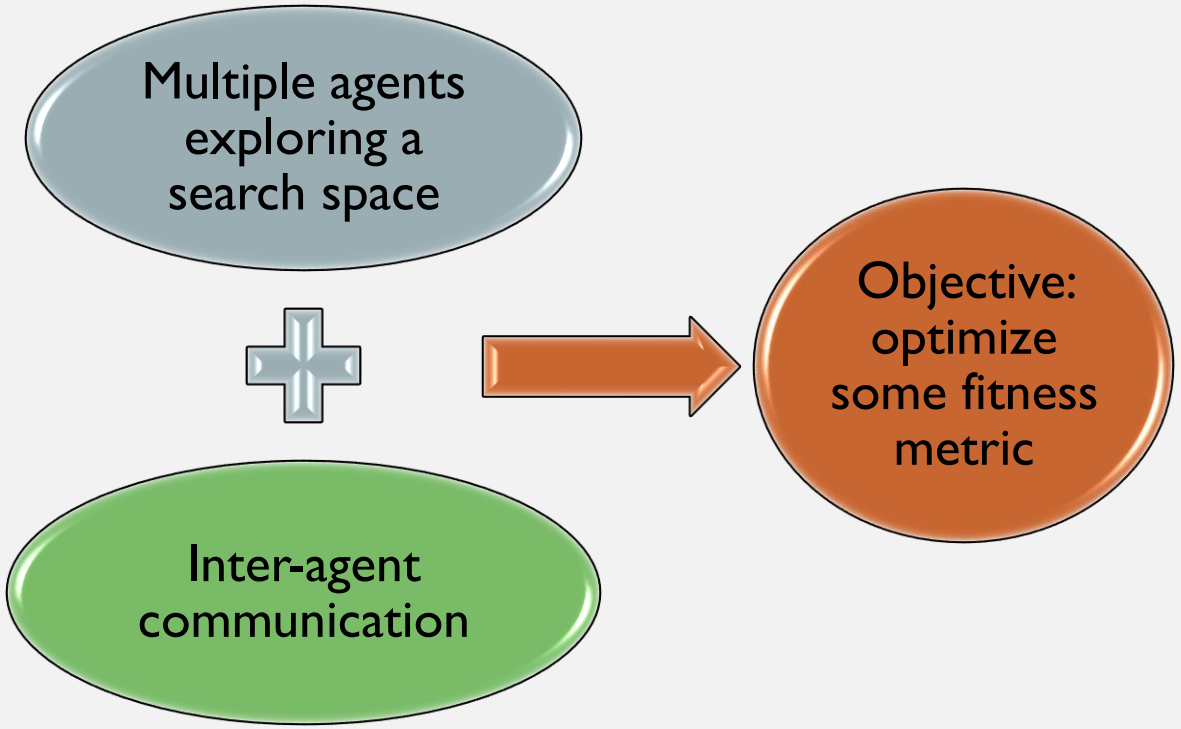


Well-established approaches in stochastic optimization literature

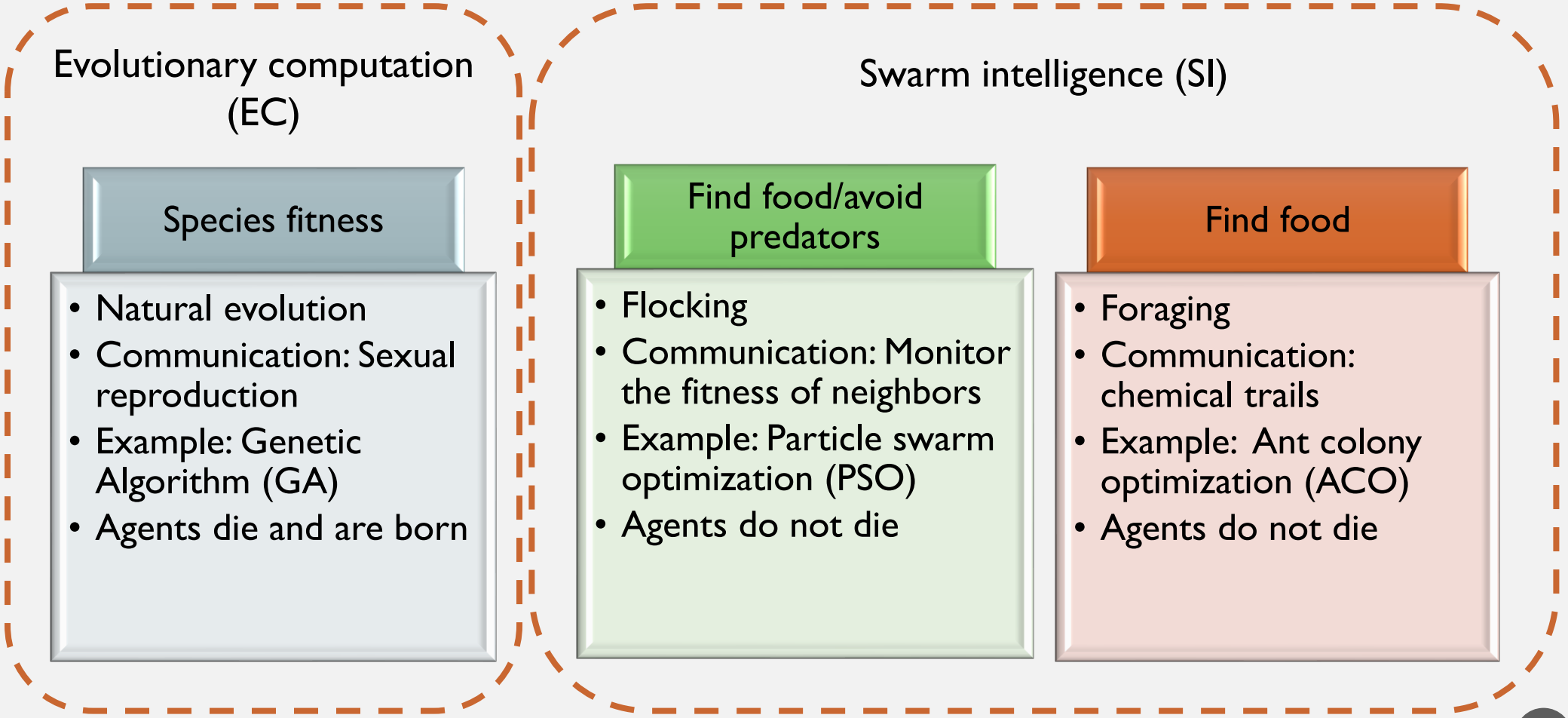
The boundaries of the metaheuristics are permeable

New metaheuristics arise quite often

OPTIMIZATION IN BIOLOGICAL PHENOMENA



BIOLOGY INSPIRED METHODS



GENETIC ALGORITHM

Initialization

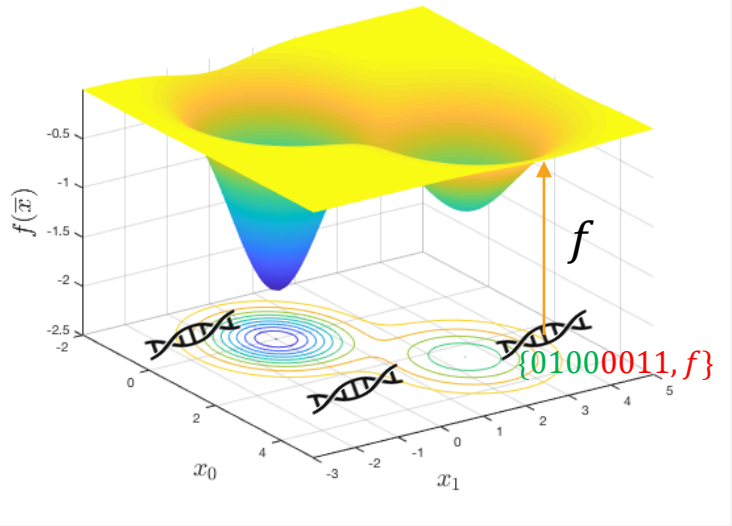
- Genome: Representation of agent location
- Genome fitness: Fitness function value at agent location

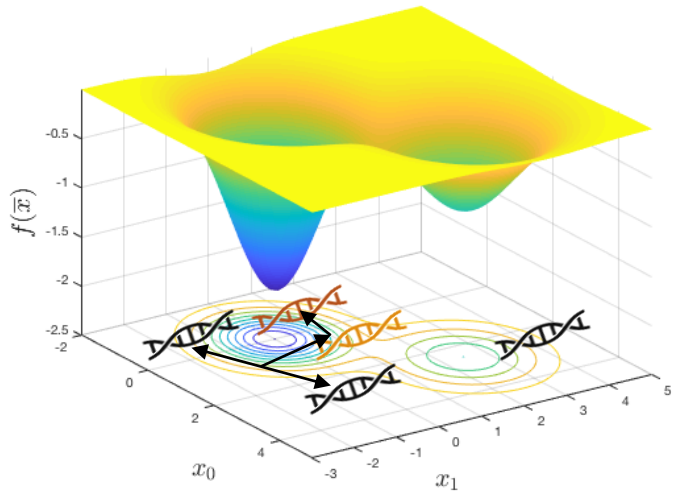
Crossover & Mutation

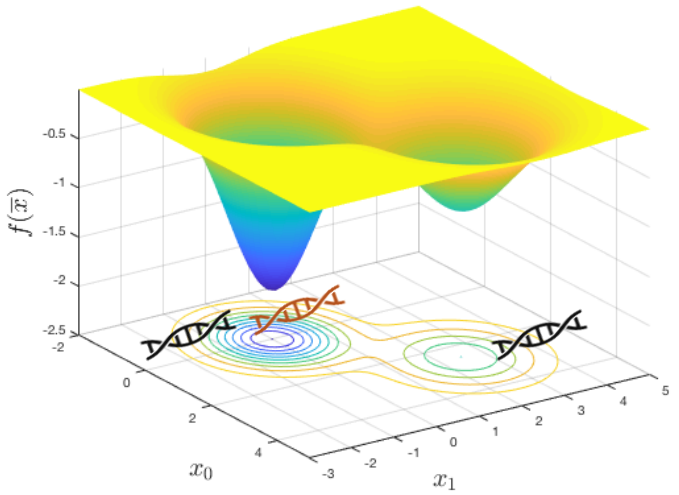
- Mix parent genomes to produce a child genome
- Random changes in child genome

Selection:

- Genomes with better fitness have higher chances of surviving



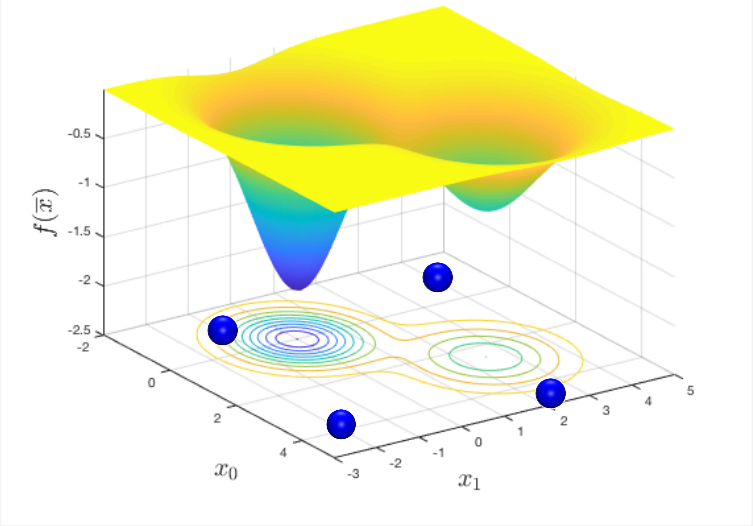




DIFFERENTIAL EVOLUTION

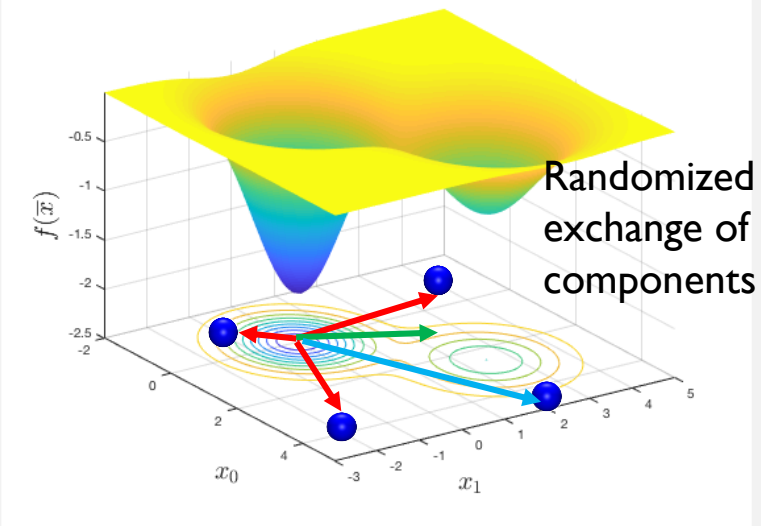
Initialization

- Genome: position vector
- Genome fitness: Fitness function value at agent location



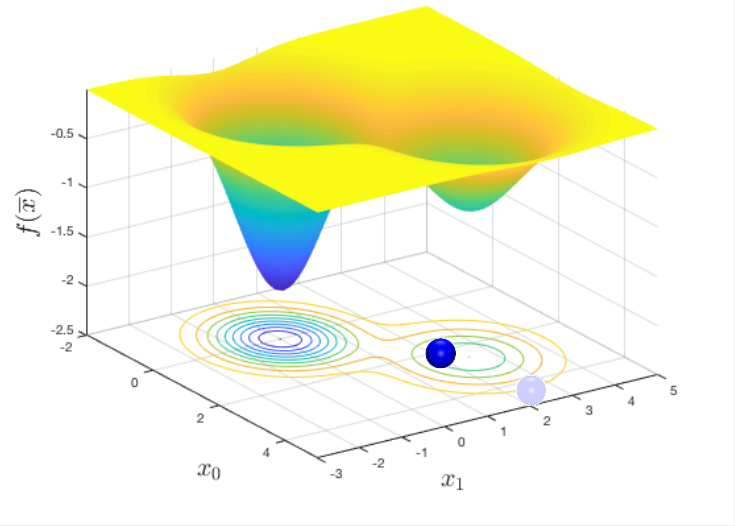
Crossover & Mutation

- Linear combination of three random genomes
- Crossover with current particle genome



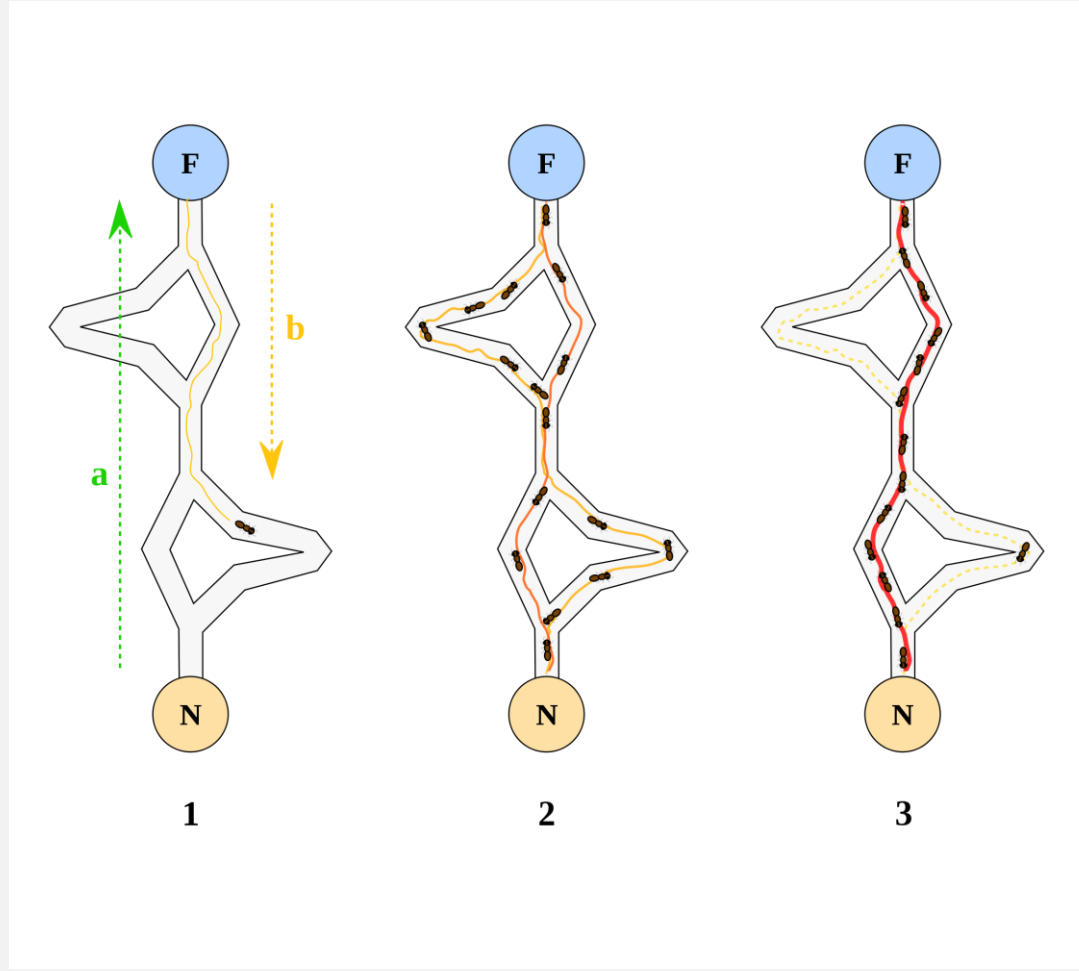
Selection:

- Select genomes with better fitness



ANT COLONY OPTIMIZATION

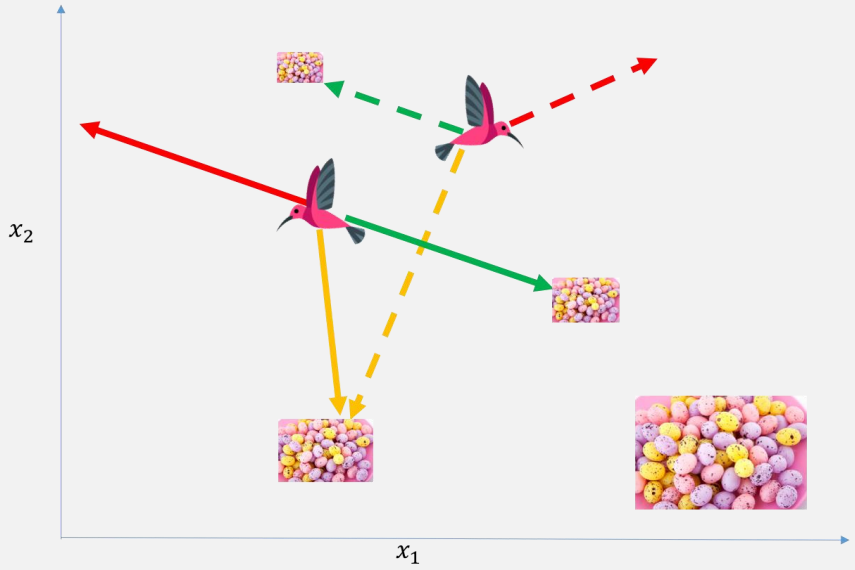
- Based on the foraging behavior of ants
- Each ant leaves a "pheromone" trail
- More pheromone attracts more ants
- Better suited to discrete optimization (like GA)
  - Shortest path problems



http://www.sciencedirect.com/science/article/pii/S0142061515005840

PARTICLE SWARM OPTIMIZATION

Kinematics and Dynamics – Basic PSO



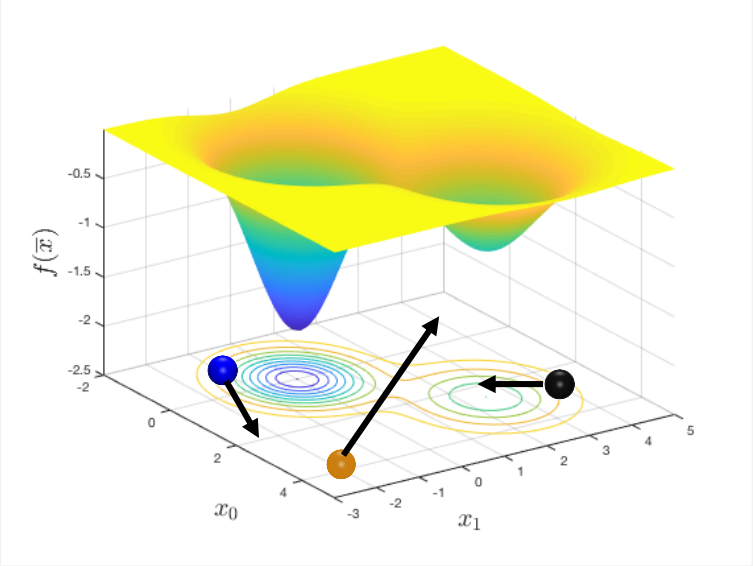
PARTICLE SWARM OPTIMIZATION

- A swarm intelligence method inspired by the flocking behavior of birds
  - Flocking: agents mimic the velocity of nearby agents
  - Swarming: velocity mimicry removed
- Each agent moves under random attraction towards the best food sources that it and the swarm have found

PARTICLE SWARM OPTIMIZATION

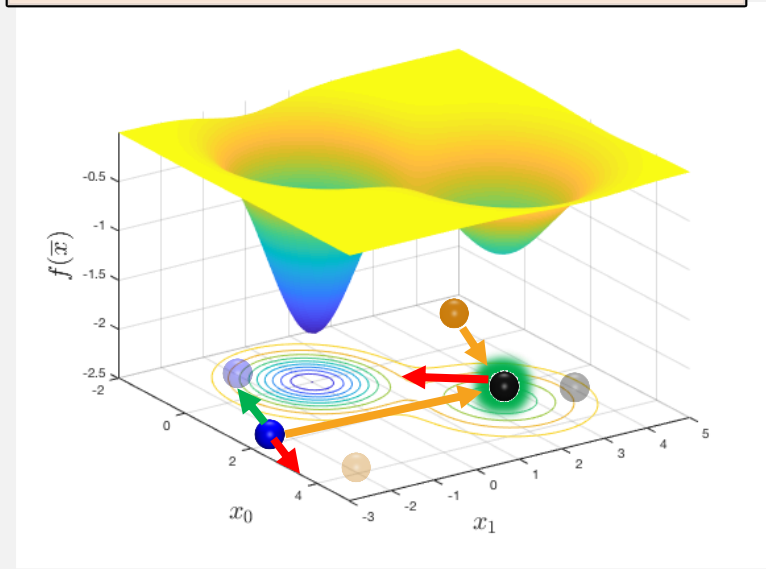
Initialization

- Particle: agent location
- Particle fitness: Fitness value at location
- Particle "velocity": Displacement vector to new position



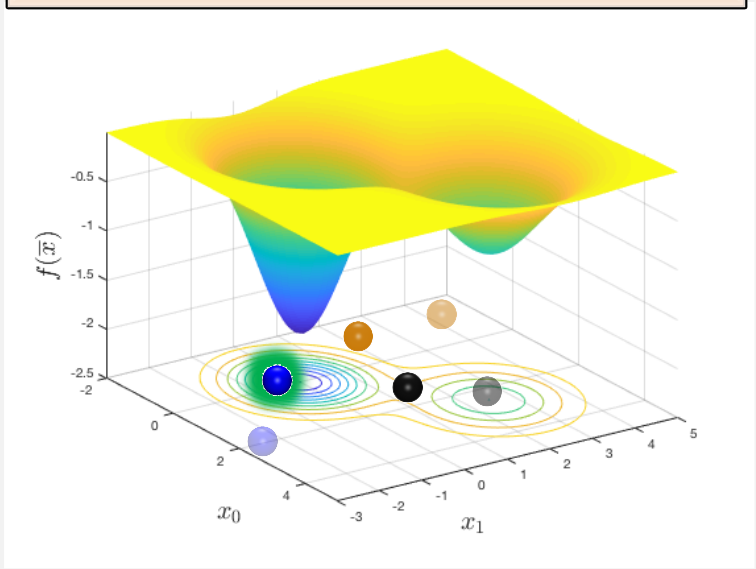
Velocity update

- New velocity: sum of old velocity + acceleration terms
- Acceleration strengths are random



Position update

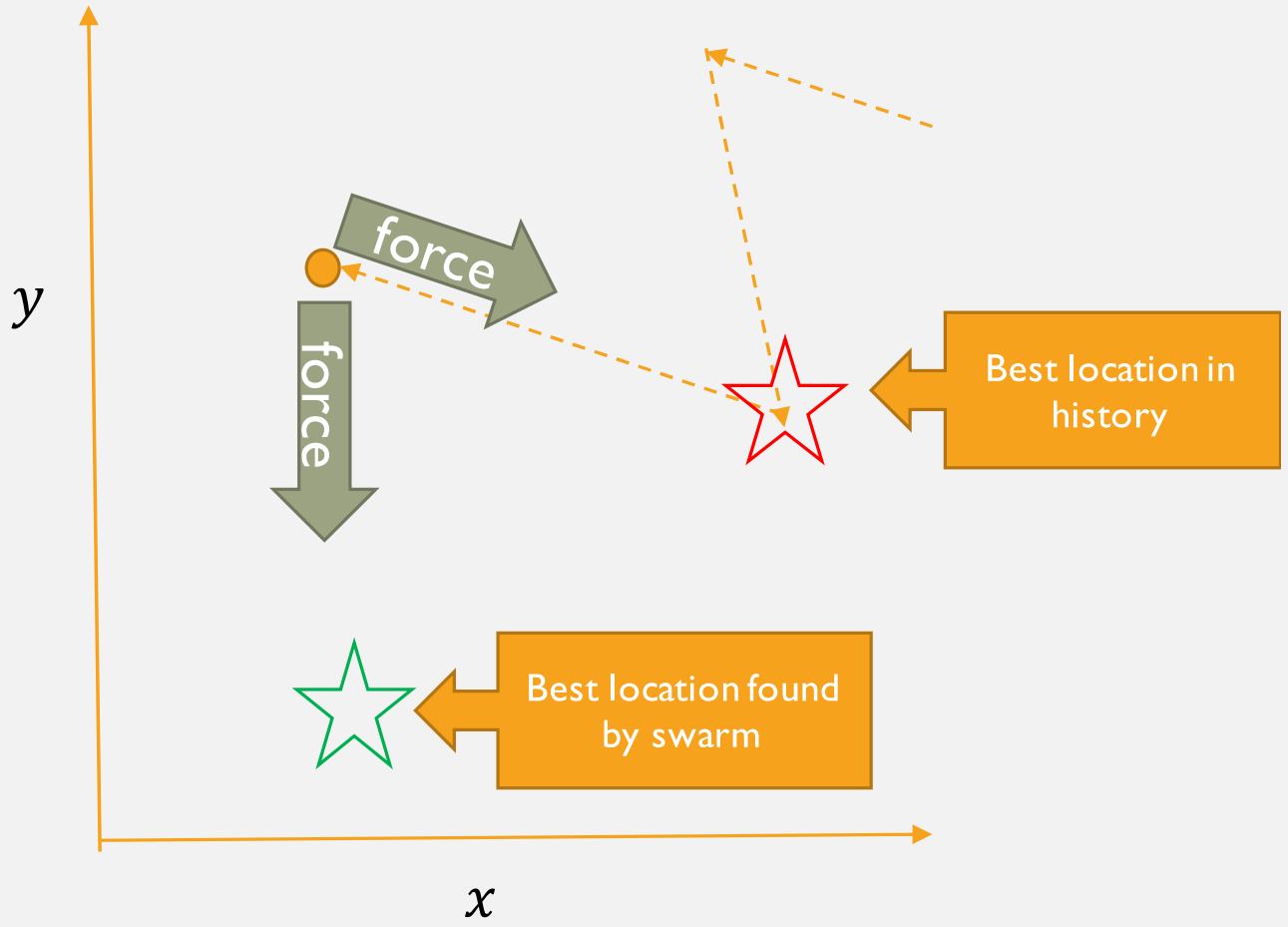
- Particles move to new positions



VELOCITY UPDATE

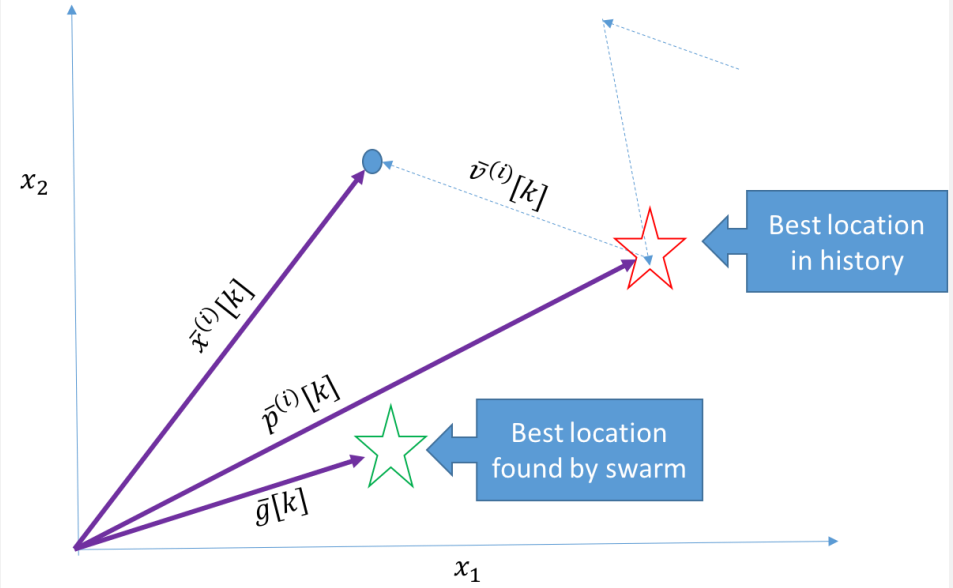
A particle explores the search space randomly but constantly feels an attractive force towards:

1. Personal best: best location it has found so far and …
2. Global best: the best location found by the swarm so far



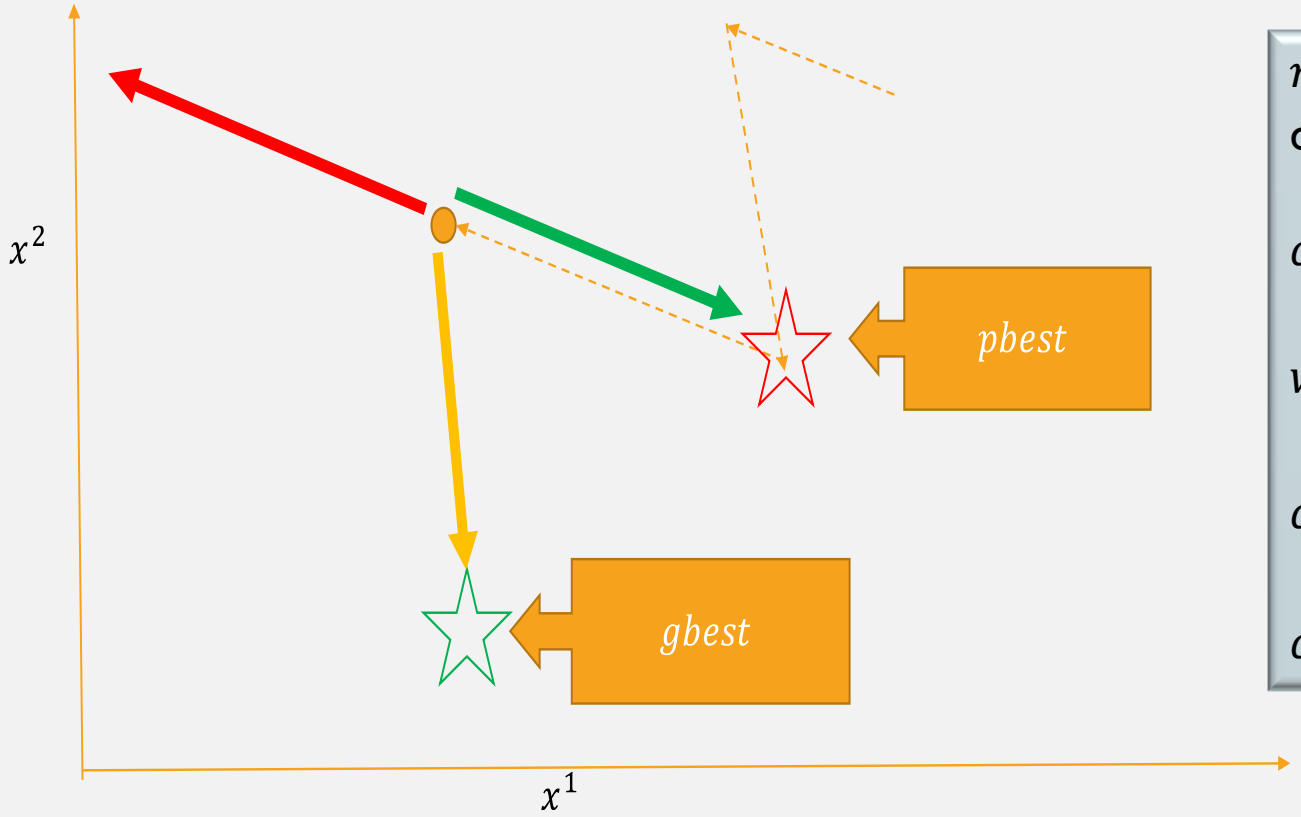
PSO TERMINOLOGY

Term	Definition
Particles	Locations in search space
$\bar{x}^{(i)}[k]$	• Position of i^{th} particle in k^{th} iteration • $\bar{x}^{(i)}[k] = (x_0^{(i)}[k], x_1^{(i)}[k], \dots, x_D^{(i)}[k])$
$\bar{v}^{(i)}[k]$	• Velocity of i^{th} particle in k^{th} iteration • $\bar{v}^{(i)}[k] = (v_0^{(i)}[k], v_1^{(i)}[k], \dots, v_D^{(i)}[k])$
$pbest$ $(\bar{p}^{(i)}[k])$	Best location found by the i^{th} particle over iterations 1 through k
$gbest$ $(\bar{g}[k])$	Best location found by any particle over iterations 1 through k
v_{max}	Maximum velocity "Velocity Clamping": $v_j^{(i)}[k] \in [-v_{max}, v_{max}]$



VELOCITY UPDATE

$$v_j^{(i)}[k+1] = w\, v_j^{(i)}[k] + c_1 r_{1,j}(p_j^{(i)}[k] - x_j^{(i)}[k]) + c_2 r_{2,j}(g_j\,[k] - x_j^{(i)}[k])$$



$r_{m,j}$: random variable with uniform distribution in $[0,1]$

c_1, c_2 :"**acceleration constants**"

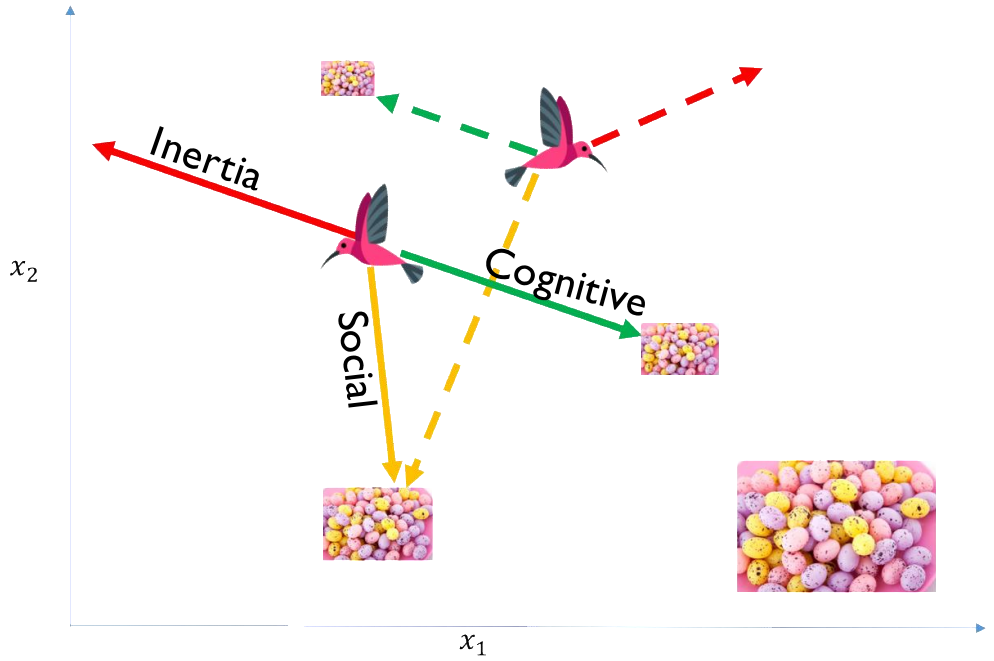
w : "inertia" $\rightarrow w\, v_j^{(i)}[k]$:"**Inertia Term**"

$c_1 r_{1,j}(p_i^j[k] - x_i^j[k])$:"**Cognitive term**"

$c_2 r_{2,j}(g[k] - x_i^j[k])$:"**Social term**"

INTERPRETATION

- Inertia term: promotes exploration
  - $w < 1$ to avoid "particle explosion"
  - Common choice: Linear decay of w
- Social and cognitive terms: promote exploitation
  - Randomization in these terms promotes exploration



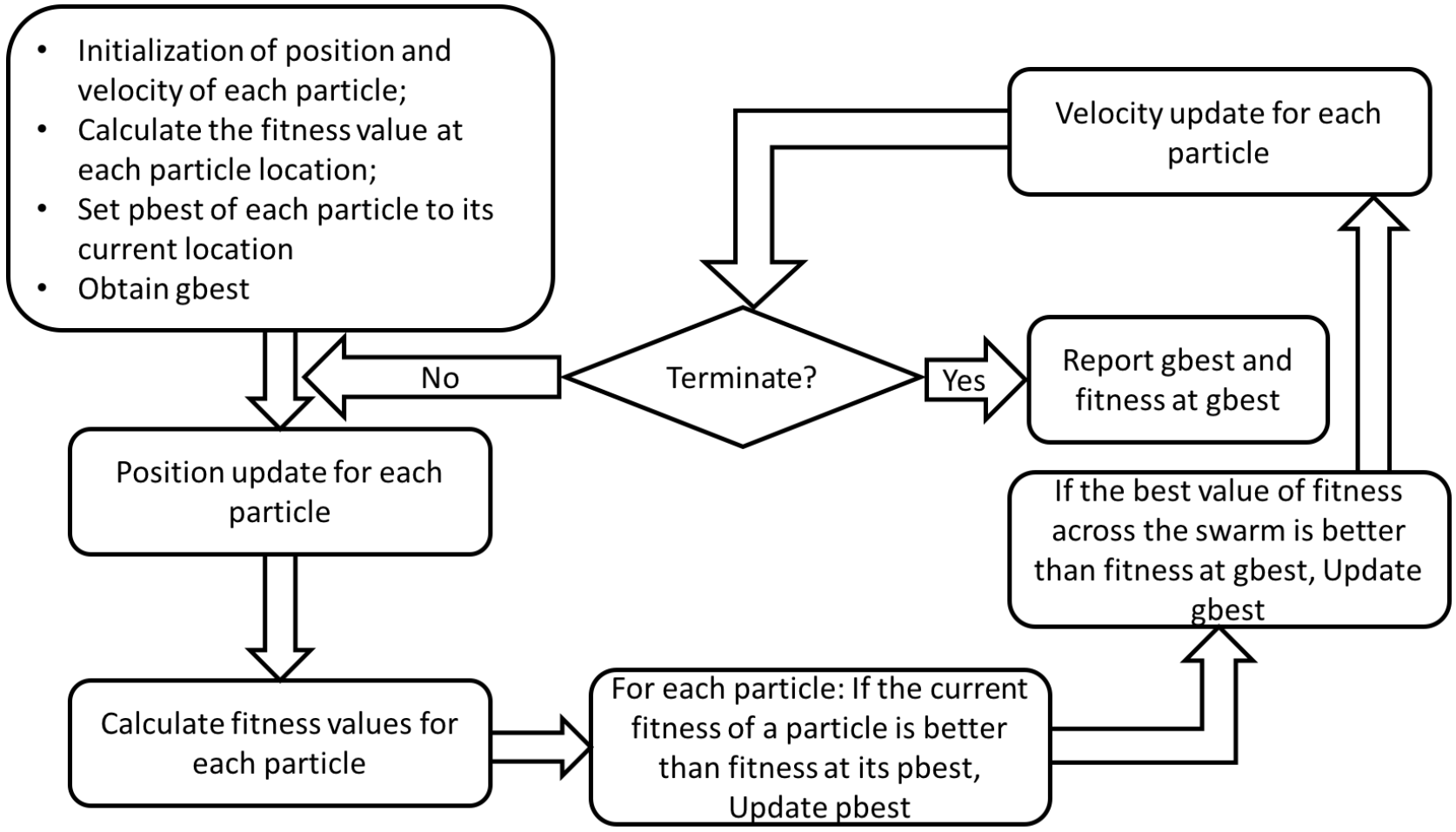
PSO DYNAMICAL EQUATIONS

Velocity update

$$v_j^{(i)}[k+1] = w\, v_j^{(i)}[k] + c_1 r_{1,j}(p_j^{(i)}[k] - x_j^{(i)}[k]) + c_2 r_{2,j}(g_j\,[k] - x_j^{(i)}[k])$$

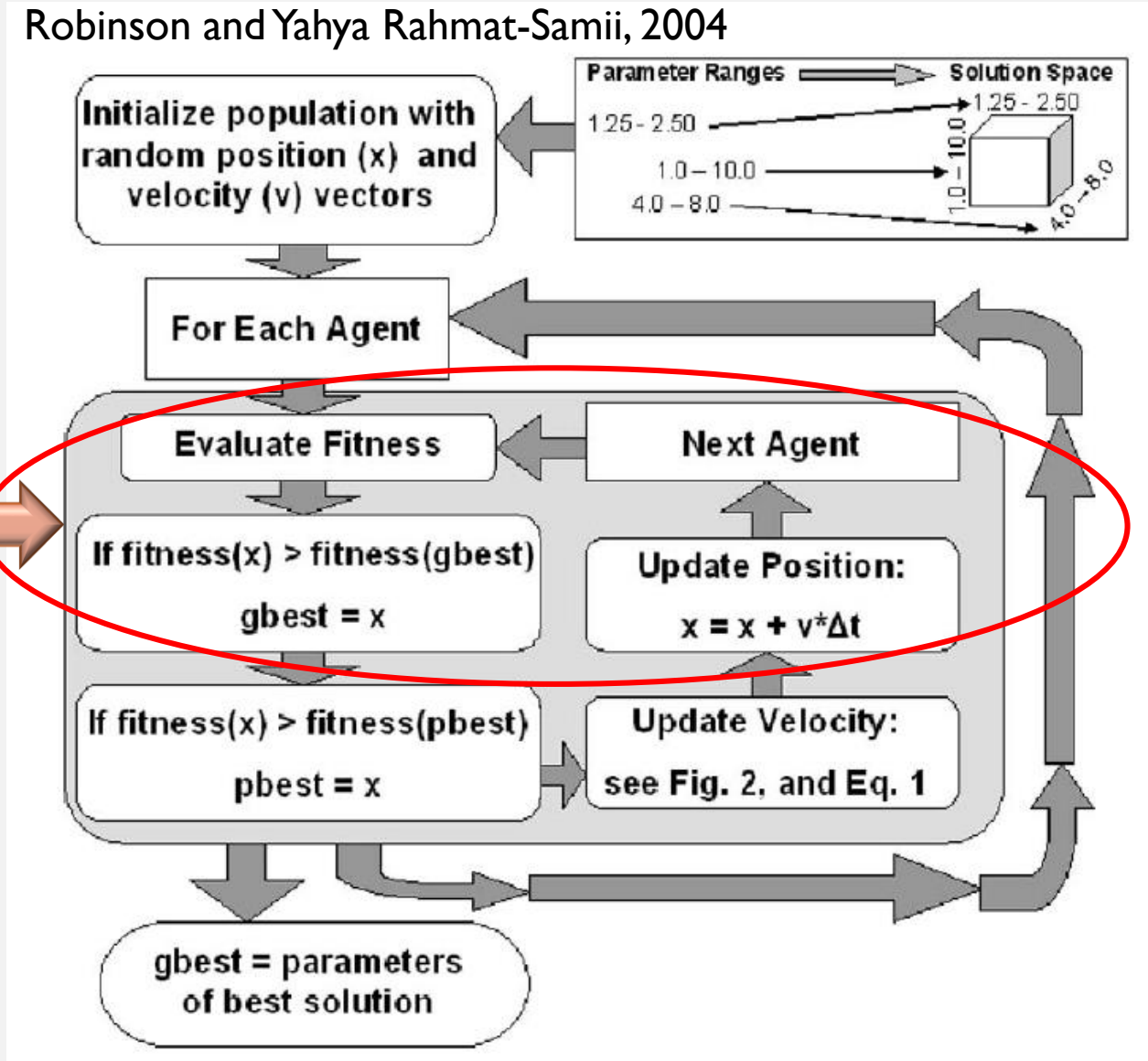
Position update

$$x_j^{(i)}[k+1] = x_j^{(i)}[k] + v_j^{(i)}[k+1]$$


Initialization of position and velocity of each particle;
Calculate the fitness value at each particle location;
Set pbest of each particle to its current location
Obtain gbest
Velocity update for each particle
Terminate?
No
Yes
Report gbest and fitness at gbest
Position update for each particle
If the best value of fitness across the swarm is better than fitness at gbest, Update gbest
Calculate fitness values for each particle
For each particle: If the current fitness of a particle is better than fitness at its pbest, Update pbest

Synchronous vs asynchronous update

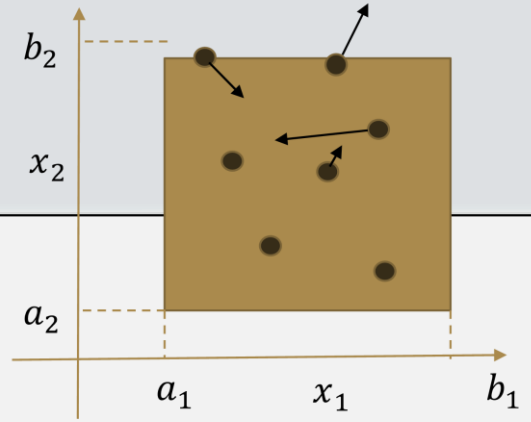
Robinson and Yahya Rahmat-Samii, 2004



INITIALIZATION AND TERMINATION

Initialization

- $x_j^{(i)}[0]$ is picked from a uniform distribution over $[a_j, b_j]$
- Search space assumed to be a hypercube



Initial velocity (variations)

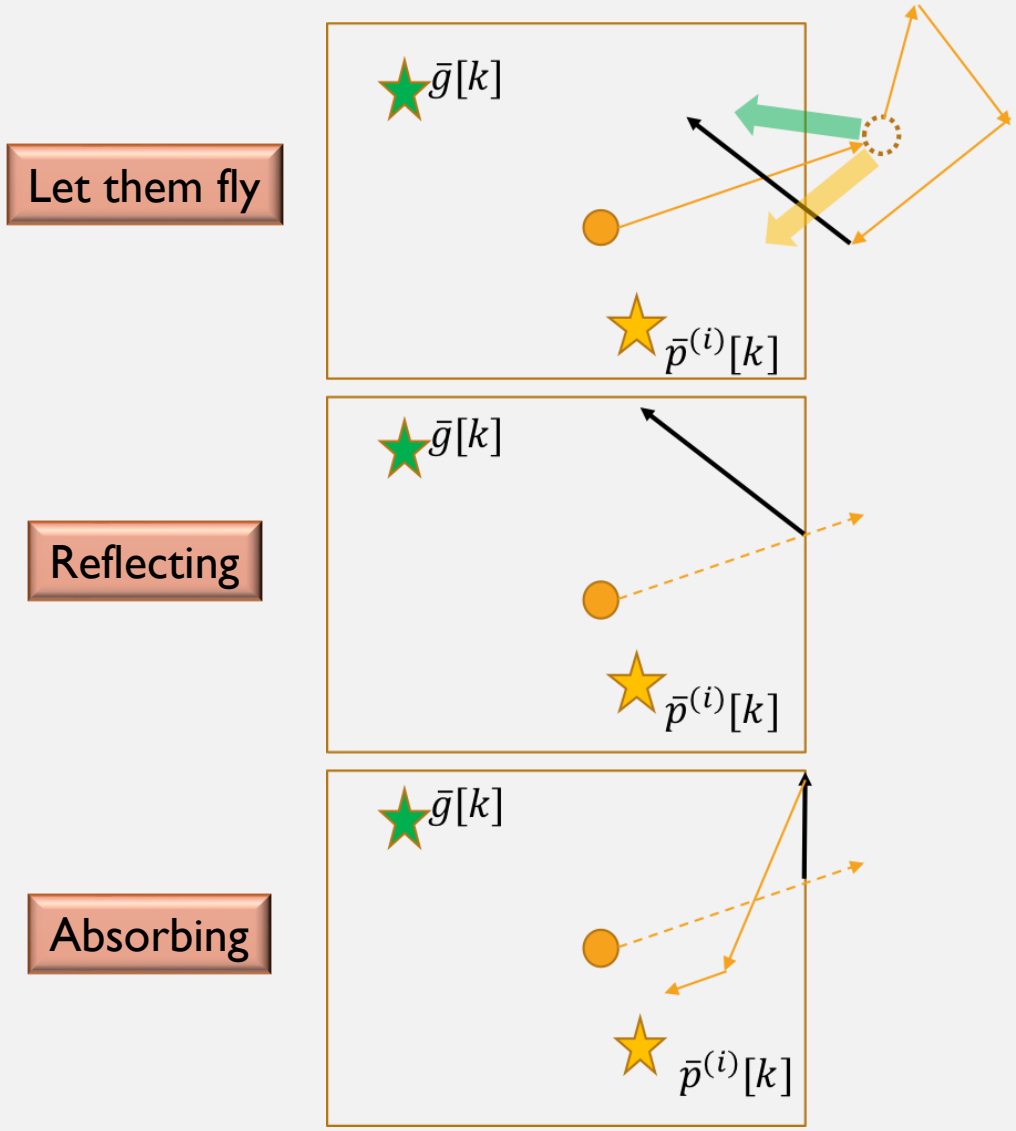
- Uniform distribution with velocity clamping
- Zero initial velocities
- Boundary constrained:
  - $v_j^{(i)}[0] \sim U(a_j - x_j^{(i)}[0], b_j - x_j^{(i)}[0])$ & velocity clamping

Termination condition (variations)

- Number of iterations (simplest)
- Density of swarm
- …

BOUNDARY CONDITIONS

- "Let them fly": set fitness to $+\infty$ outside the boundary and continue to iterate the dynamical equations
  - $pbest$ and $gbest$ eventually pull the particle back
- "Reflecting walls" : Change the sign of the velocity component perpendicular to the boundary surface
- "Absorbing Walls": zero the velocity component perpendicular to the boundary surface



SUMMARY

STOCHASTIC OPTIMIZATION

- Exploration-Exploitation trade-off is the key characteristic of practical methods
- NFL theorem: benchmarking is indispensable for characterizing performance
- Problem of interest: Tuning required
- BMR strategy recommended if parallel computing available
  - Reduces the danger of over-tuning
  - Reduces effort needed to achieve well-tuned performance

EC AND SI METHODS

- EC and SI: Biology inspired metaheuristics
- Different methods have different numbers of algorithm parameters
- Algorithm parameters influence exploration-exploitation trade-off
- Curse of high dimensionality:
  - Algorithm settings in low dimensional search space can lead to unforeseen effects in high dimensional ones
  - Exercise caution in making your own modifications: Benchmarking essential