Differential Evolution

Thea Behrens,
Jonas Görlich,
Inga Ibs,
Noa Kallioinen,
Aiswarya Rajendran Nair Minikutty

2017-12-07

Table of Contents

Introduction

Introduction

Differential Evolution

Continuous (real valued vectors)

Works for optimization problems that are non- differentiable

Parallelizability to cope with computation intensive cost functions.

Ease of use because of few control variables that are easy to choose and are robust to steer the minimization.

Comparatively good convergence properties.

Introduction

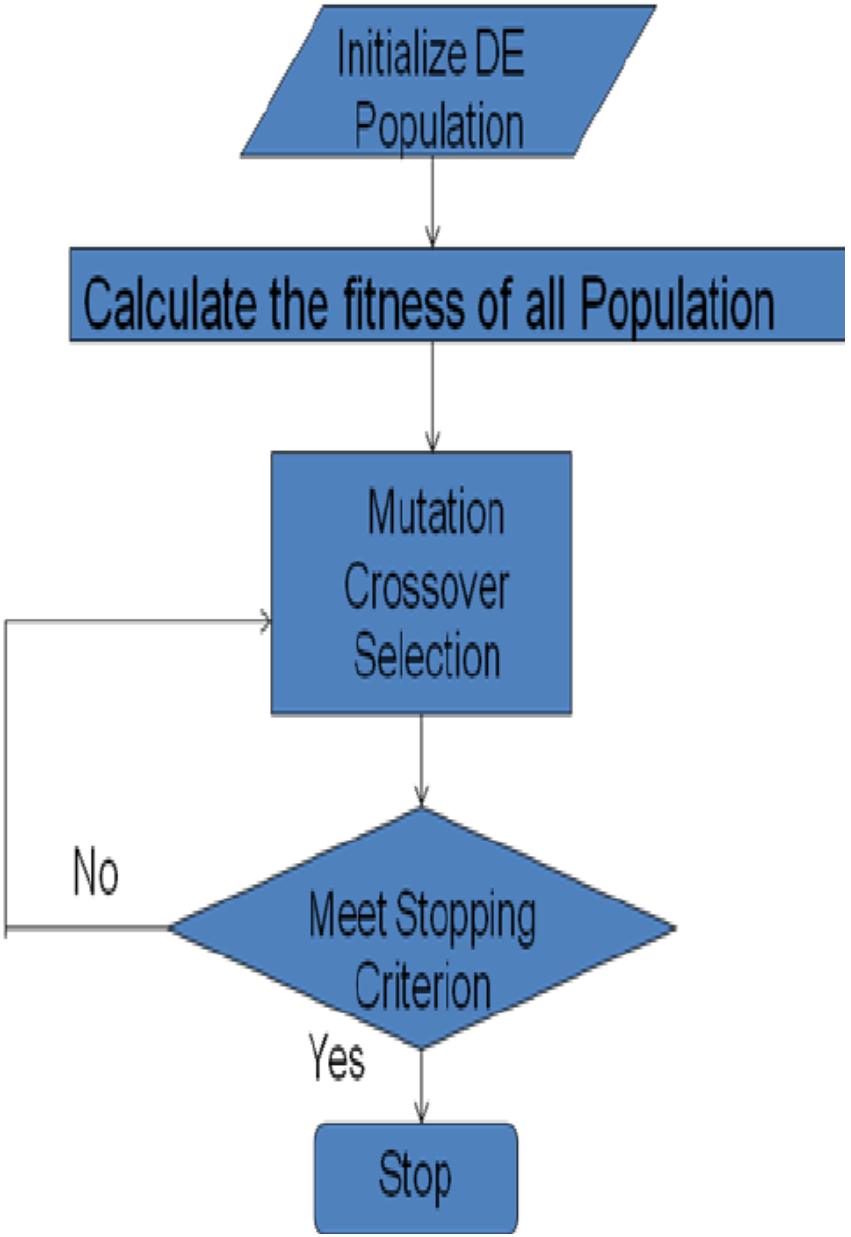
DE uses distance and direction information from the current population to guide the search process.

In DE the initial population, n consists of d-dimensional vectors.

Vector is an object that has both a magnitude and a direction.

Magnitude of a vector $= |\mathbf{v}| = \sqrt[1]{x^2 + y^2}$ where $\mathbf{v} = \langle x, y \rangle$

Overview


Initialize DE Population
Calculate the fitness of all Population
Mutation
Crossover
Selection
No
Meet Stopping Criterion
Yes
Stop

Advantages of using Vectors

The information about the fitness landscape, as represented by the current population, is used to direct the search.

If the population is sufficiently large then the population wont suffer genetic drift.

Differential Evolution

Key idea

Replace random mutation with *differential mutation*

Overview

Minimize objective function $f : \mathbb{R}^D \to \mathbb{R}$

Population	N D-dimensional parameter vectors
Control Parameters	Crossover rate $Cr \in [0, 1]$
	Scaling factor $F > 0$
	Population size $N > 4$

Overview

Initialize population
repeat
    Select *target* vectors from population
    for each target **in** targets
        Create a *donor* by *differential mutation*
        Crossover target and donor to form *trial*
    Select next population
until stopping criterion reached

Overview

Initialize population

repeat

~~Select *target* vectors from population~~

for each target **in** population

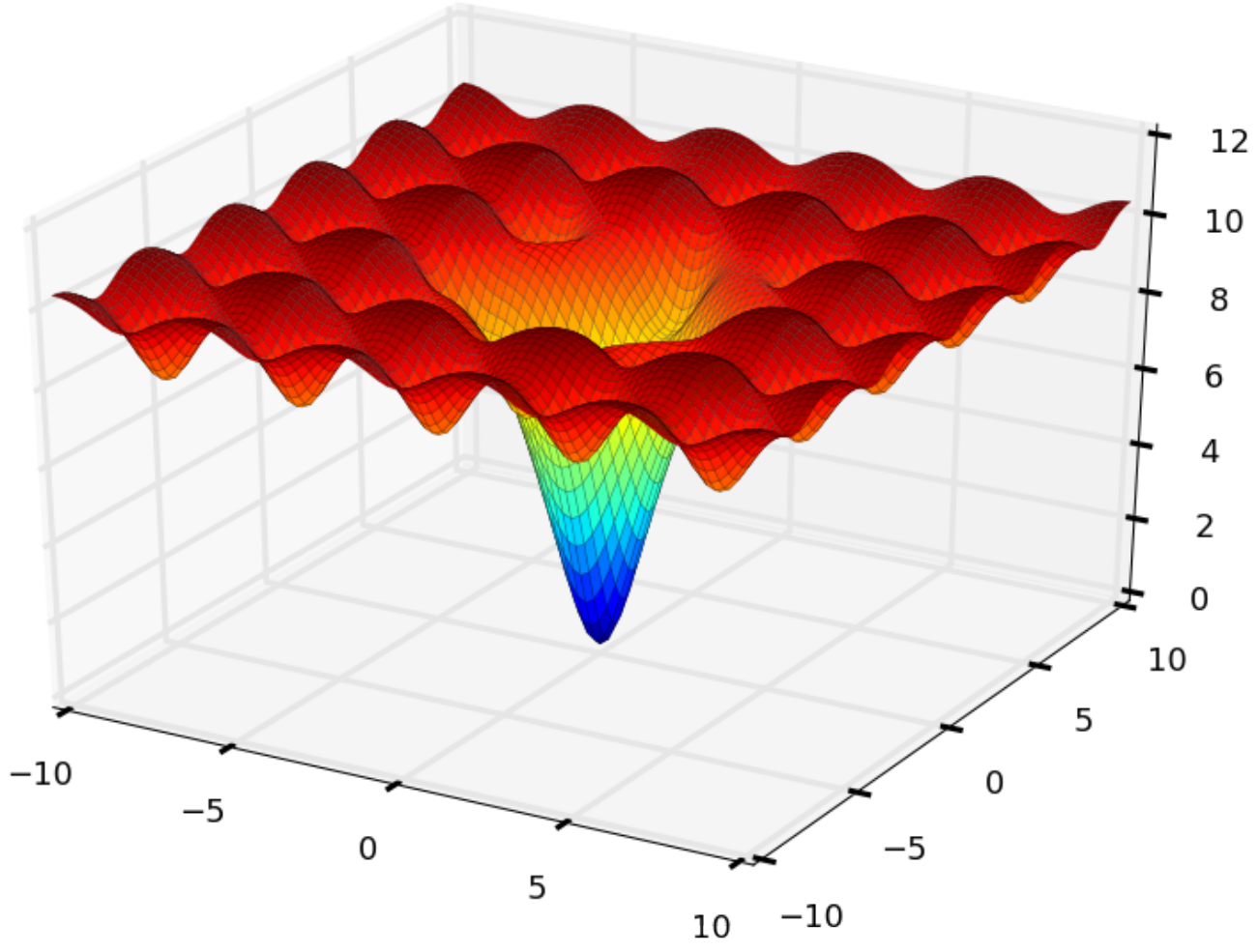
Create a *donor* by *differential mutation*

Crossover target and donor to form *trial*

Select next population

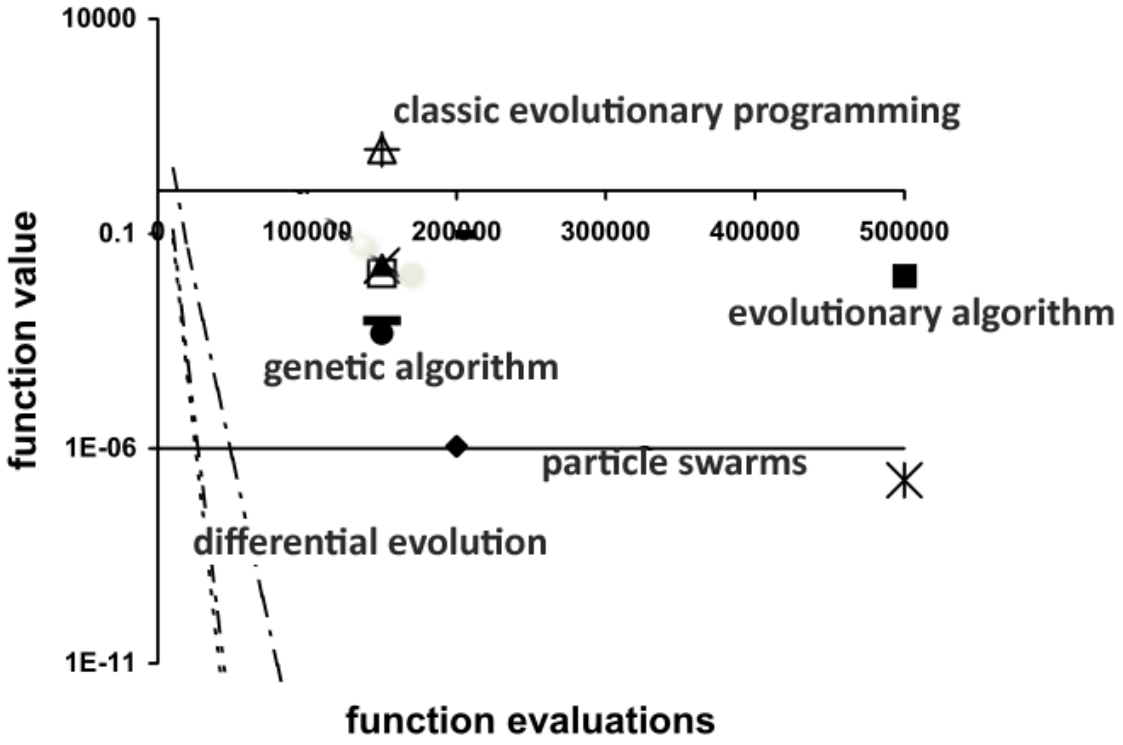
until stopping criterion reached

Ackley's function



Motivational benchmark

30-dimensional Ackley's function



[1]

[1] figure adapted from Price et al. [2006]

Overview: Initialization

Initialize population

repeat

  for each target **in** population

    Create a *donor* by *differential mutation*

    Crossover target and donor to form *trial*

  Select next population

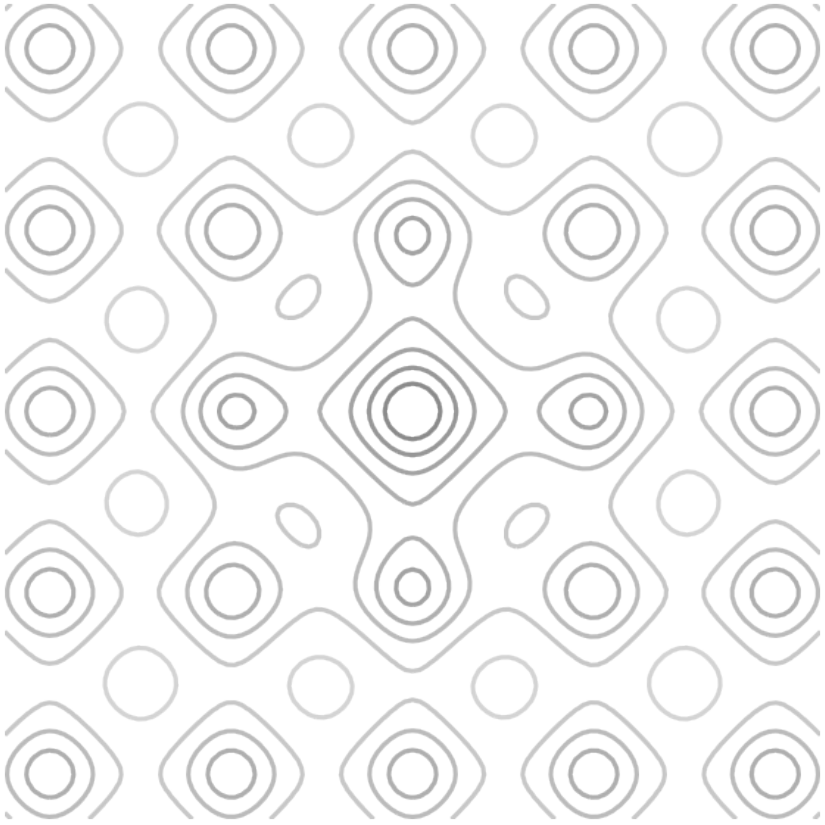
until stopping criterion reached

Initialization

- Initialize N D-dimensional population vectors t:

  $t = (t_1, t_2, ..., t_D), \quad t_i \sim PDF_i$

- Should ideally contain the global optimum



Example

$N = 6$

$PDF_i = \mathcal{U}(-9, 9)$,

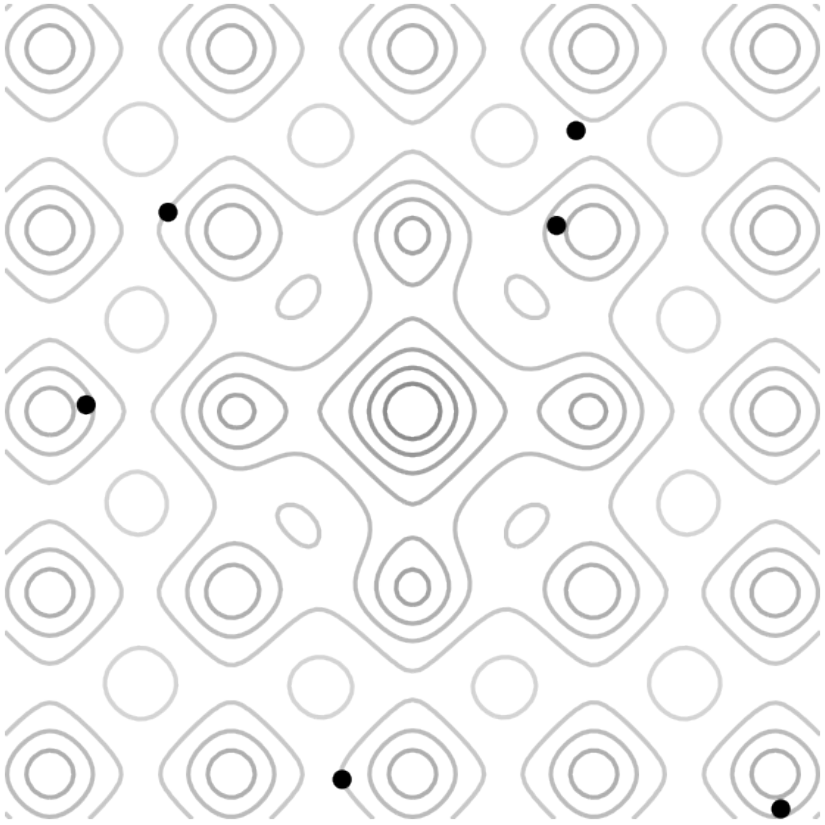
i.e. uniform over plot area for all components

Initialization

- Initialize N D-dimensional population vectors t:

  $t = (t_1, t_2, ..., t_D), \quad t_i \sim PDF_i$

- Should ideally contain the global optimum



Example

$N = 6$

$PDF_i = \mathcal{U}(-9, 9)$,

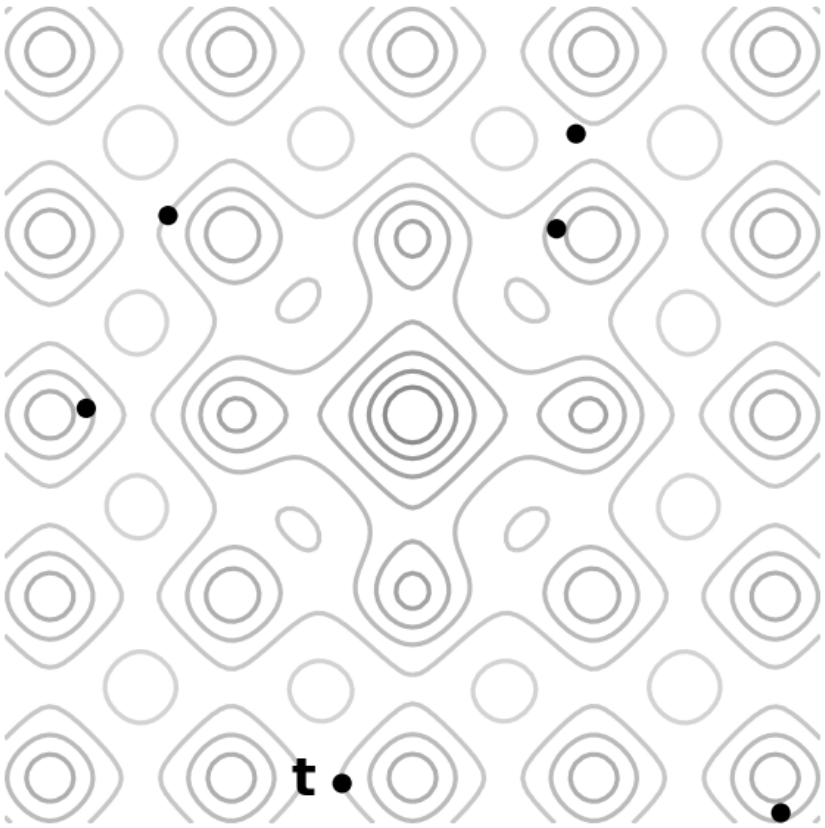
i.e. uniform over plot area for all components

Overview: Differential Mutation

Initialize population
repeat
    for each target **in** population
        Create a *donor* by *differential mutation*
        Crossover target and donor to form *trial*
    Select next population
until stopping criterion reached

Differential Mutation

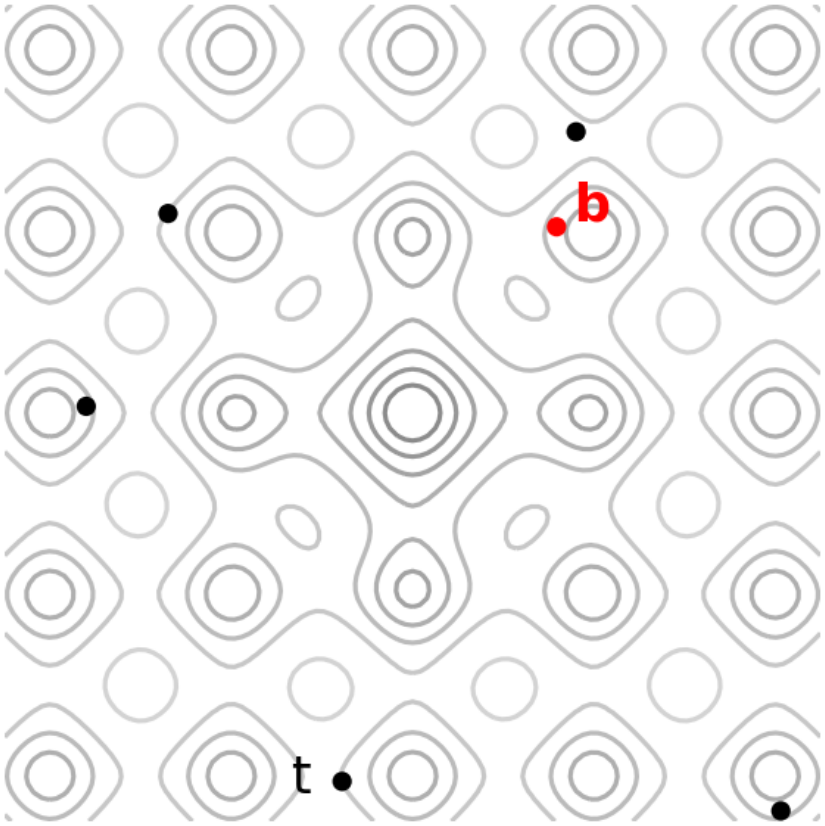
Generate a donor vector d for each target by differential mutation:



Differential Mutation

Generate a donor vector d for each target by differential mutation:

- Select *base* vector b



Example

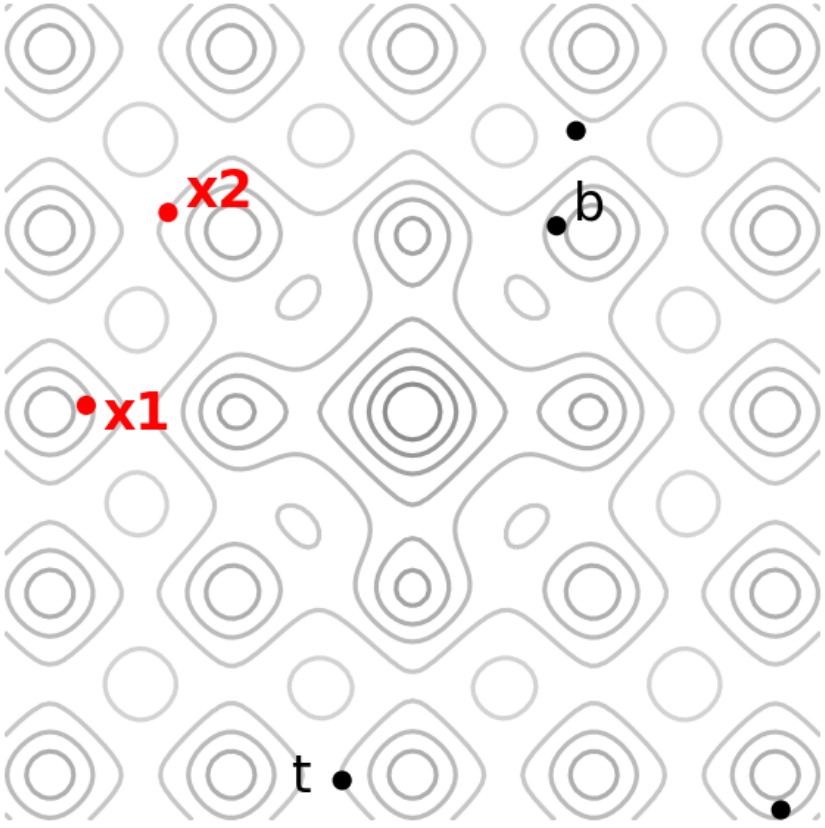
random selection,

$b \neq t$

Differential Mutation

Generate a donor vector d for each target by differential mutation:

- Select *base* vector b
- Select two different vectors x_1, x_2



Example

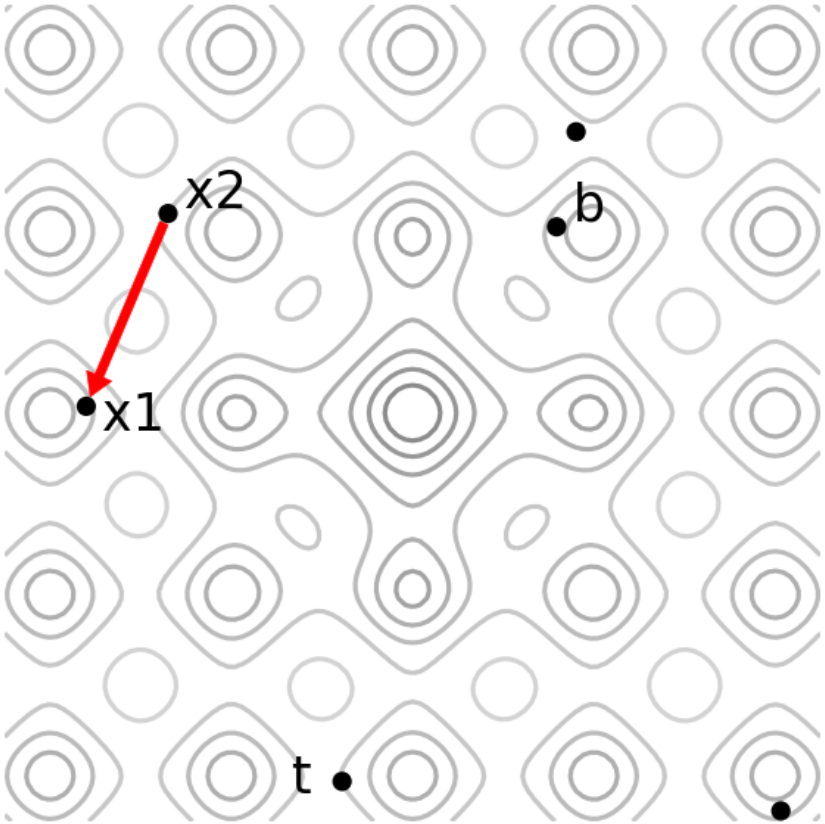
random selection,

x_1, x_2, b, t all distinct

Differential Mutation

Generate a donor vector d for each target by differential mutation:

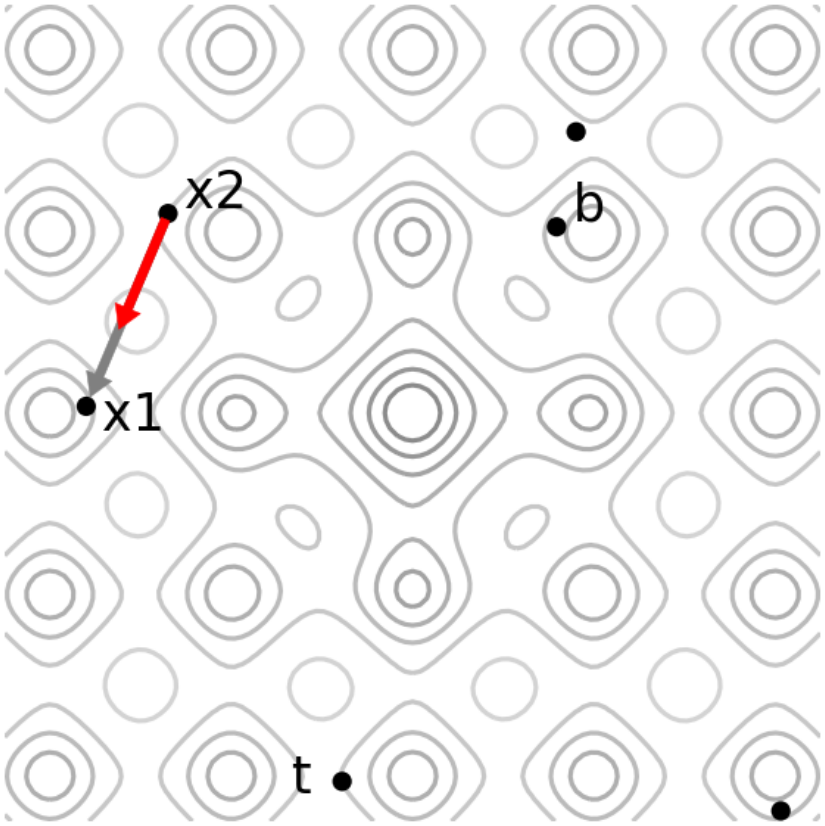
- Select *base* vector b
- Select two different vectors x_1, x_2
- Compute difference $x_1 - x_2$



Differential Mutation

Generate a donor vector d for each target by differential mutation:

- Select *base* vector b
- Select two different vectors x_1, x_2
- Compute difference $x_1 - x_2$
- Scale difference by scaling factor F



Example

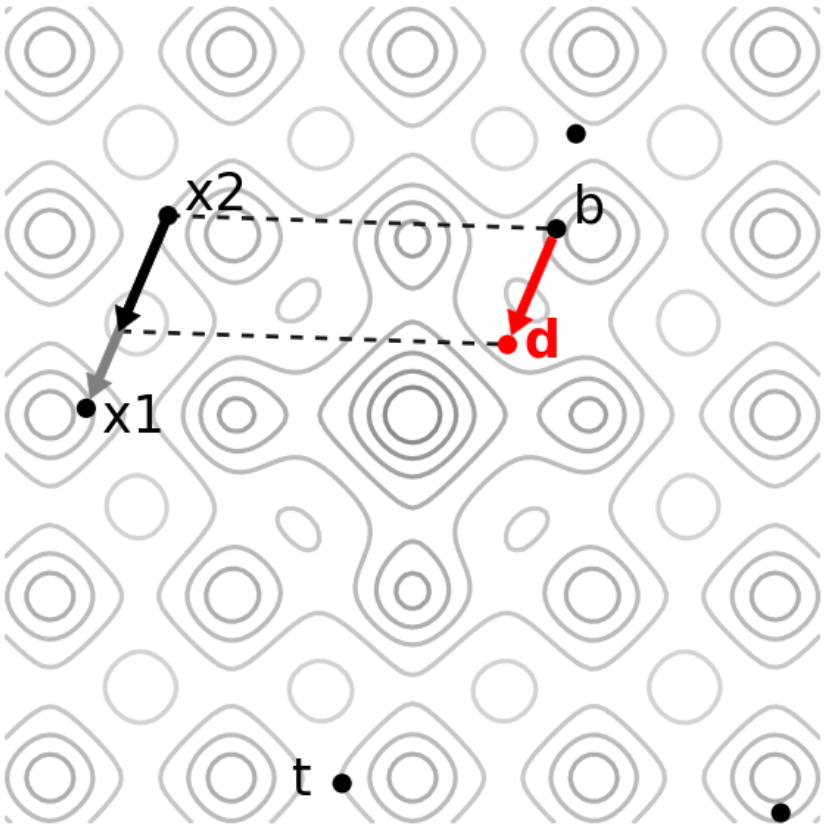
$F = 0.5$, but could be anything > 0

Differential Mutation

Generate a donor vector d for each target by differential mutation:

- Select *base* vector b
- Select two different vectors x_1, x_2
- Compute difference $x_1 - x_2$
- Scale difference by scaling factor F
- Add scaled difference to base

$$d = b + F(x_1 - x_2)$$

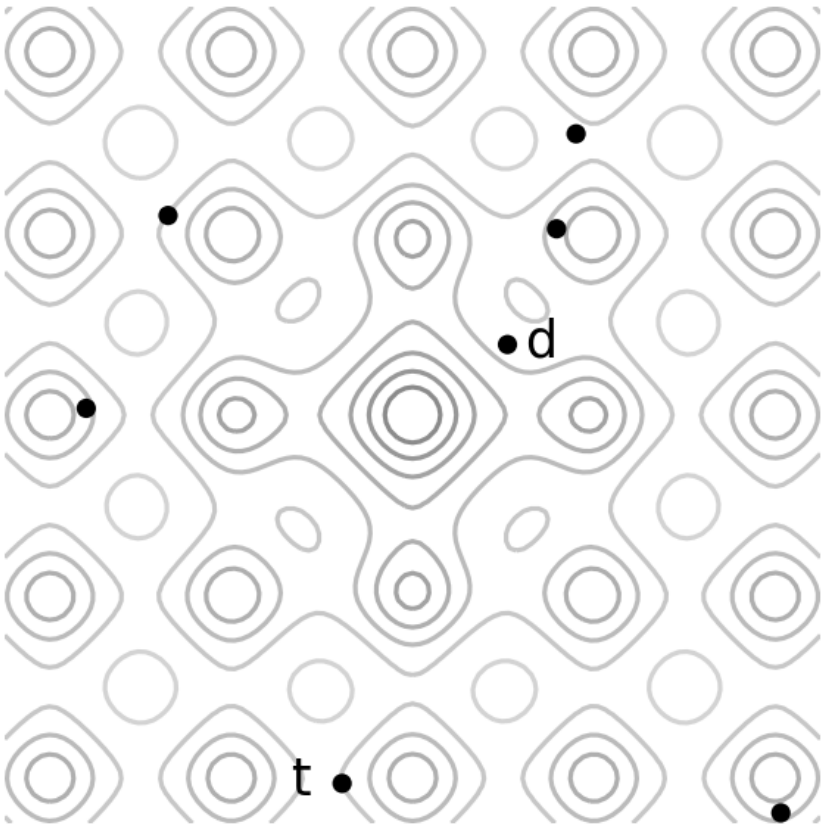


Differential Mutation

Generate a donor vector d for each target by differential mutation:

- Select *base* vector b
- Select two different vectors x_1, x_2
- Compute difference $x_1 - x_2$
- Scale difference by scaling factor F
- Add scaled difference to base

$$d = b + F(x_1 - x_2)$$



Overview: Crossover

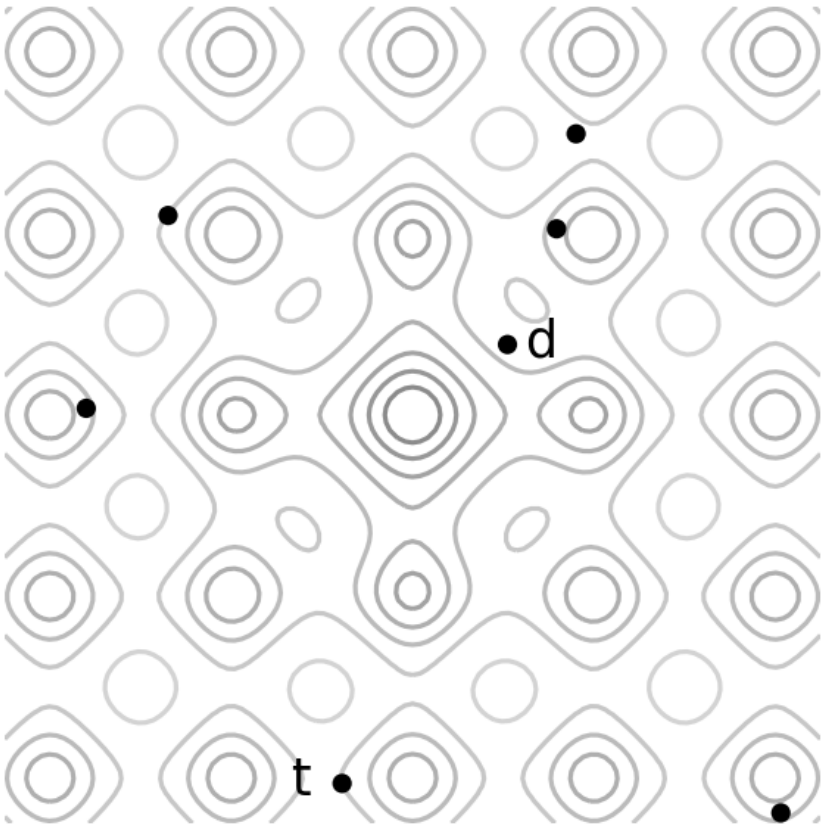
Initialize population
repeat
    for each target **in** population
        Create a *donor* by *differential mutation*
        Crossover target and donor to form *trial*
    Select next population
until stopping criterion reached

Binomial Crossover

Generate *trial* vector z for each donor d and target t:

- Pick $r \in [1..D]$ randomly
- Combine *target* with *donor*:

$$z_i = \begin{cases} d_i & \text{if } \mathrm{rand}(0,1) \leq CR \\ d_i & \text{else if } i = r \\ t_i & \text{otherwise} \end{cases}$$



Example

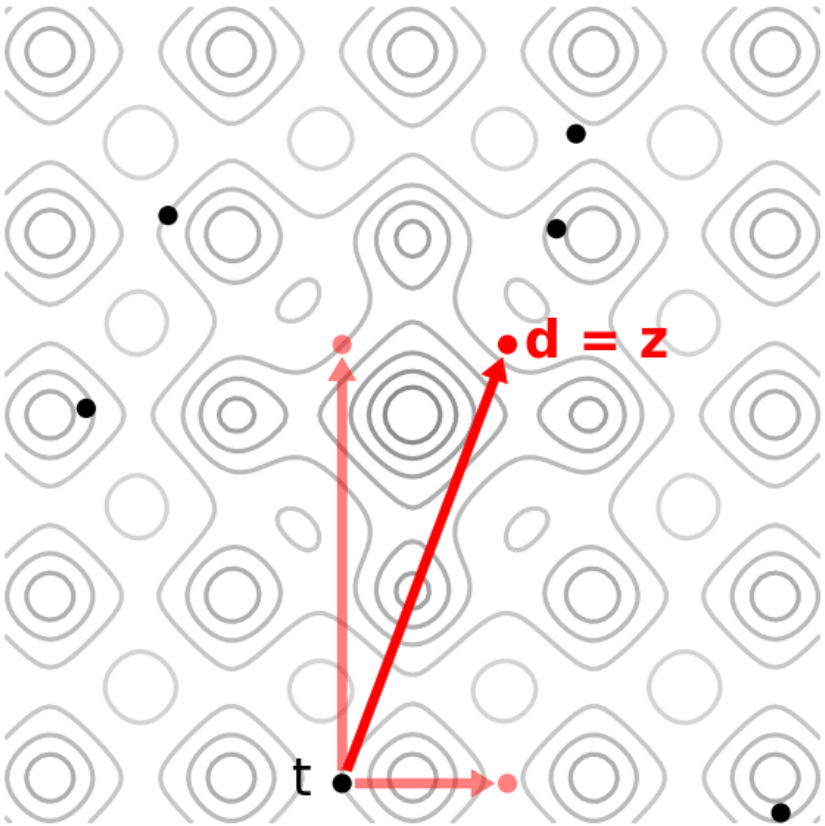
Crossover-rate $CR = 1$, other possibilites transparent

Binomial Crossover

Generate *trial* vector z for each donor d and target t:

- Pick $r \in [1..D]$ randomly
- Combine *target* with *donor*:

$$z_i = \begin{cases} d_i & \text{if } \mathrm{rand}(0,1) \leq CR \\ d_i & \text{else if } i = r \\ t_i & \text{otherwise} \end{cases}$$



Example

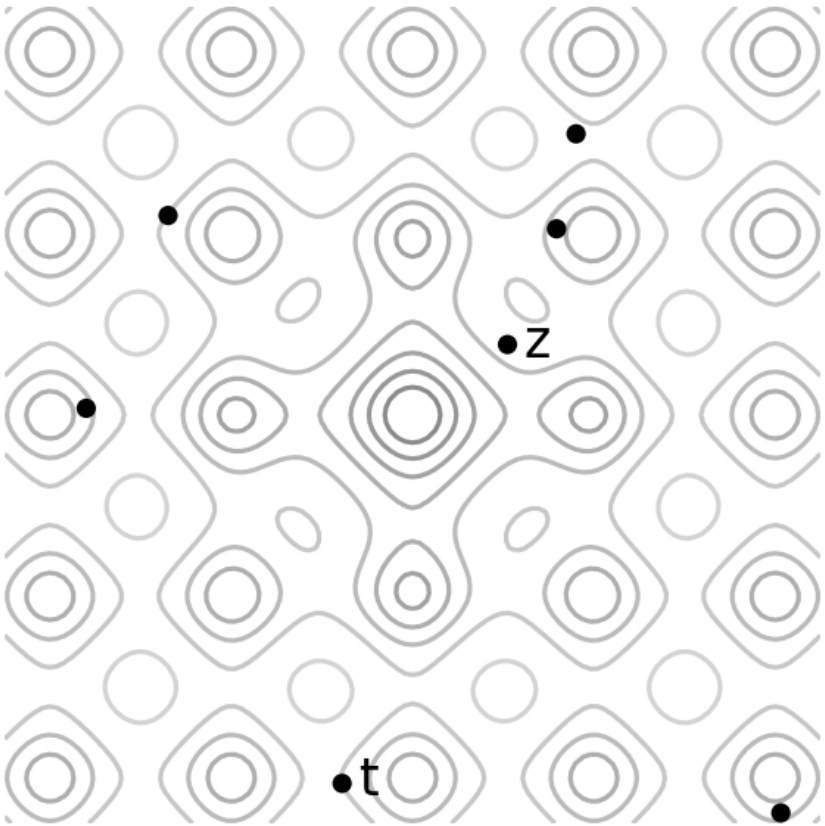
Crossover-rate $CR = 1$, other possibilites transparent

Binomial Crossover

Generate *trial* vector z for each donor d and target t:

- Pick $r \in [1..D]$ randomly
- Combine *target* with *donor*:

$$z_i = \begin{cases} d_i & \text{if } \mathrm{rand}(0,1) \leq CR \\ d_i & \text{else if } i = r \\ t_i & \text{otherwise} \end{cases}$$



Example

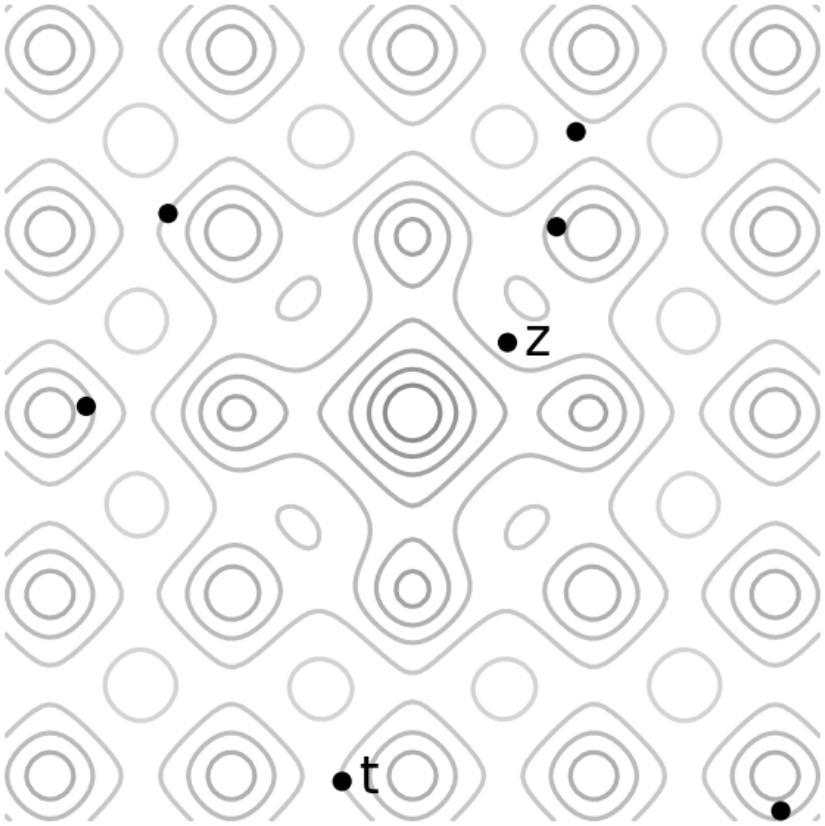
Crossover-rate $CR = 1$, other possibilites transparent

Overview: Selection

Initialize population
repeat
    for each target **in** population
        Create a *donor* by *differential mutation*
        Crossover target and donor to form *trial*
    Select next population
until stopping criterion reached

Selection

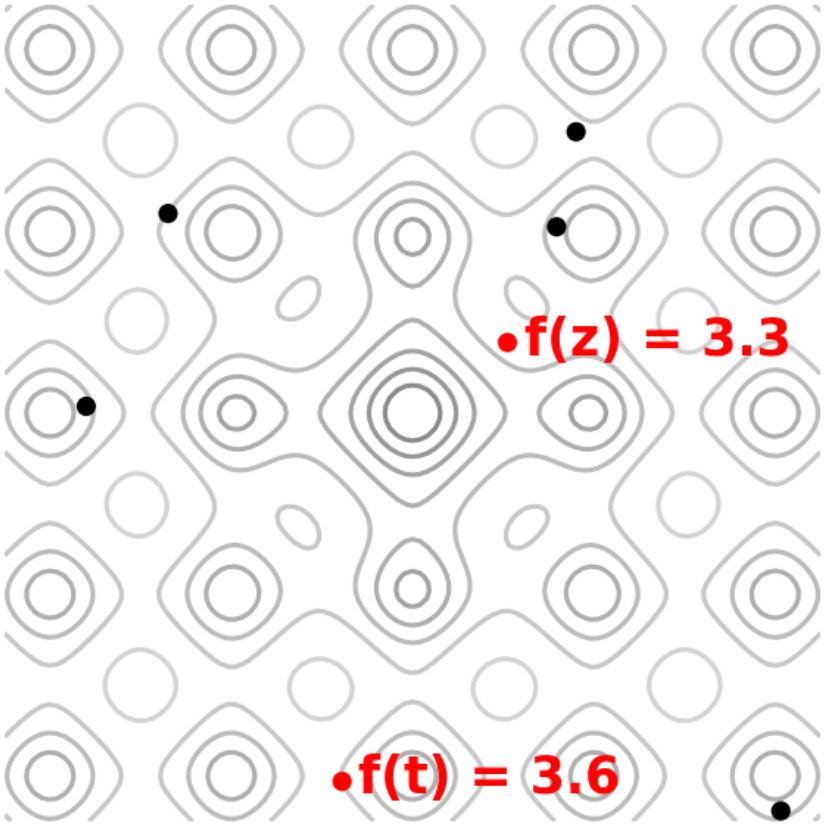
Select the population for the next iteration:



Selection

Select the population for the next iteration:

- Evaluate each target and corresponding trial vector

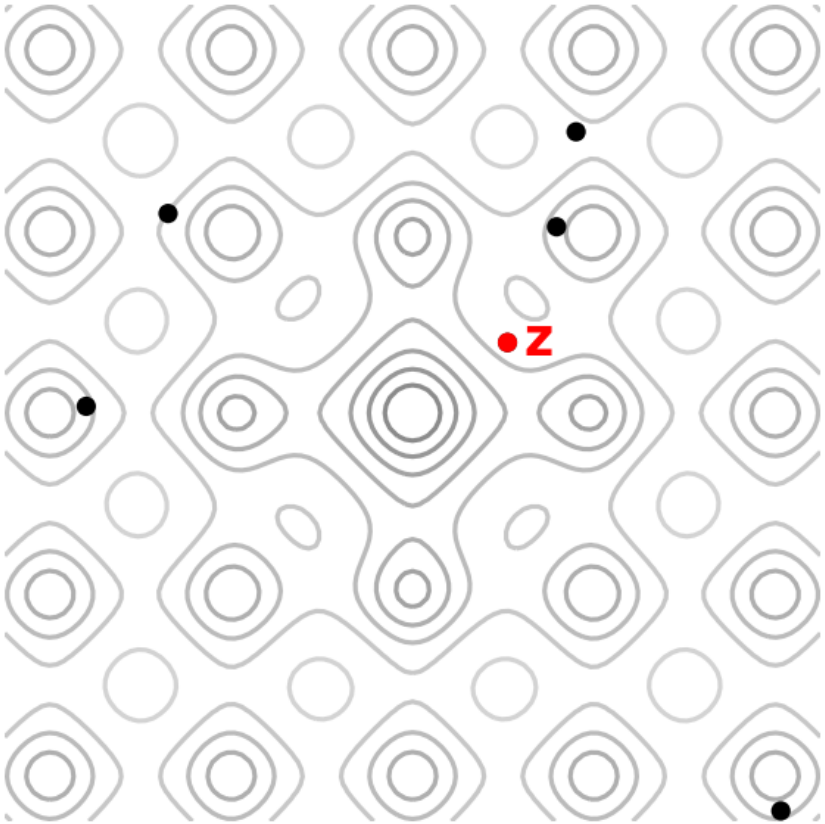


Selection

Select the population for the next iteration:

- Evaluate each target and corresponding trial vector
- If trial is better, replace target in next iteration

$$t_{next} = \begin{cases} z & \text{if } f(z) < f(t) \\ t & \text{otherwise} \end{cases}$$

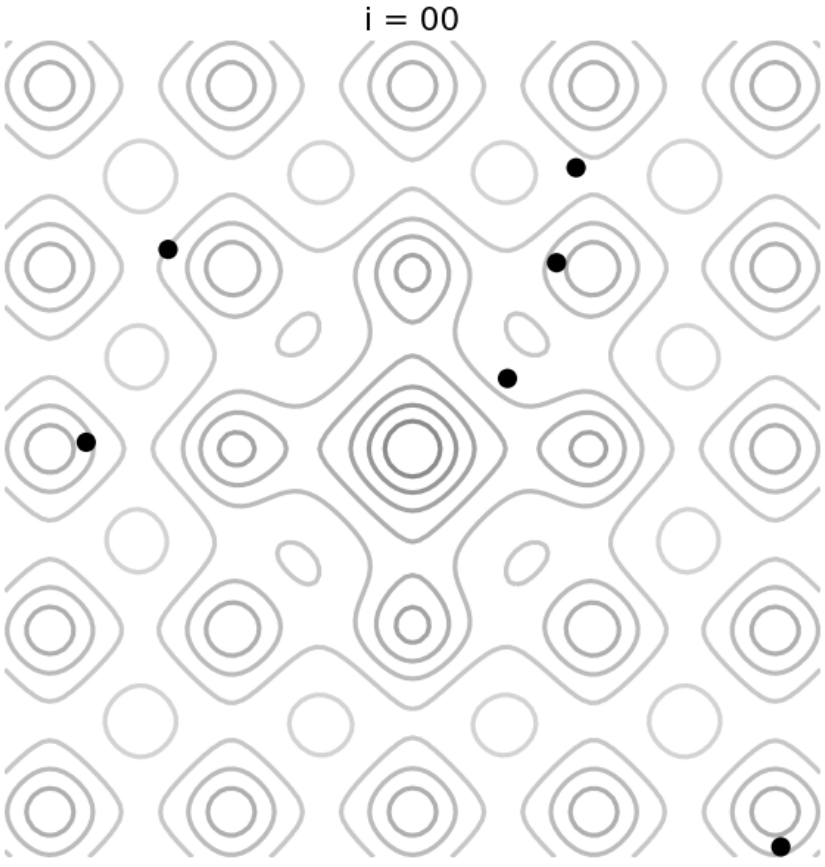


Overview

Initialize population
repeat
    for each target **in** population
        Create a *donor* by *differential mutation*
        Crossover target and donor to form *trial*
    Select next population
until stopping criterion reached

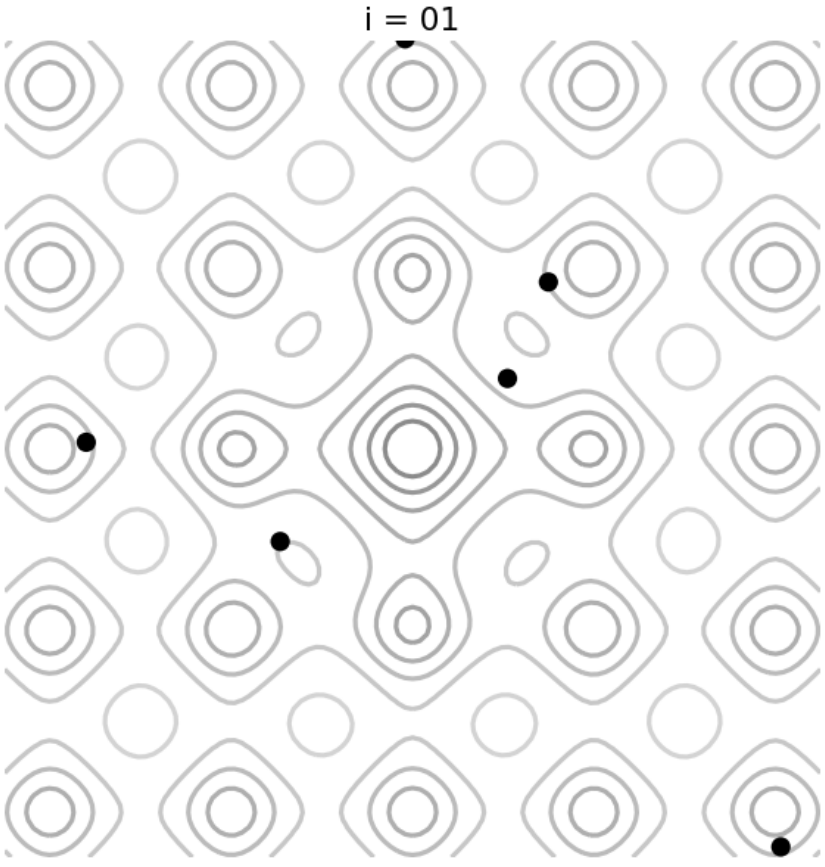
Demo

The population eventually converges to the global optimum



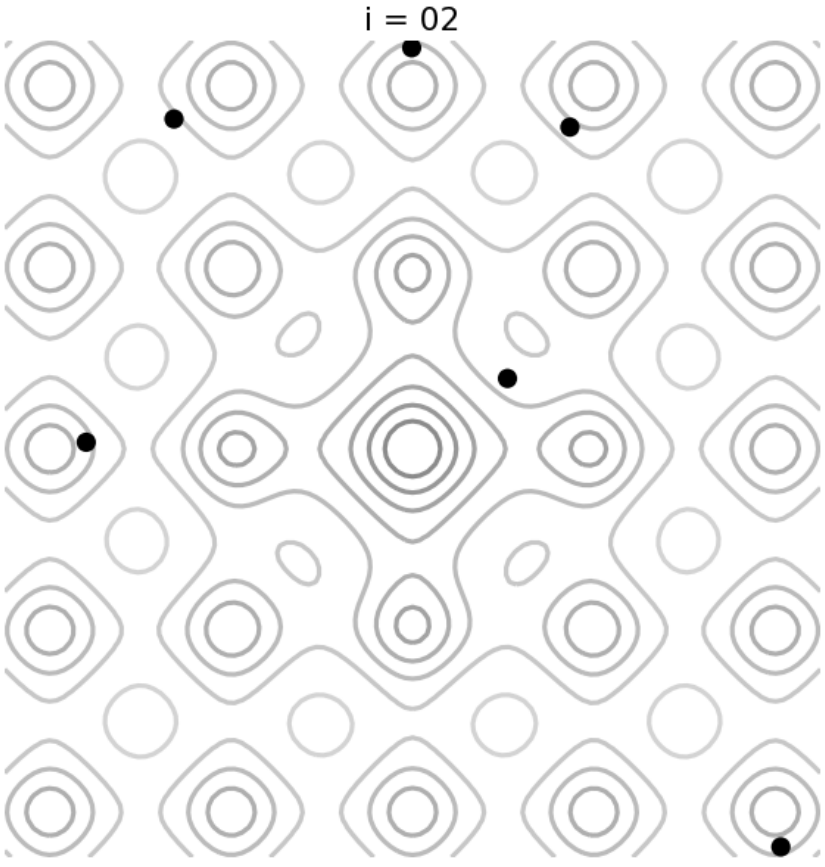
Demo

The population eventually converges to the global optimum



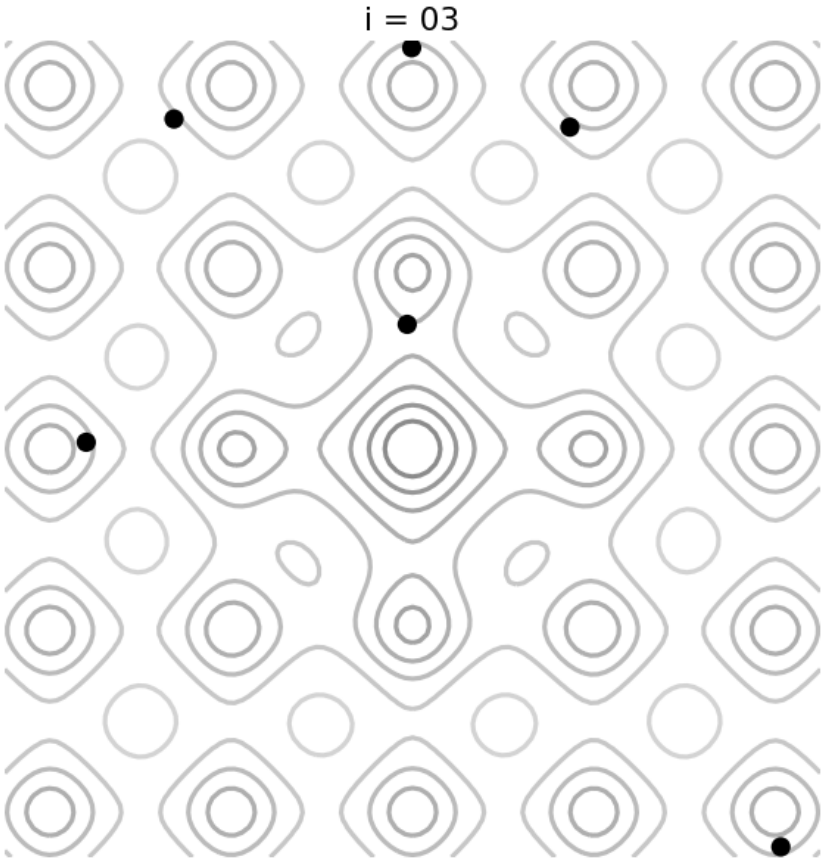
Demo

The population eventually converges to the global optimum



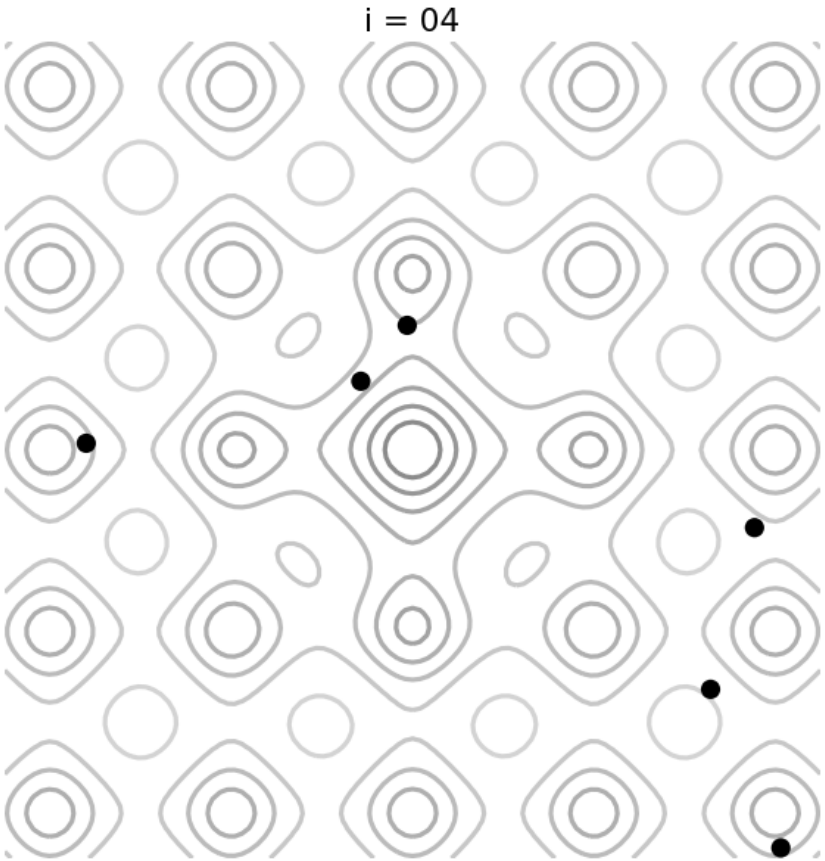
Demo

The population eventually converges to the global optimum



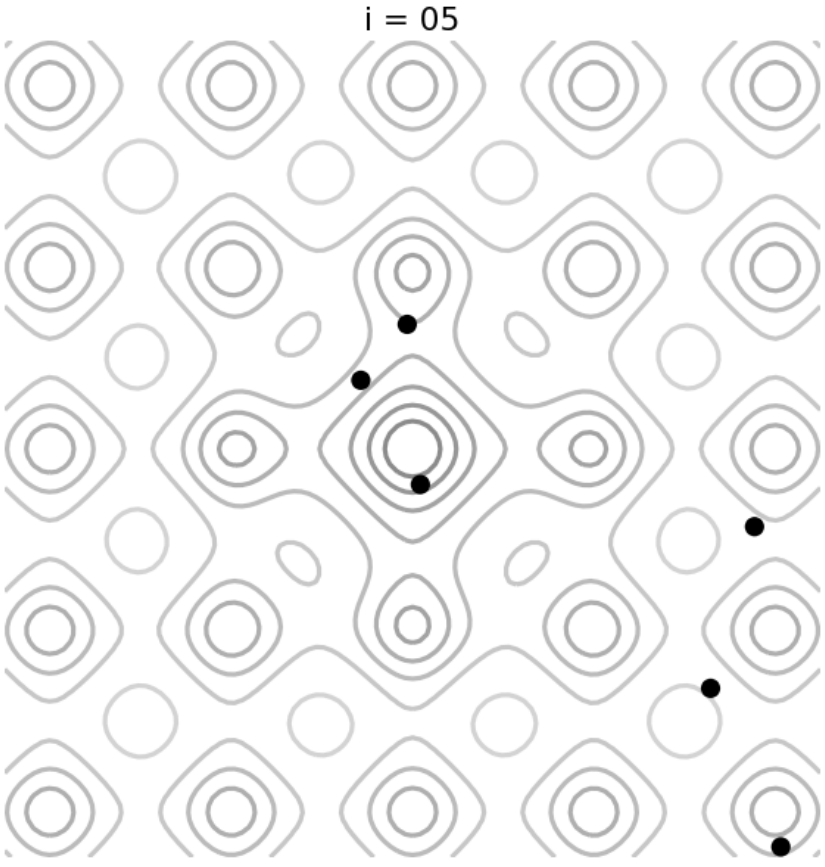
Demo

The population eventually converges to the global optimum



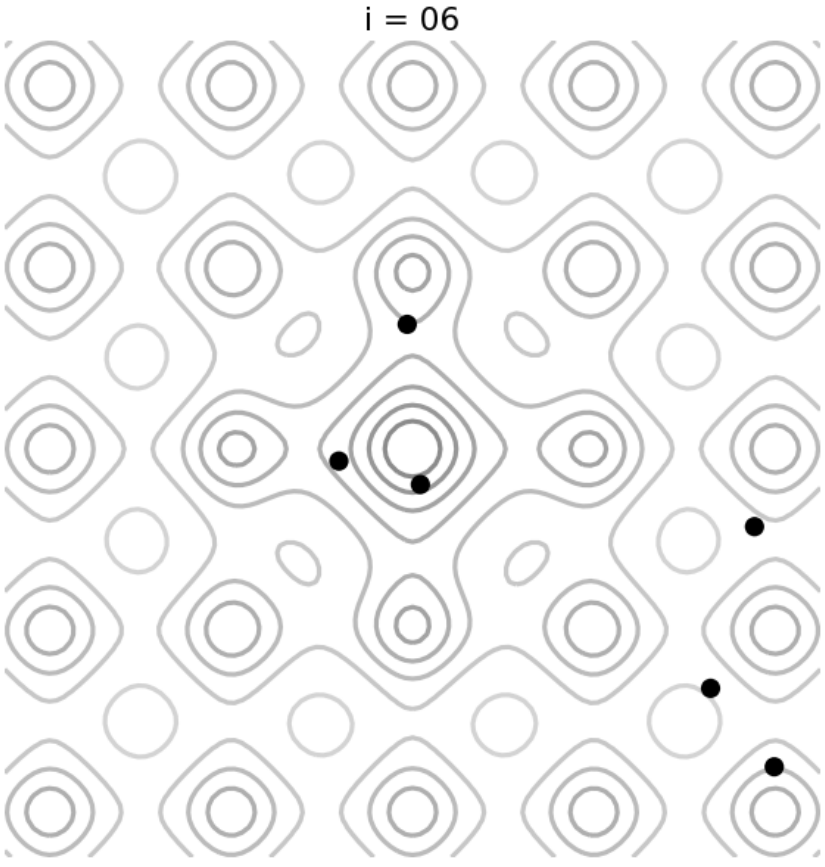
Demo

The population eventually converges to the global optimum



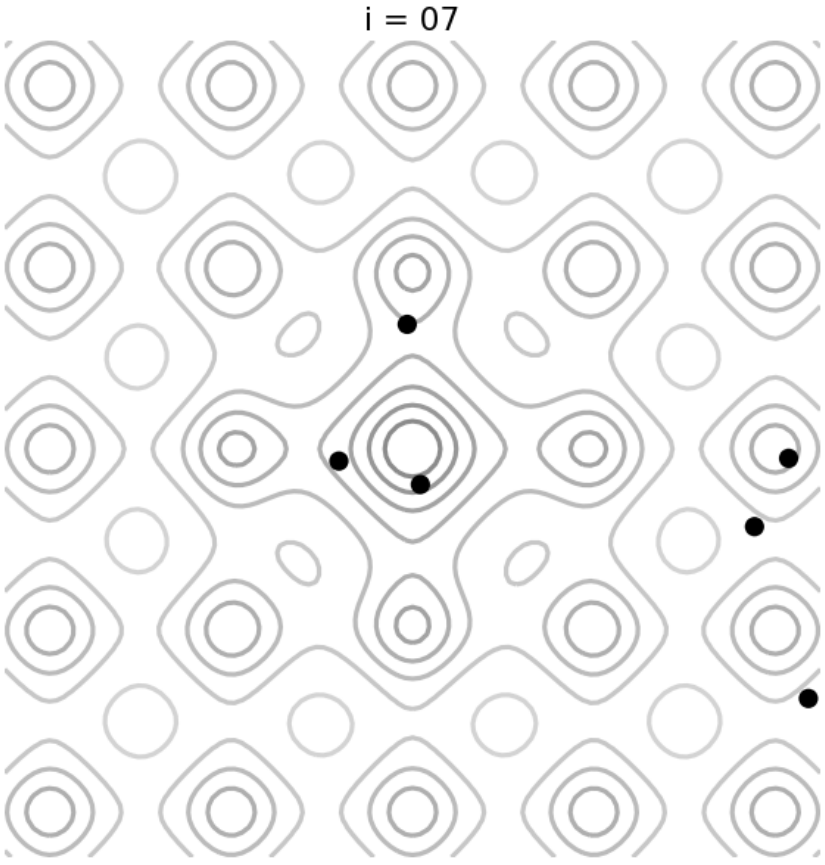
Demo

The population eventually converges to the global optimum



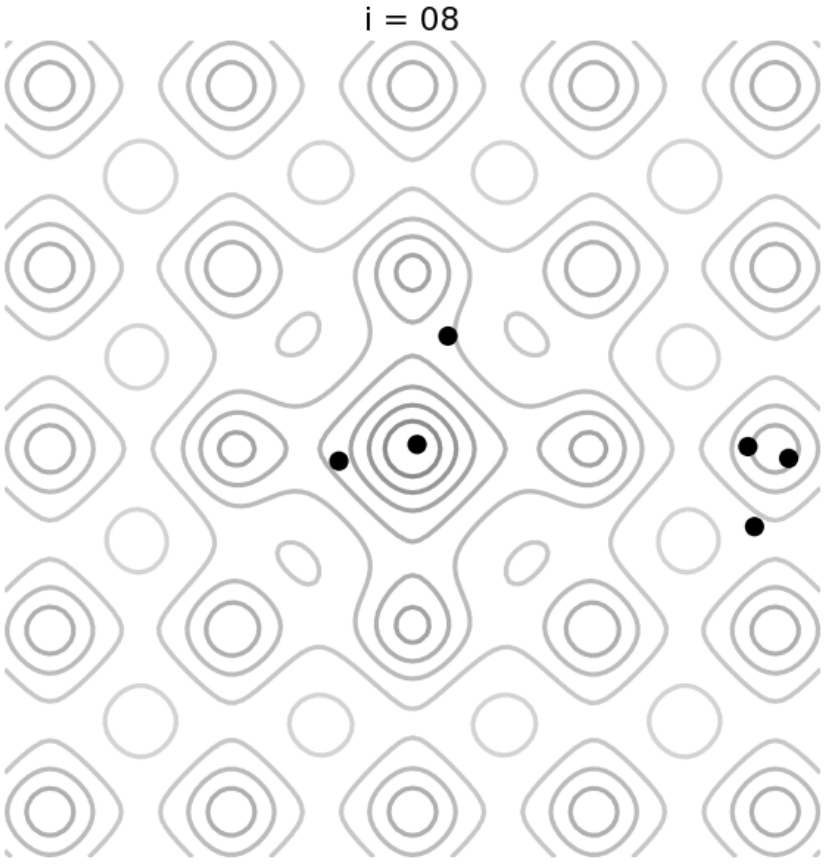
Demo

The population eventually converges to the global optimum



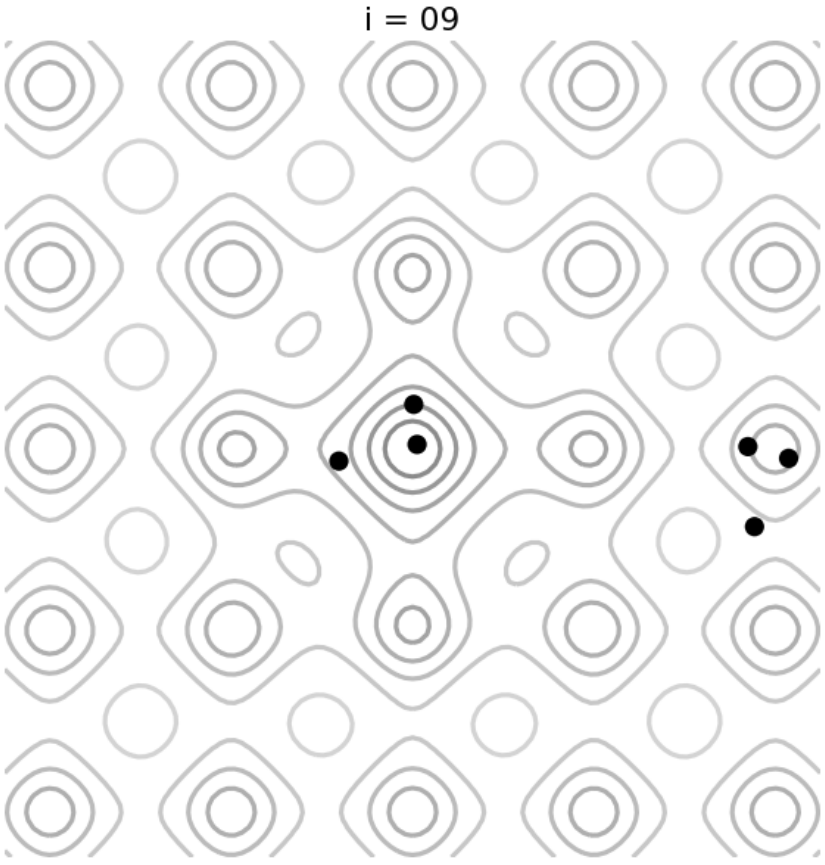
Demo

The population eventually converges to the global optimum



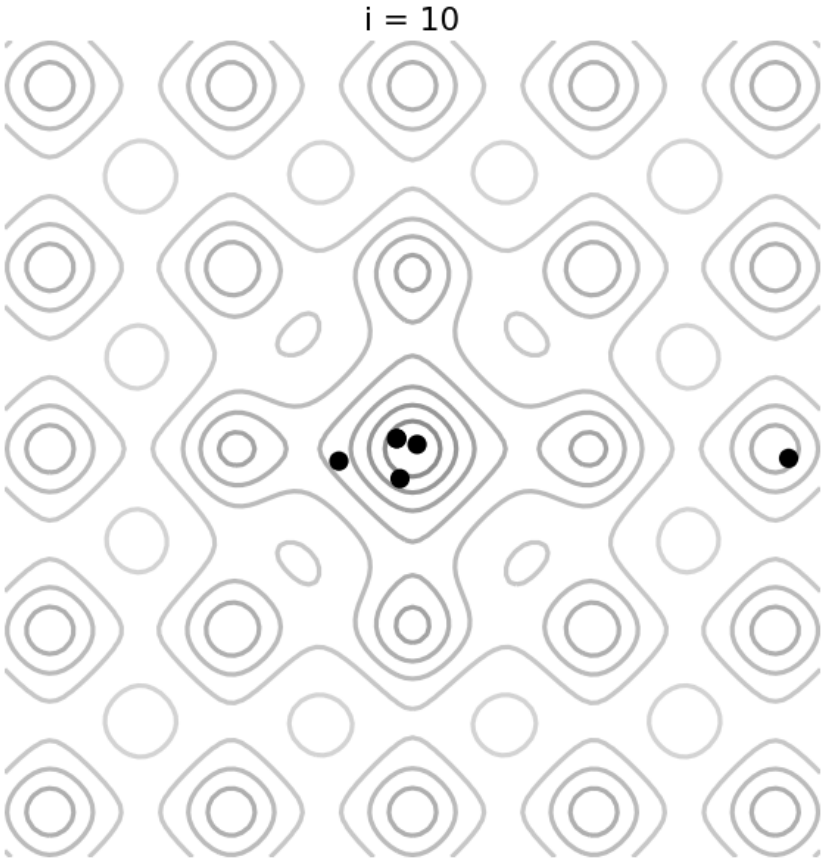
Demo

The population eventually converges to the global optimum



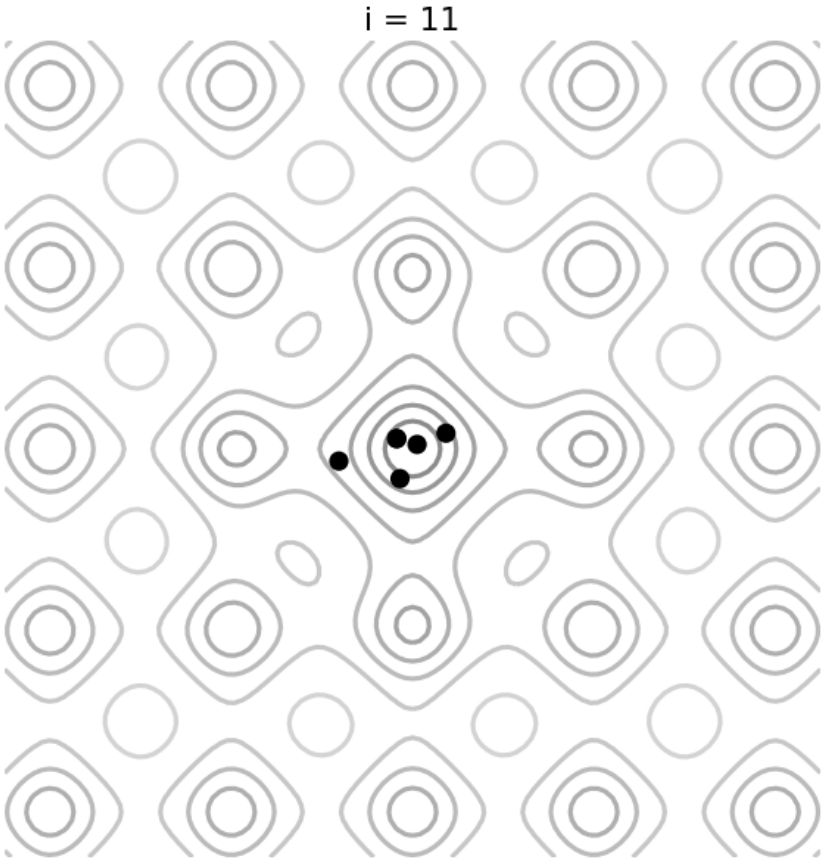
Demo

The population eventually converges to the global optimum



Demo

The population eventually converges to the global optimum

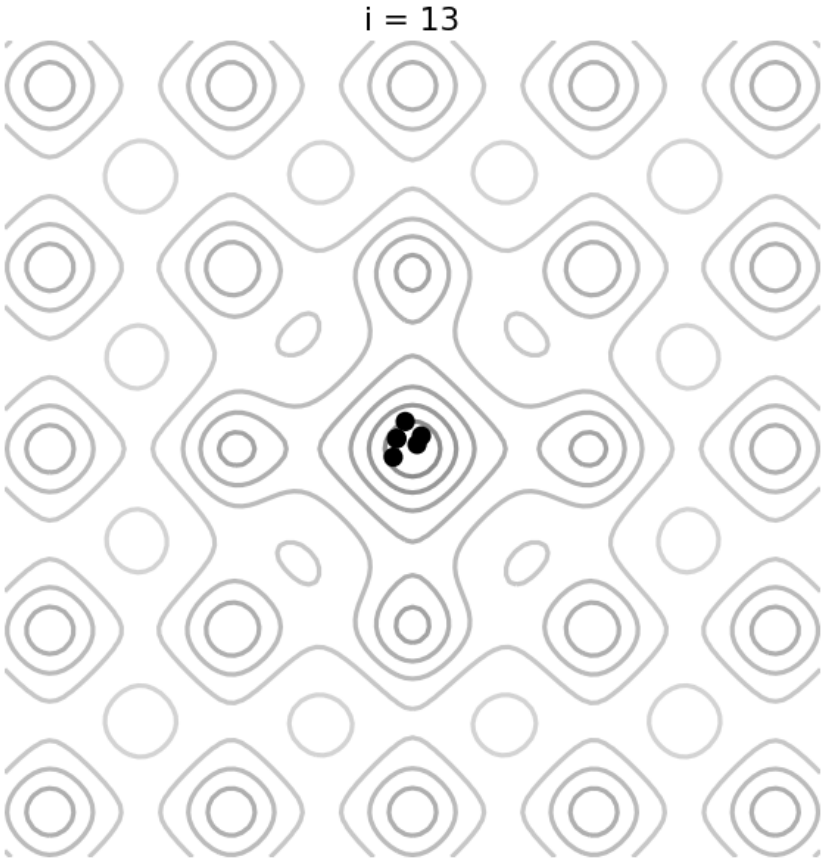


Demo

The population eventually converges to the global optimum

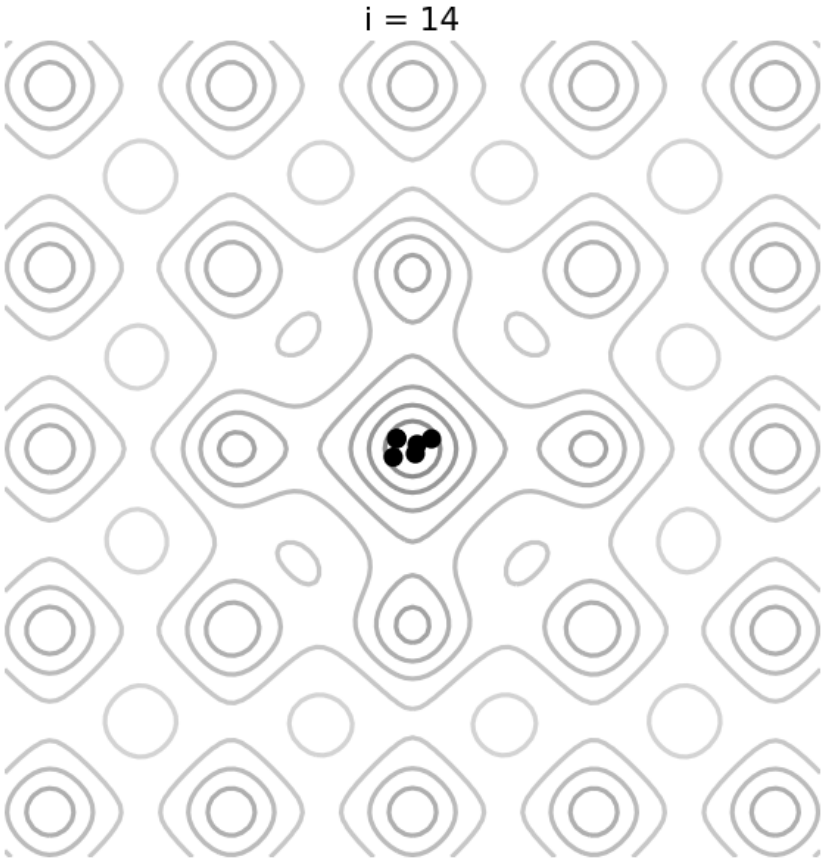
Demo

The population eventually converges to the global optimum



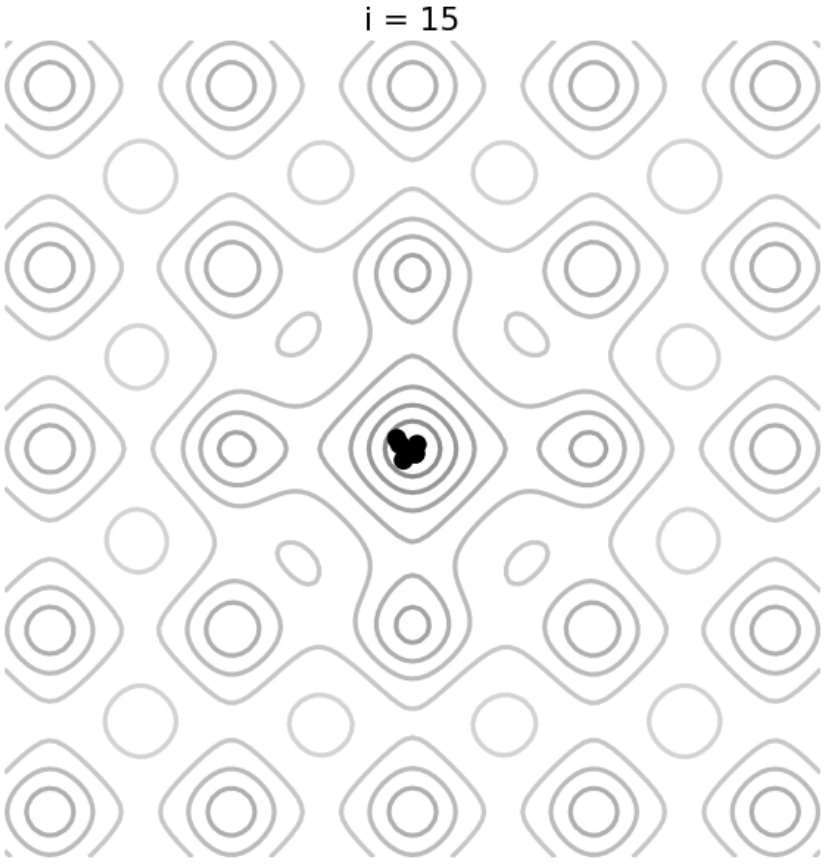
Demo

The population eventually converges to the global optimum



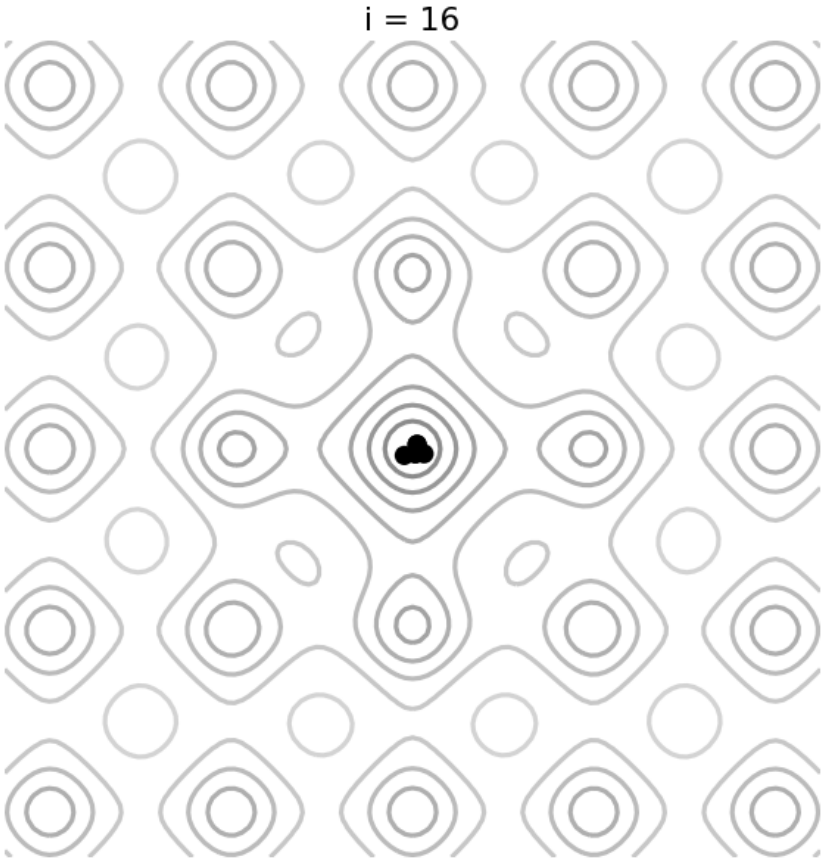
Demo

The population eventually converges to the global optimum



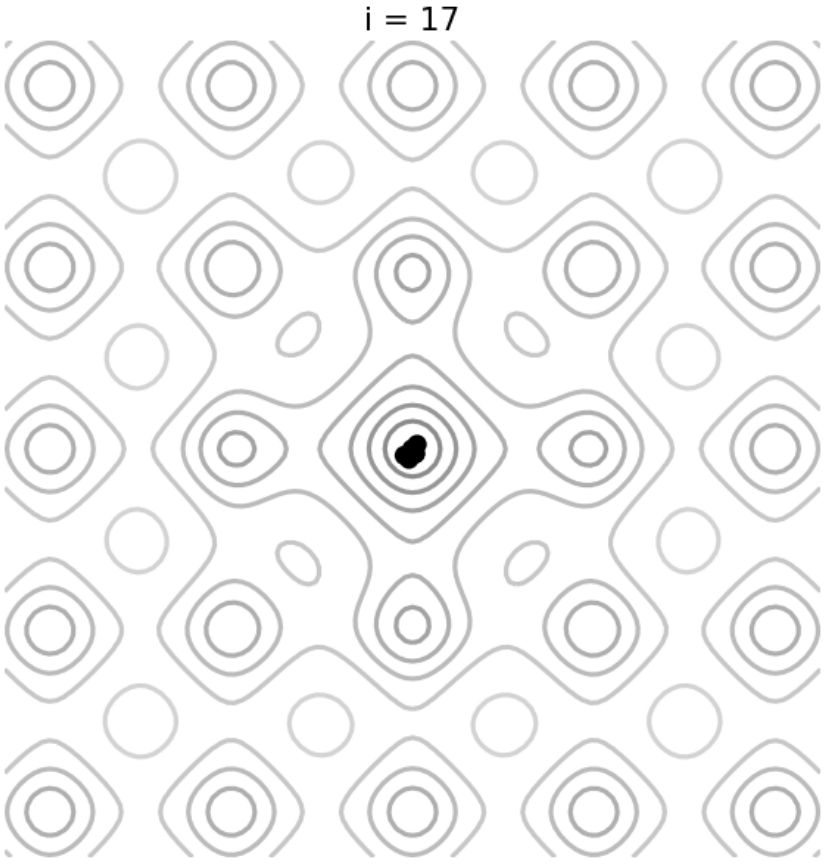
Demo

The population eventually converges to the global optimum



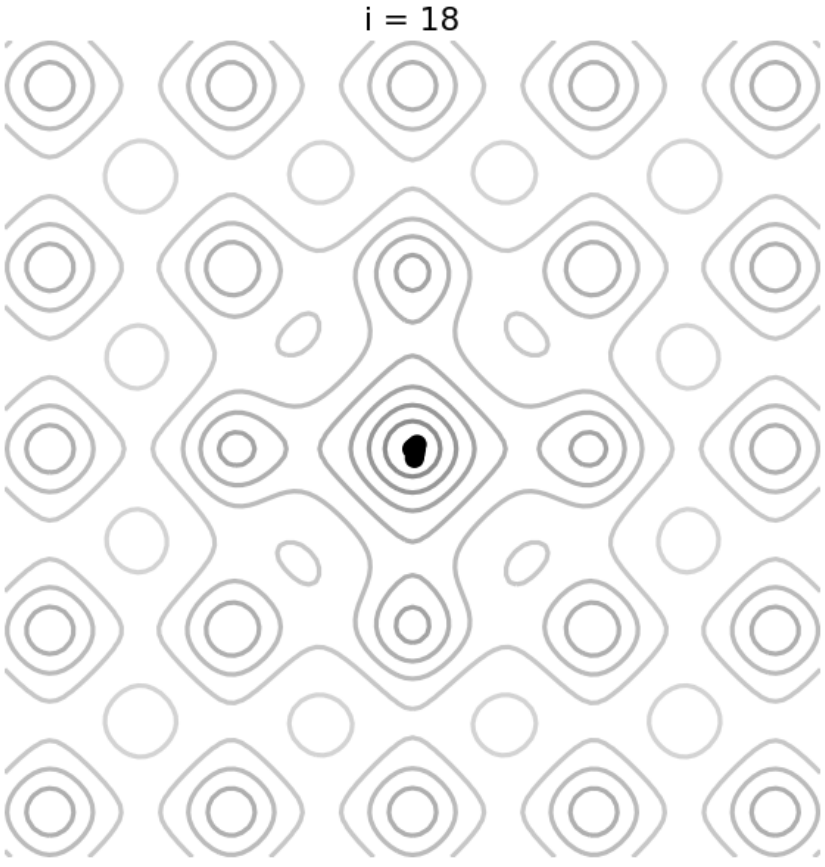
Demo

The population eventually converges to the global optimum



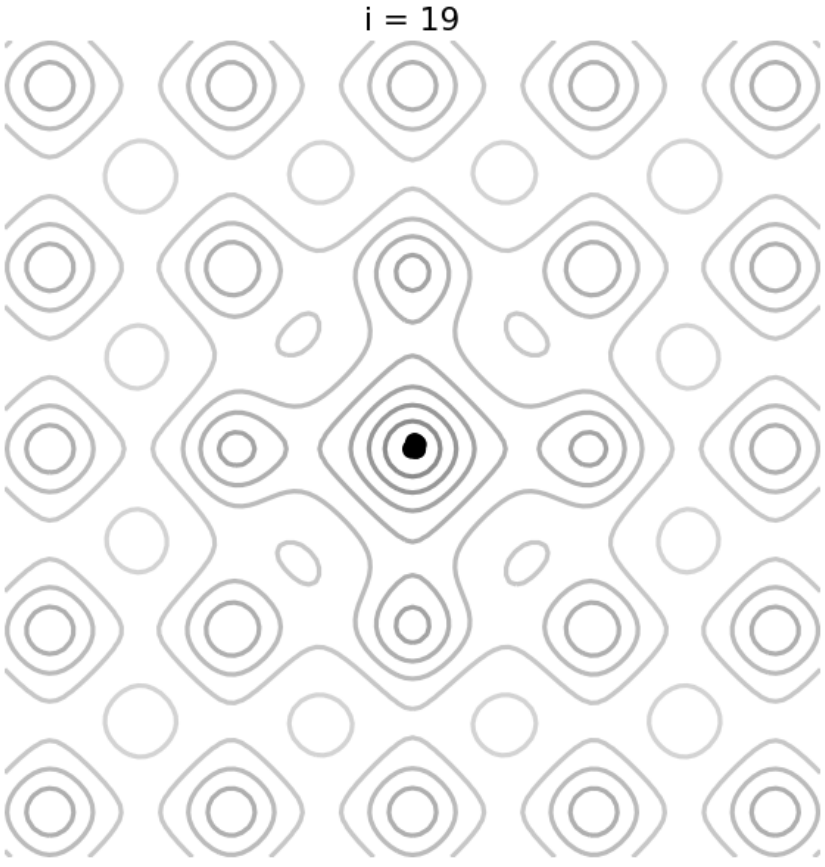
Demo

The population eventually converges to the global optimum



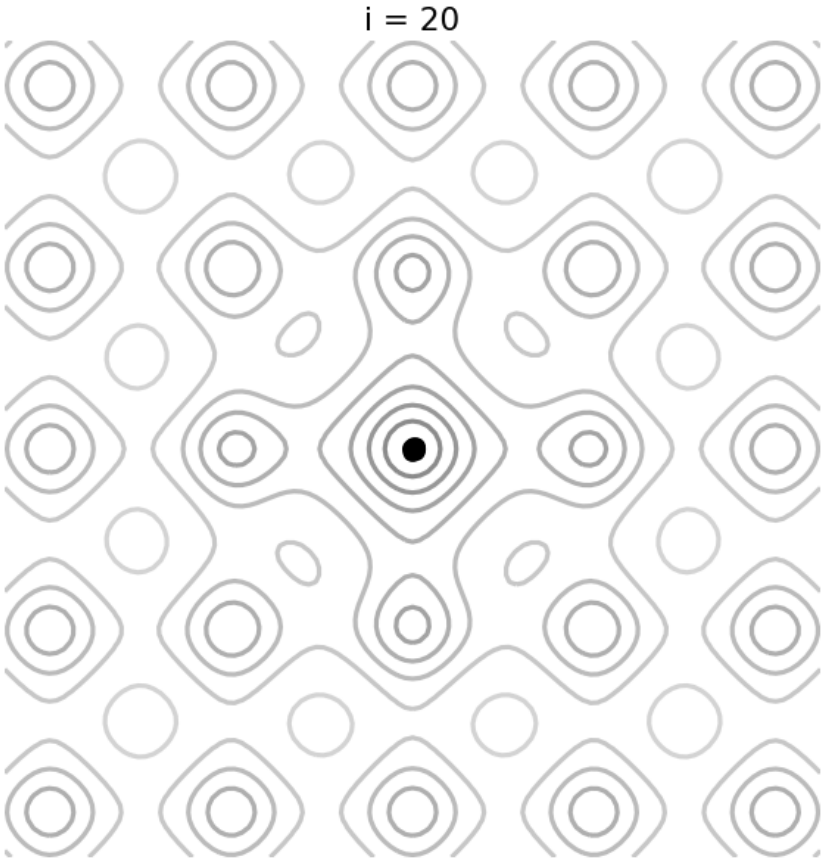
Demo

The population eventually converges to the global optimum



Demo

The population eventually converges to the global optimum

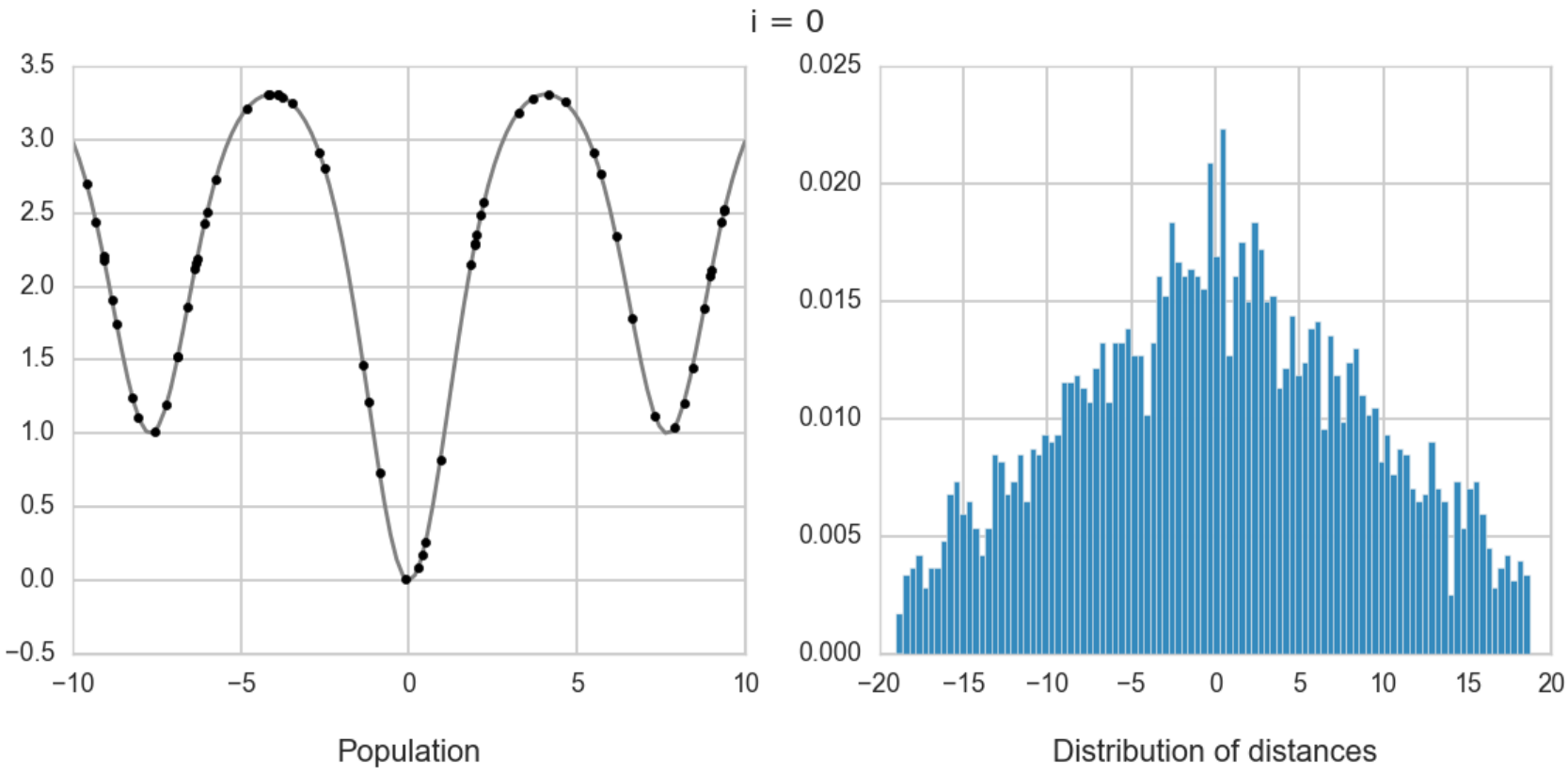


Contour matching

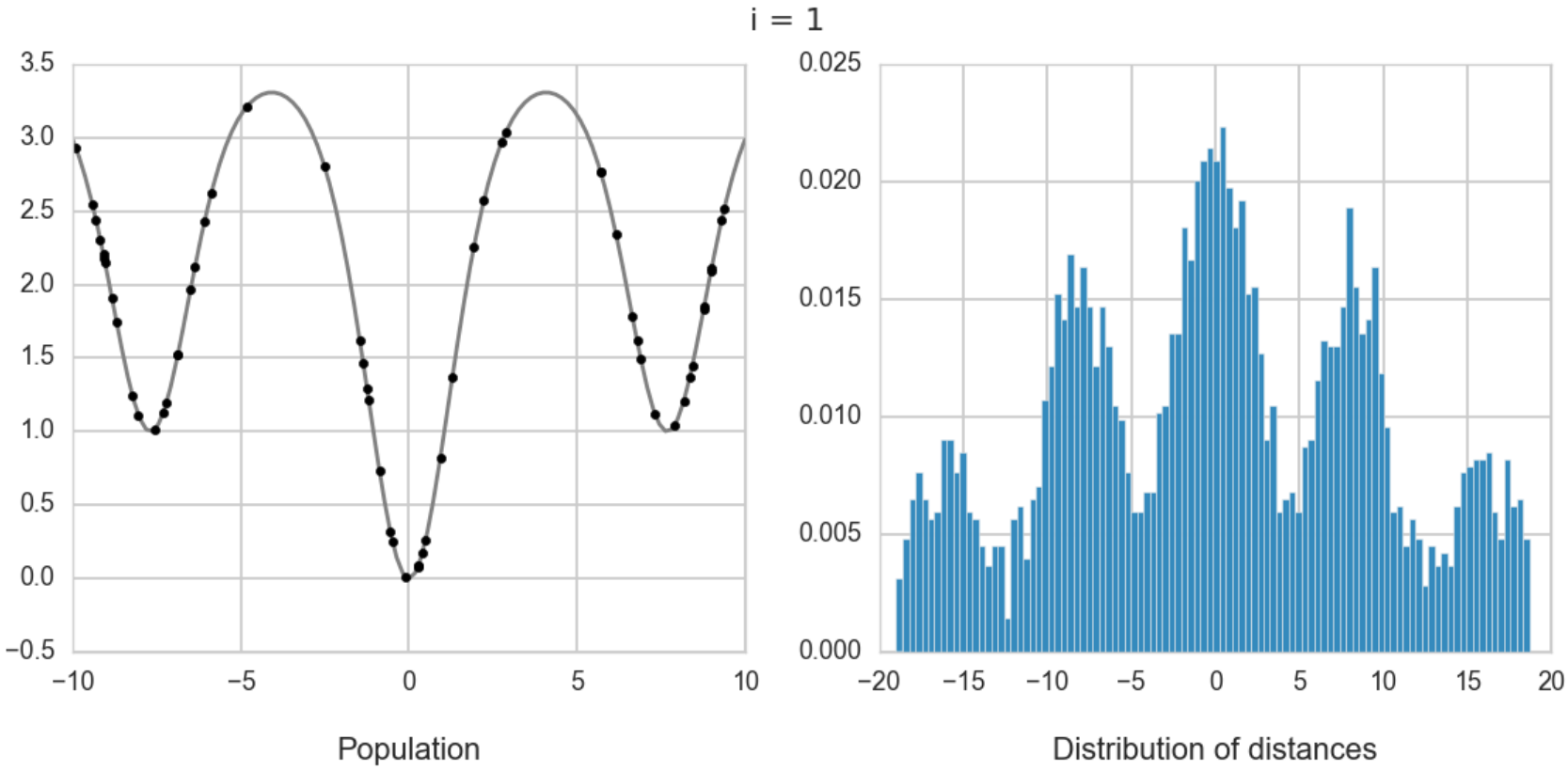
DE "learns" about the objective function by distributing the population along the *contours* of the objective function.

- Automatically adapt step size and orientation
- Enable jumps between different *basins of attraction*

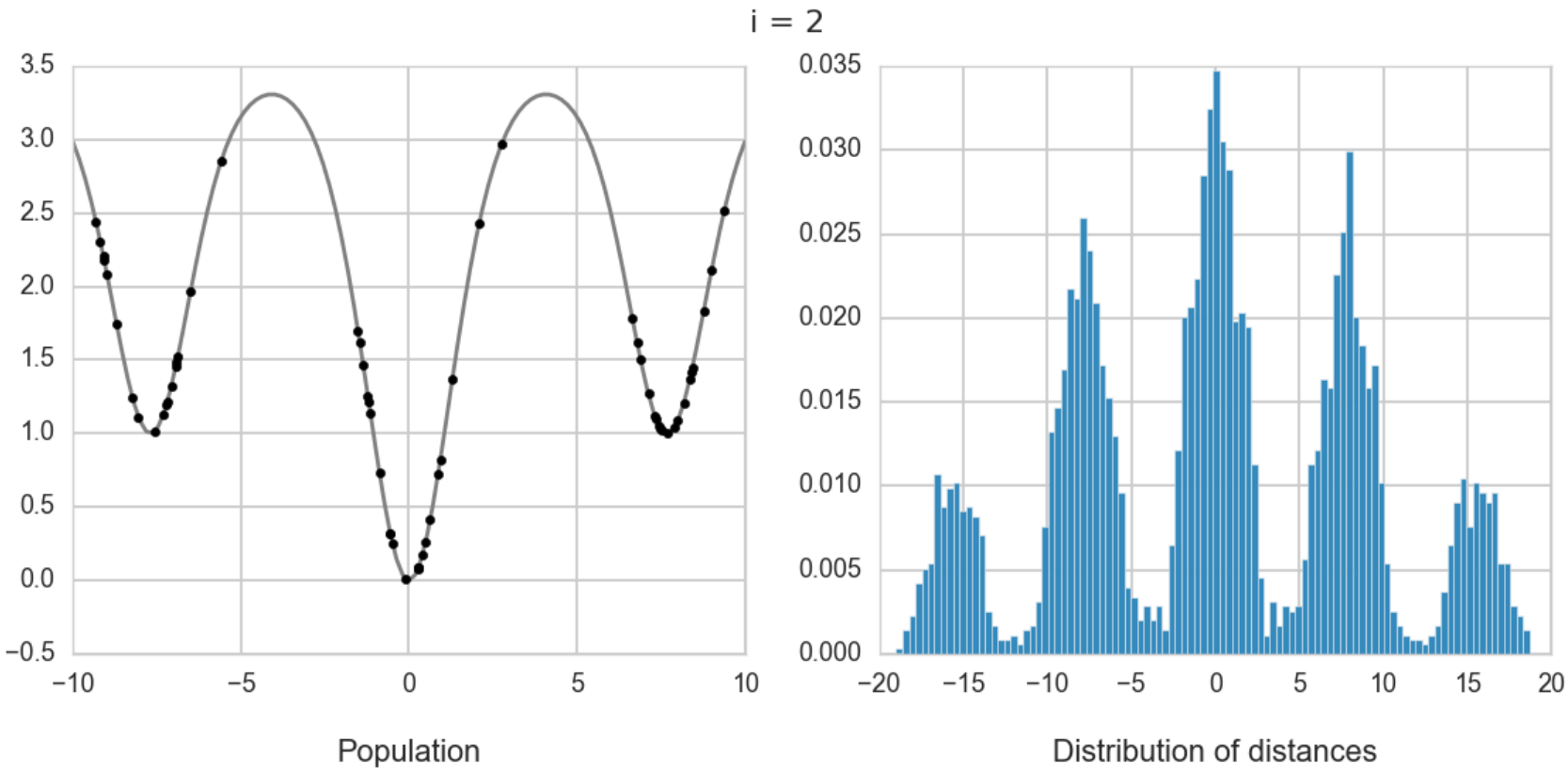
Contour matching



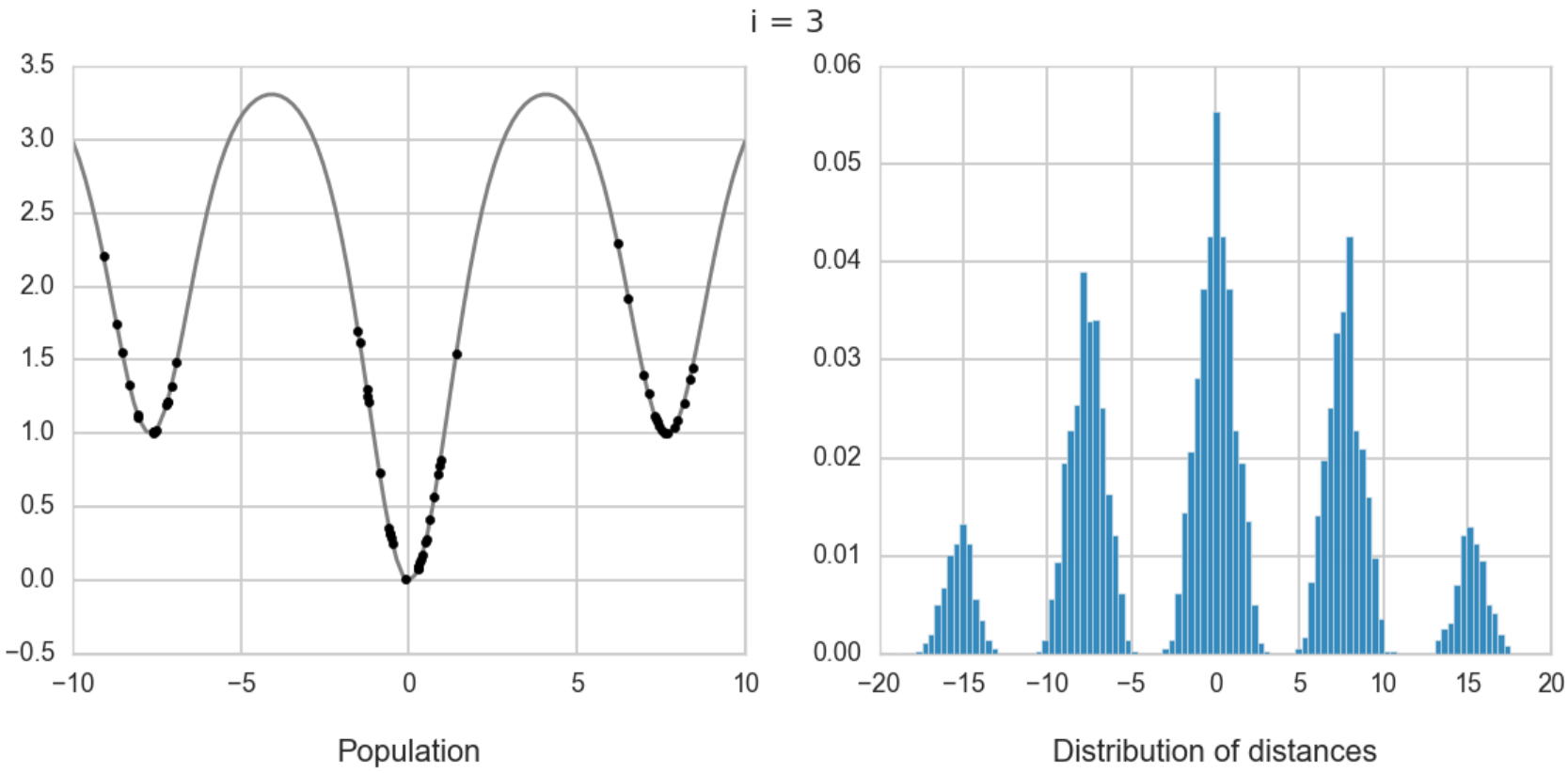
Contour matching



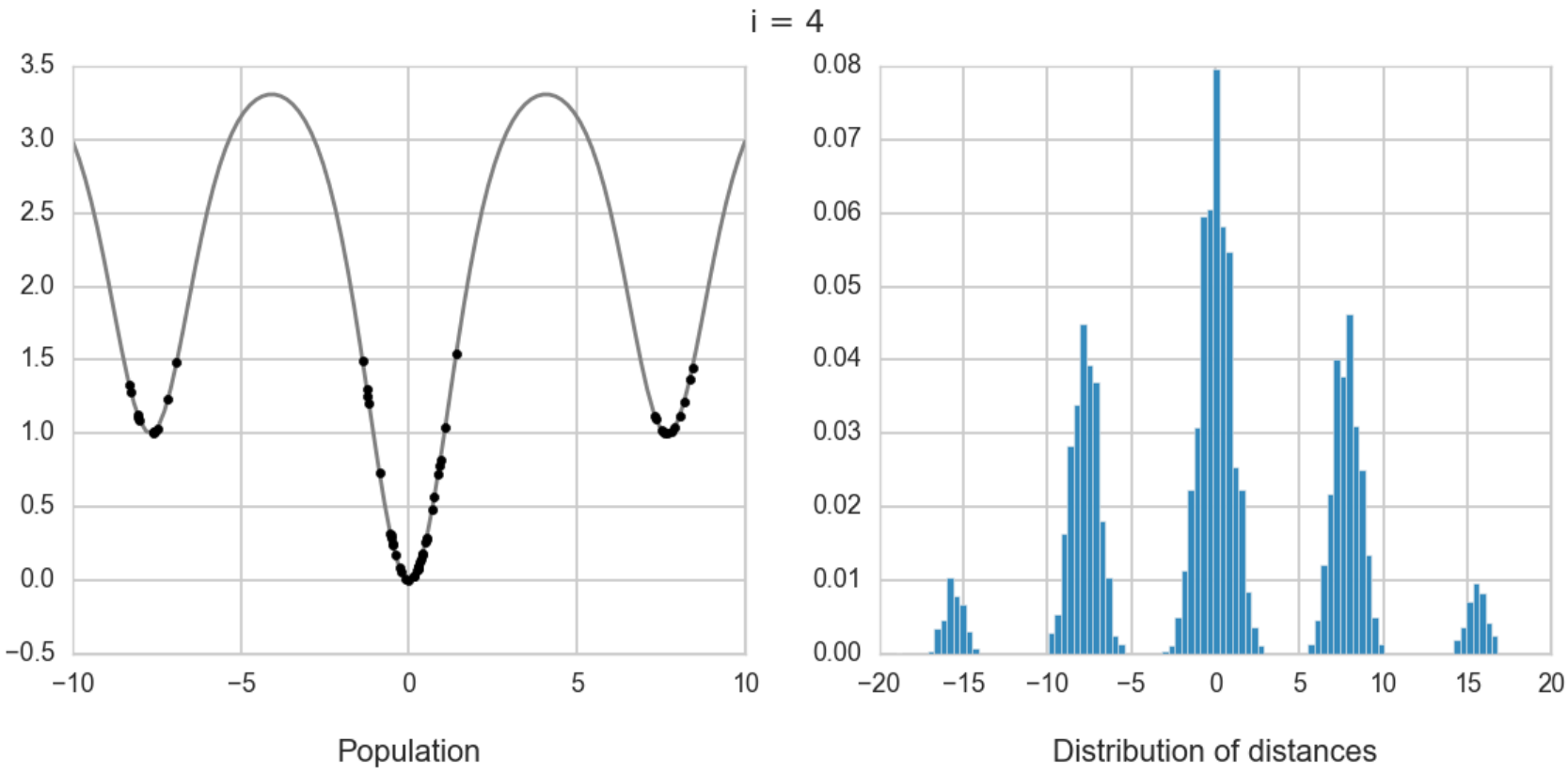
Contour matching



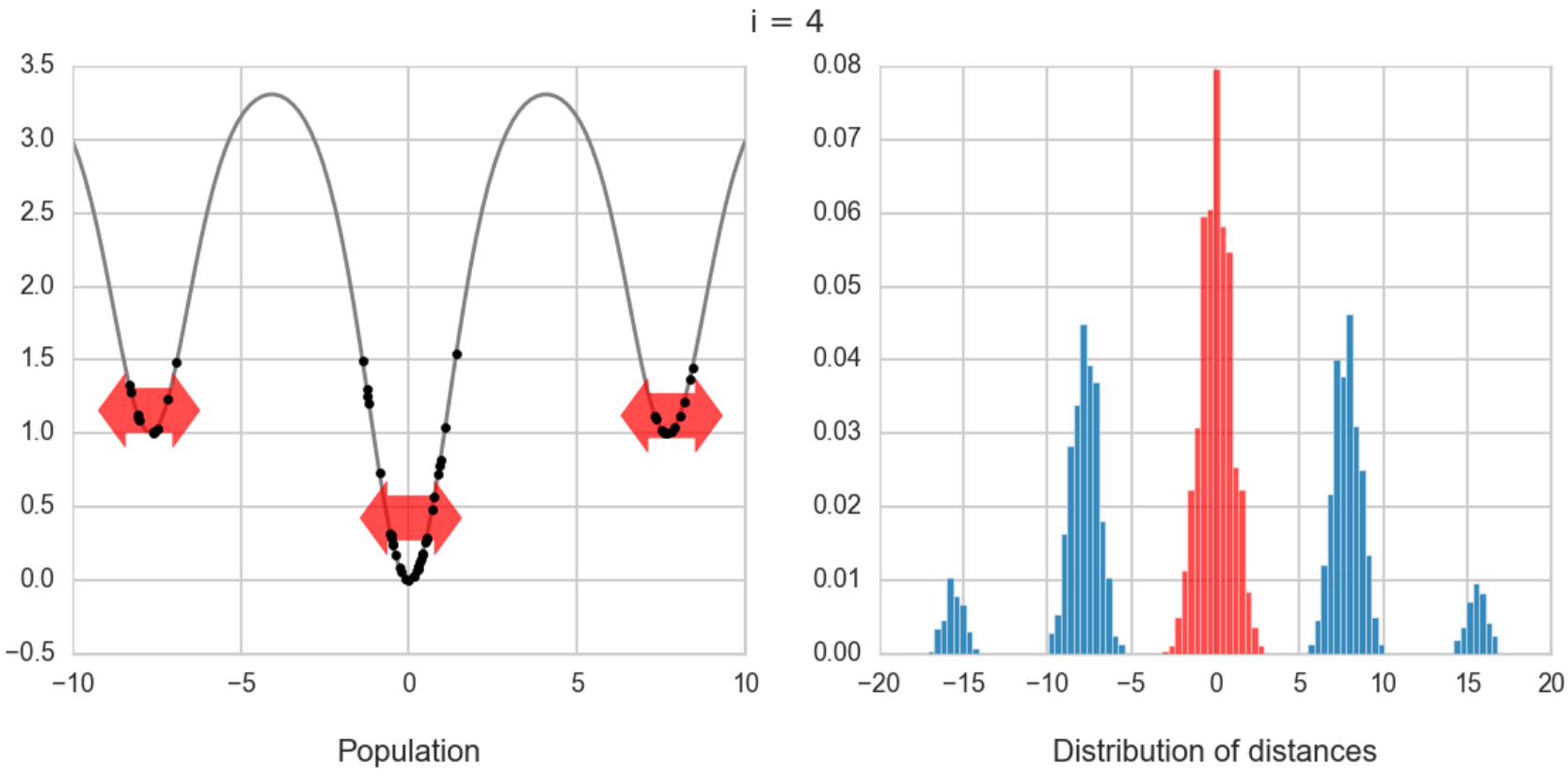
Contour matching



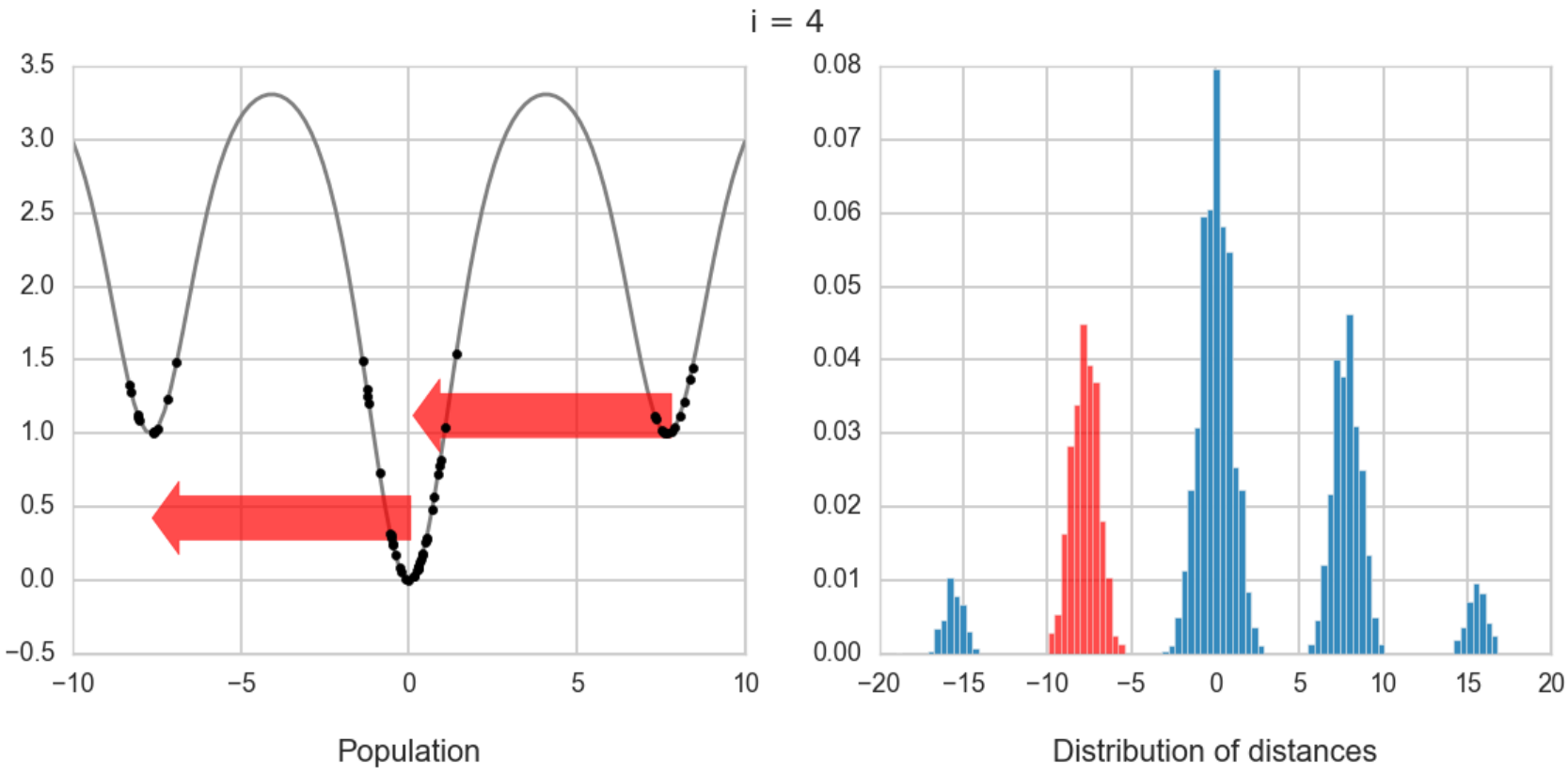
Contour matching



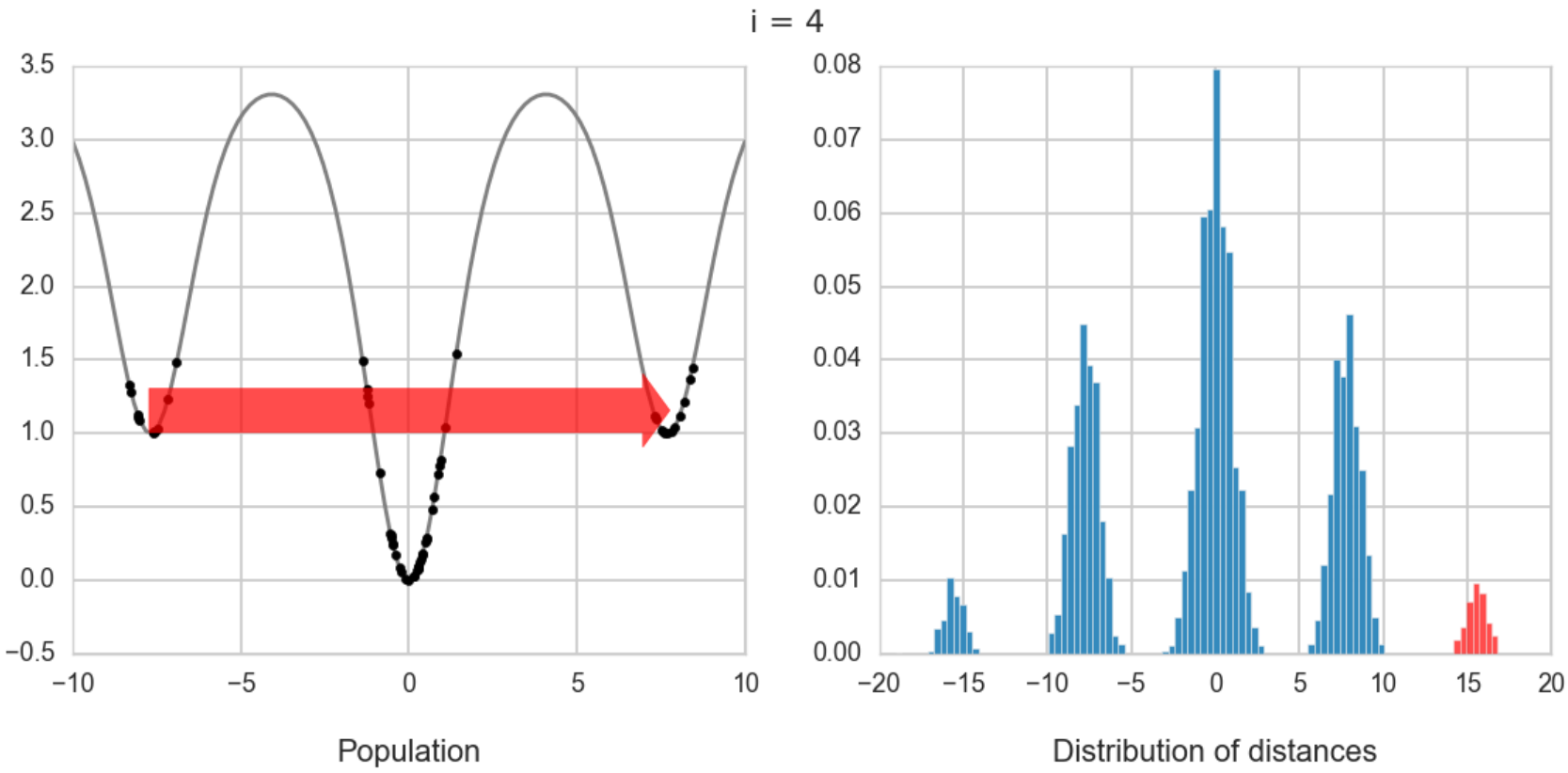
Contour matching



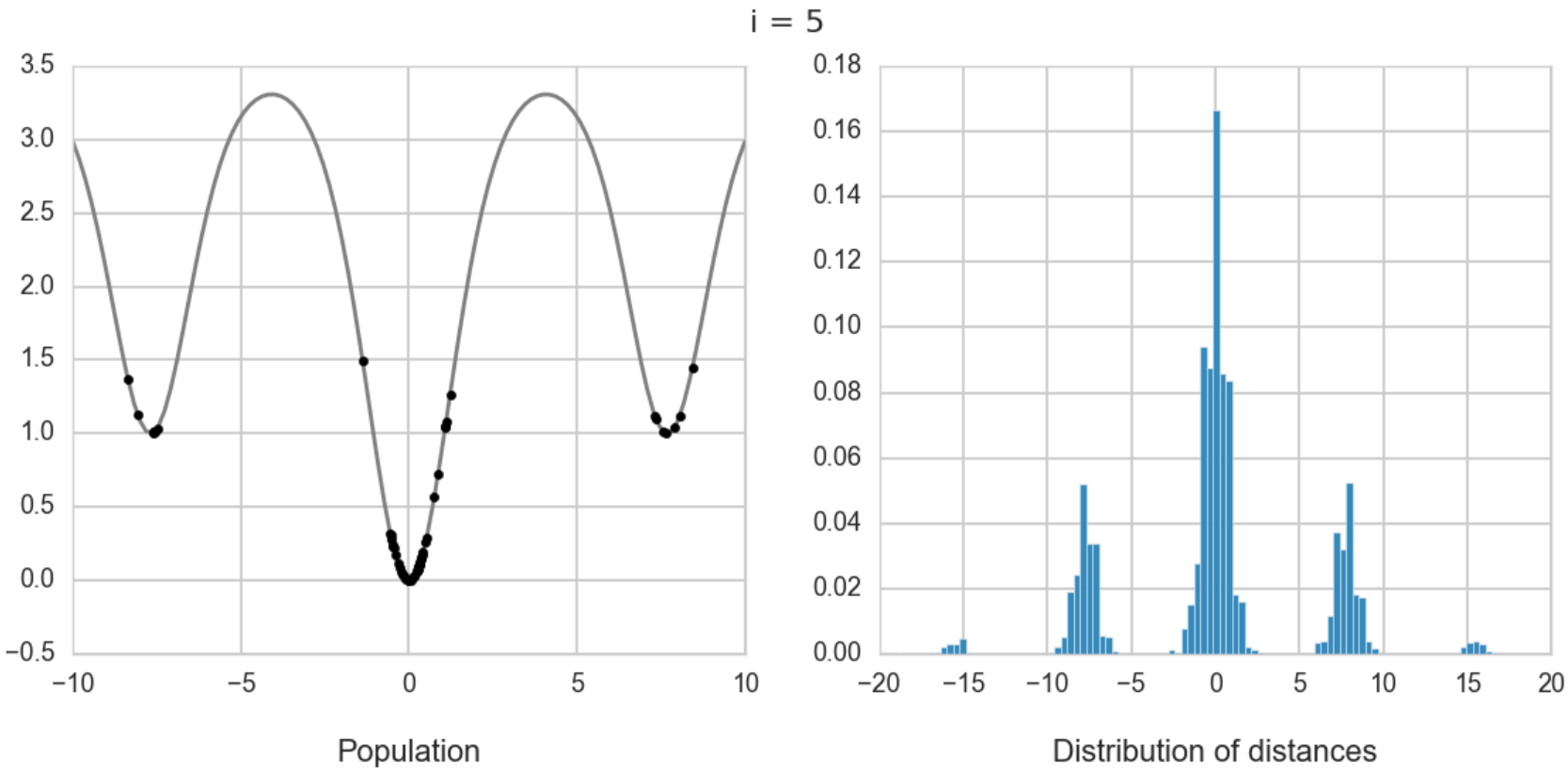
Contour matching



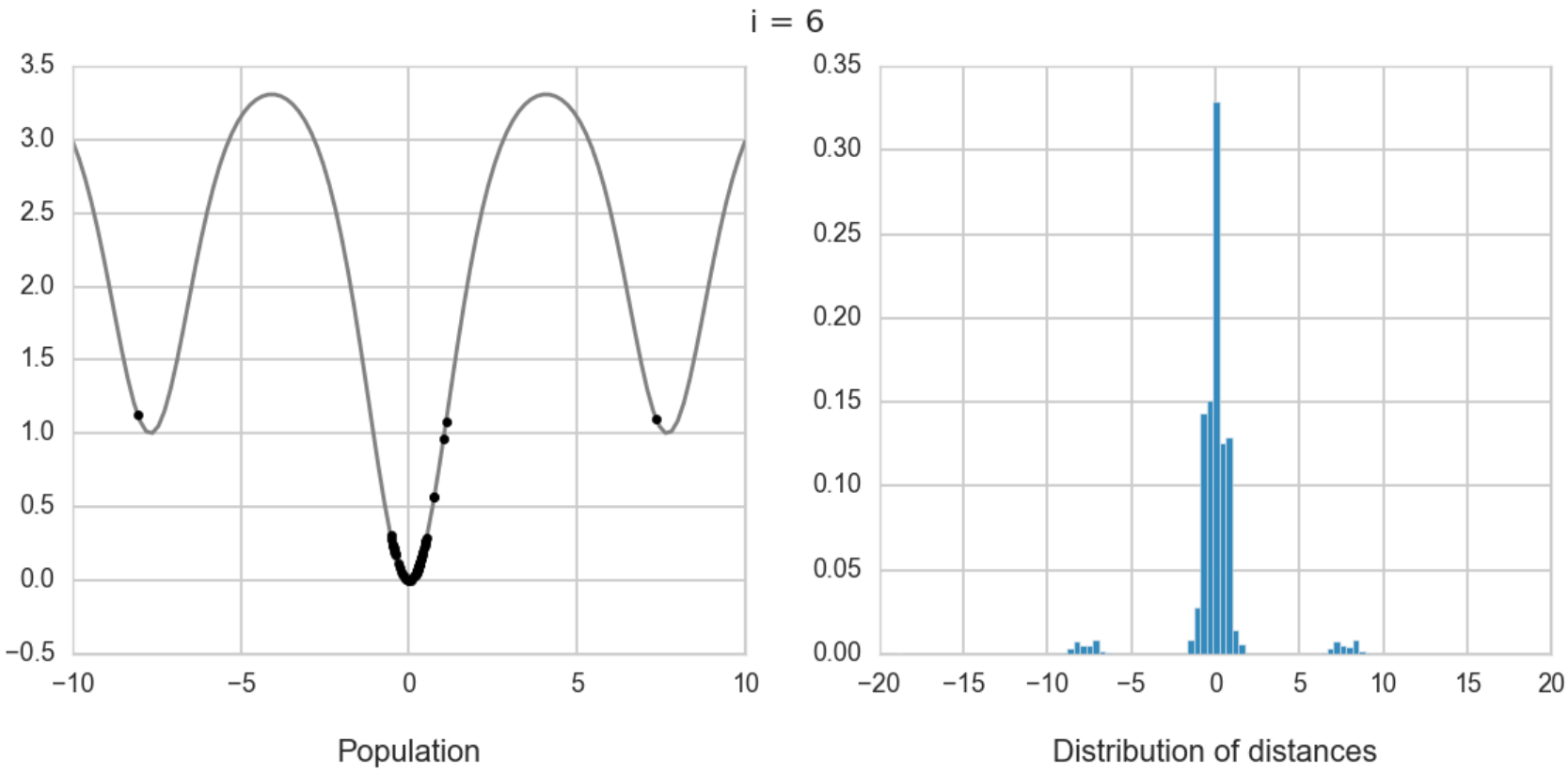
Contour matching



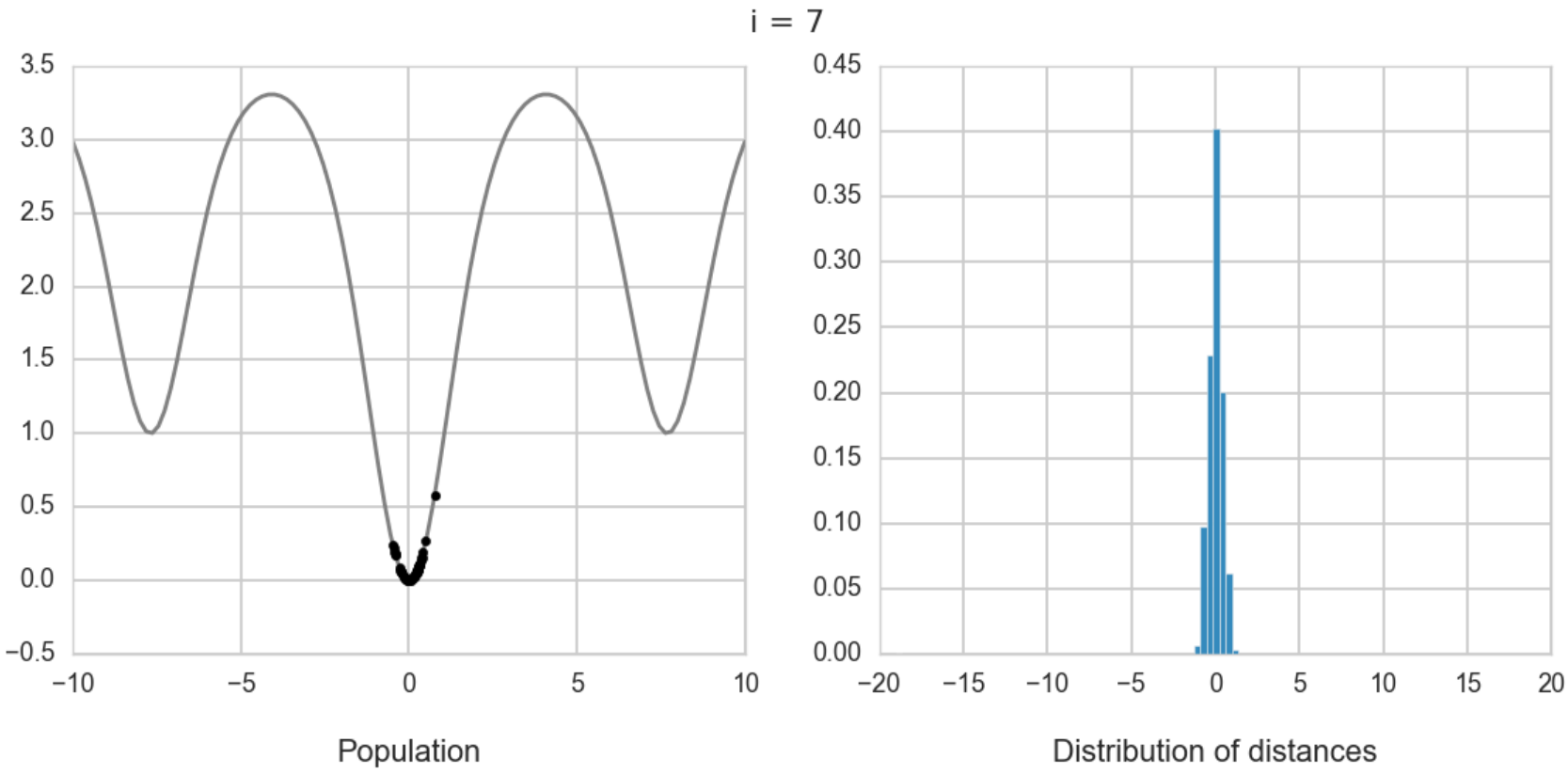
Contour matching



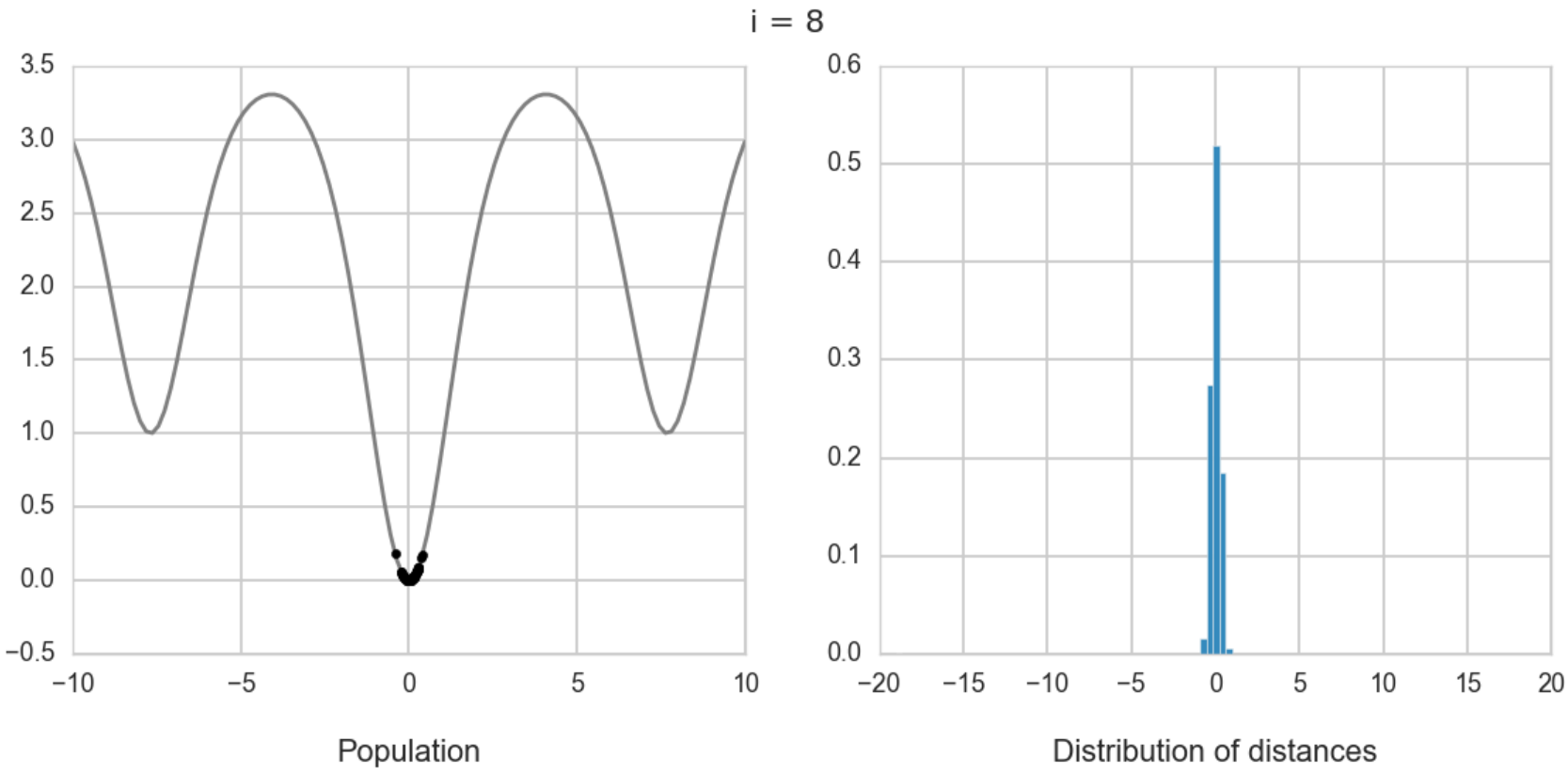
Contour matching



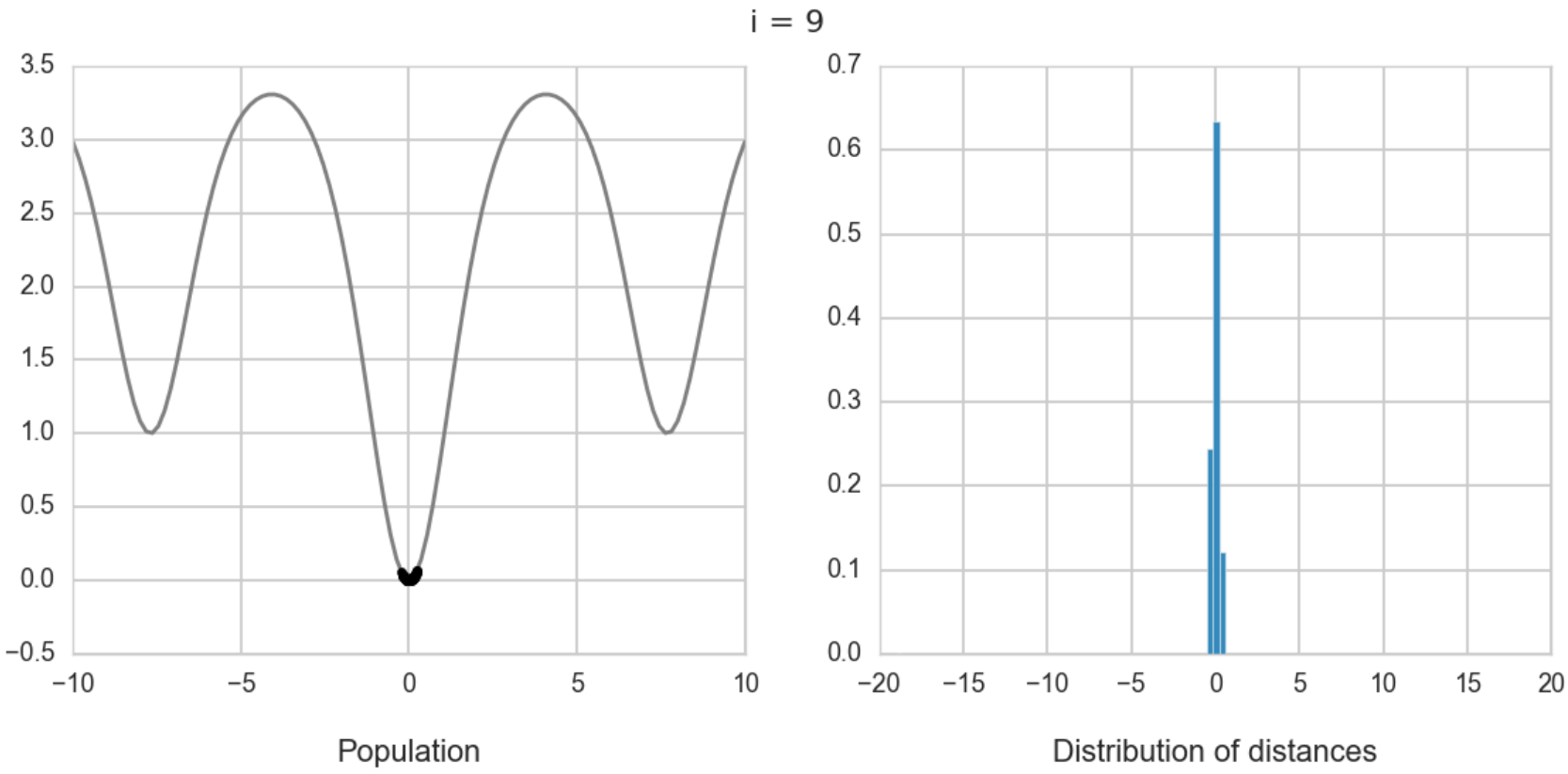
Contour matching



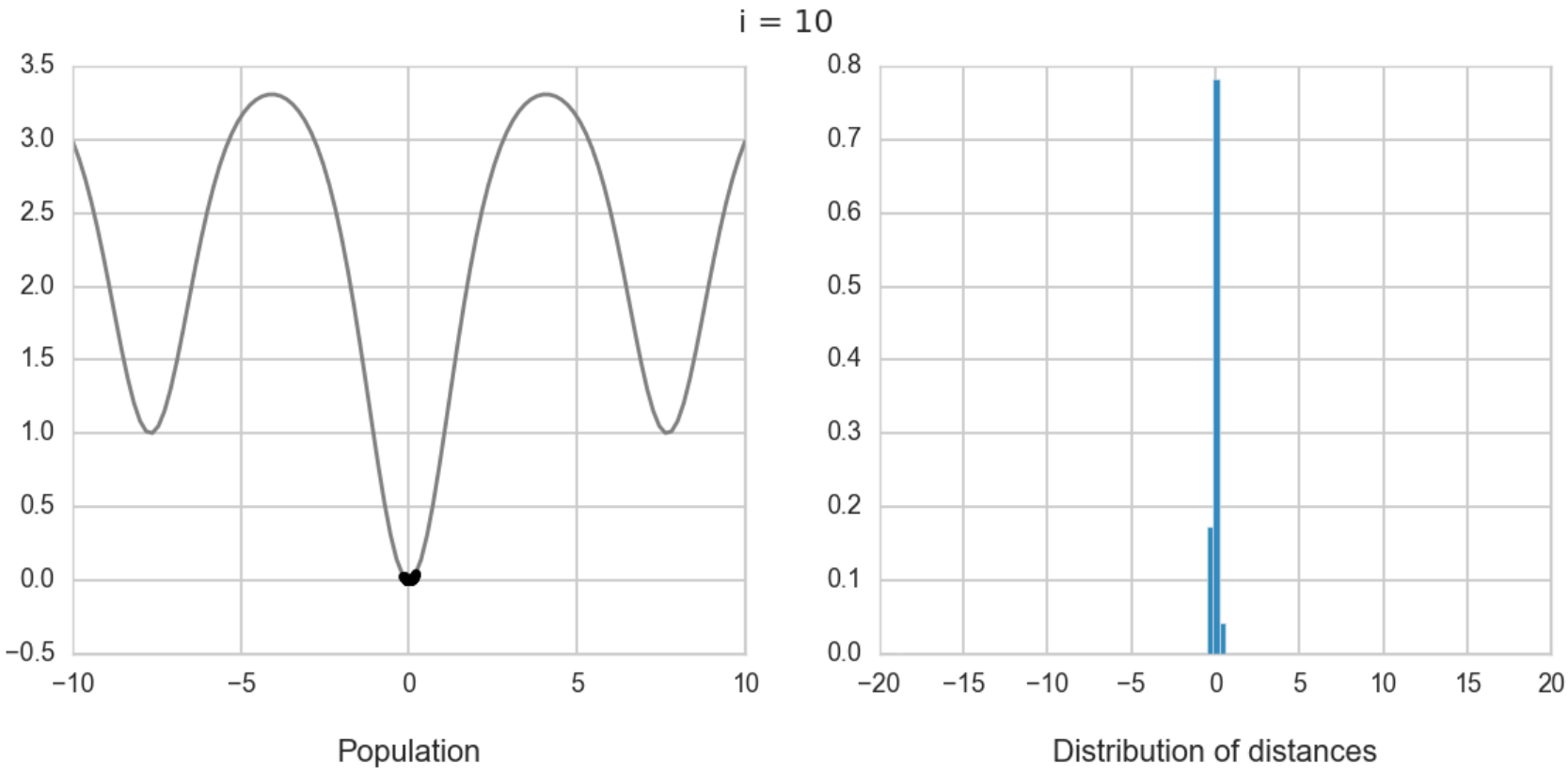
Contour matching



Contour matching



Contour matching



Degenerate combinations

What happens if target t, base b or one of the difference vectors x_1, x_2 coincide ($p = \frac{1}{N}$)?

$x_1 = x_2$	**No mutation.** Zero difference, base vector will be equal to trial, only crossover.
$x_1 = b$	**Arithmetic Recombination**. Move base vector along line between x_1 and b.
$t = b$	**Mutation only**.
$t = x_1$	;-)

Variations

Variations

So far we have only discussed 'classic' DE with:

- a **random** base vector
- **one** difference vector
- (approximately) **binomial** crossover

Classic DE is therefore also called: *DE/rand/1/bin*.

Each of these components can be modified to produce a different variation of the algorithm.

Variations

Recall:

- *donor* = *base* + *scaled difference(s)*
- *trial* = crossover between *donor* and *target*

Component	**Common schemes**
base vector	random, best, target-to-best
difference vectors	1, 2
crossover	binomial, exponential

Variations – Base vector

donor = **base** + *scaled difference(s)*

Possible base vectors (b):

- a **random** vector from the population: $b = x_{random}$
- the **best** performing vector: $b = x_{best}$
- the **target** moved in the direction of the **best**:
  $b = t + \lambda \cdot (x_{best} - t)$

Variations – Difference vectors

donor = *base* + **scaled difference(s)**

Recommended options:

- one difference vector: $F(x_1 - x_2)$
- two difference vectors: $F(x_1 - x_2) + F(x_3 - x_4)$

Reminder: to avoid **degenerate vectors**, base vectors and vectors for differences should be mutually exclusive.

Variations – Mutation schemes

A combination of base vector and difference vector schemes forms a mutation scheme.

Mutation schemes suggested by Price et al. [2006] include:

- *rand/1* (classic)
- *best/1*
- *target-to-best/1*
- *best/2*
- *rand/2*

Variations – Crossover

trial = crossover between *target* and *donor*

In addition to a guaranteed inheritence of one component from donor:

- inherit each component from donor with uniform probability (approximately **binomial** crossover)
- 1- or 2-point crossover (approximately **exponential** crossover)

Variations – Crossover

Exponential crossover

start from random index (inherit this component from d)
loop over indices with wrap around
while $rand(0, 1) < Cr$ **do**
    z inherit from d
z inherit all remaining components from t
return z

Variations – Summary

Component	**Common schemes**
base vector	random, best, target-to-best
difference vectors	1, 2
crossover	binomial, exponential

Example variation: *DE/best/2/exp*

Control Parameters

Control parameters

DE can be controlled with the following parameters:

- Population size (N)
- Scaling factor (F)
- Crossover rate (Cr)

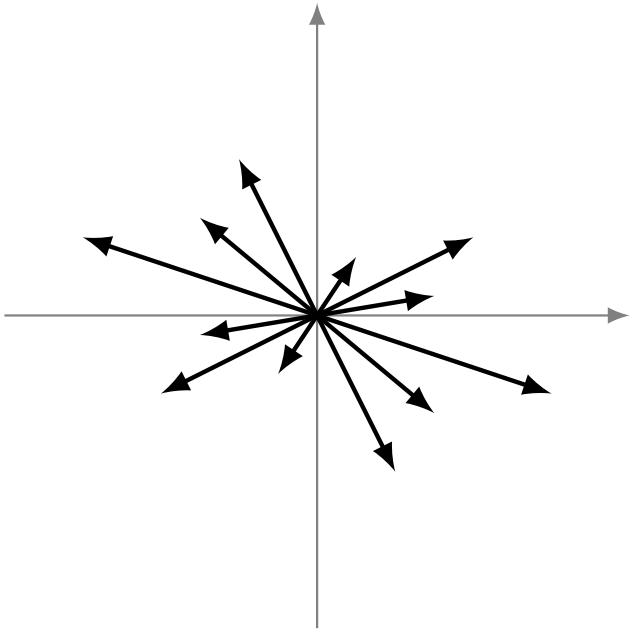
Control Parameters – Population size N

The population size (N) has two direct impacts on the characteristics of the algorithm:

- number of solutions generated at each iteration
- size of the set of difference vectors used for mutation

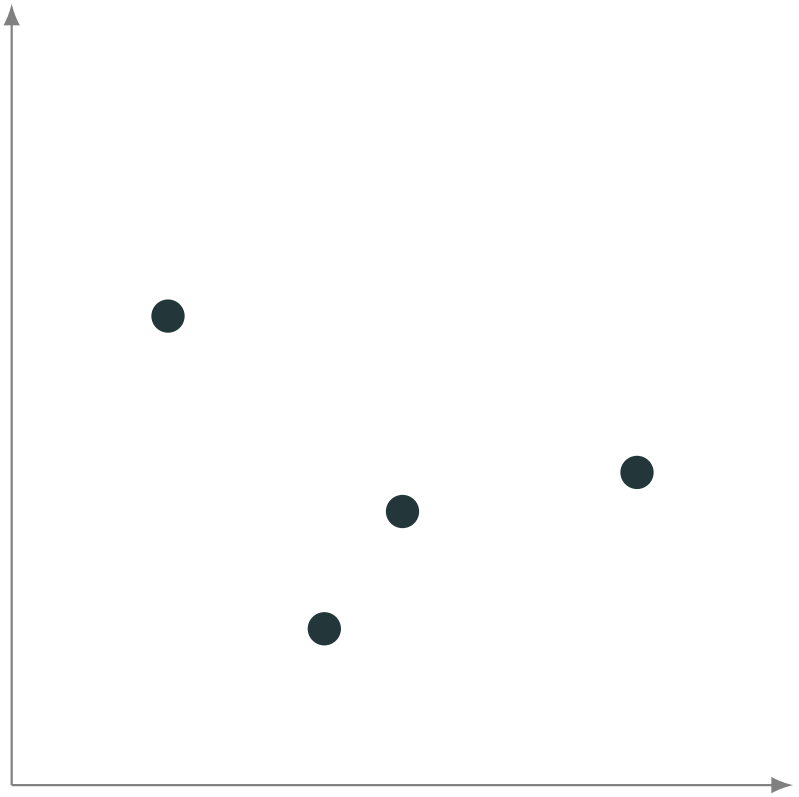
Control Parameters – Population size N

- The non-zero difference vectors used for perturbing the base vector come from a set of size $N \cdot (N-1)$.
- The population size needs to be large enough to have enough variation in the choice of the difference vector(s).



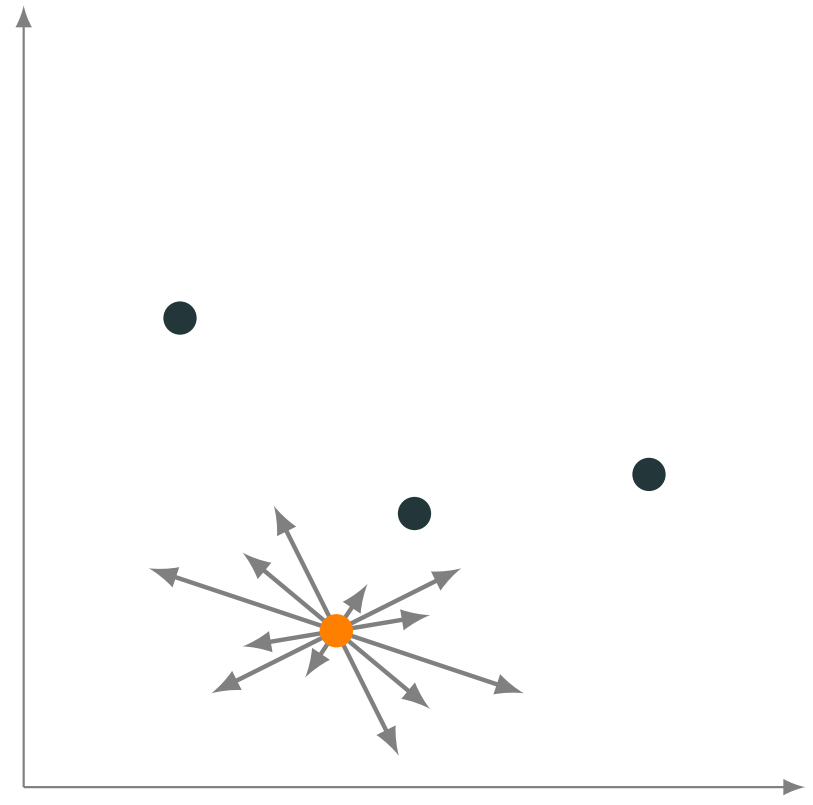
12 difference vectors distributed around the origin. $N = 4$.

Control Parameters – Population size N



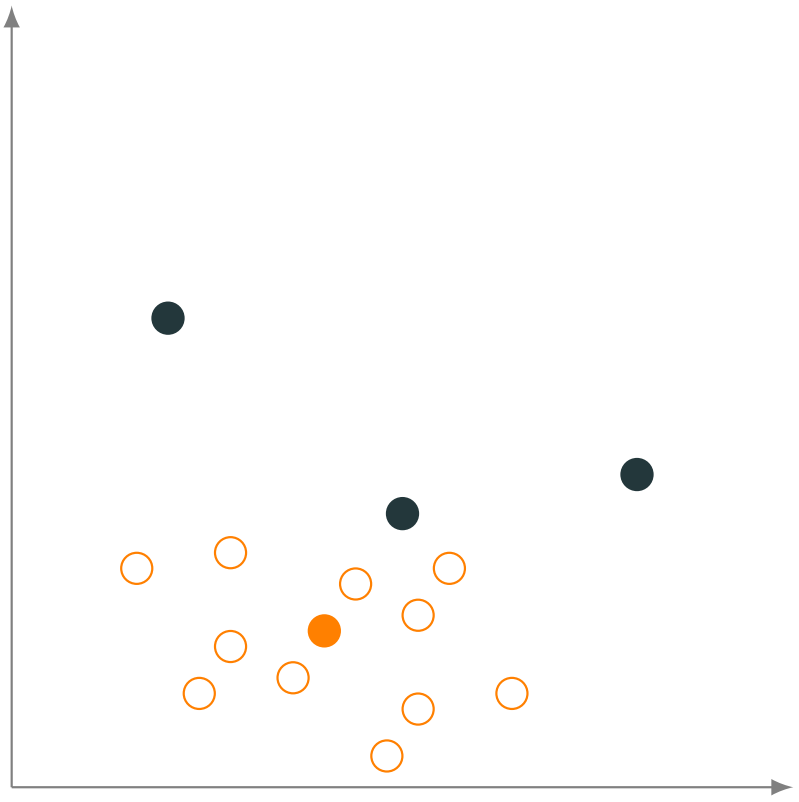
Population vectors.
$N = 4, F = 0.4$.

Control Parameters – Population size N



Possible perturbations of a base vector.
$N = 4, F = 0.4$.

Control Parameters – Population size N



Possible donor vectors resulting from perturbation.
$N = 4, F = 0.4$.

Notice that the search space is not well covered.

Control Parameters – Population size N

Recommended values for N include 5- and 10-times the dimensionality of the search space.

- Example: if the search space is 2-dimensional, a population size of $5D$ would result in a distribution of 90 distinct difference vectors.
- This set would likely provide enough variation in the donor vectors to adequately explore the search space.

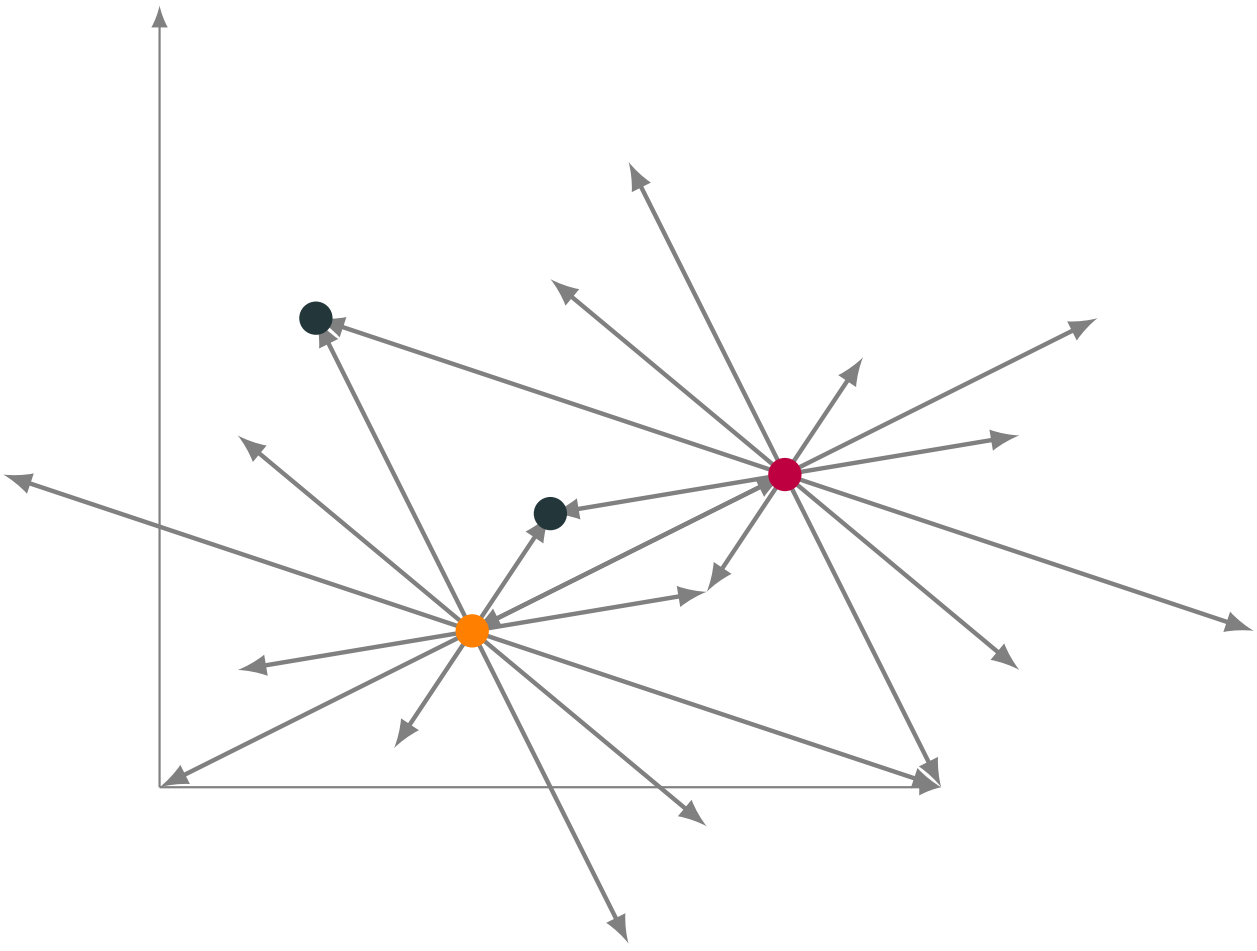
Control Parameters – Scaling factor F

Recall: $d = b + F(x_1 - x_2) + F(x_3 - x_4)$

The amount that the difference vectors are scaled by is controlled by the Scaling Factor (F).

- F should be in the interval $(0, 1)$.
- $F = 1$ can lead to decreased variation in mutation.

Control Parameters – Scaling factor F



Consequences of setting $F = 1$.
Perturbations of different base vectors may result in the same donor vector.

Control Parameters – Scaling Factor F

A guideline from Zaharie [2002] to avoid convergence on a flat objective function landscape is:

$$F \geq \sqrt{\frac{0.5 \cdot (1 - Cr)}{N}}.$$

Control Parameters – Crossover rate Cr

The crossover rate Cr controls the likelihood of the trial vector (z) inheriting components from the donor vector (d) instead of the target vector (t).

- Cr should be in the interval $(0, 1]$.
- Generally, successful Cr values are close to 0 or close to 1.
- If a problem is decomposable (each vector component can be optimized individually) Cr close to 0 works well
- If a problem is not decomposable, Cr close to 1 works well

Control Parameters – Summary

Parameter	**Meaning**	**Recommended values**
N	Population size	5- or 10- times dimensionality
F	Scaling factor	< 1 (see previous slides for minimum)
Cr	Crossover rate	close to 0 or close to 1

Constrained optimization

Constrained Problems

Find an assignment to all D dimensions of x

to minimize the objective function $f(x)$

Subject to:

- inequality constraints: $\gamma_m(x) \leq 0, m = 1, 2, ..., M$
- equality constraints: $\phi_n(x) = 0, n = 1, 2, ..., N$ [2]
- boundary constraints: $lower_j \leq x_j \leq upper_j, j = 1, 2, ..., D$

[2] Difficult to handle for DE. Common approach: transform them into pairs of inequality constraints: $\phi_n(\boldsymbol{x}) = 0 \Rightarrow (0 - \epsilon) \leq \phi_n(\boldsymbol{x}) \leq (0 + \epsilon)$

Boundary Constraints

Penalty methods

- **Brick wall penalty**: if any parameter violates constraint, make value of objective function so high that it won't be selected
- **Adaptive penalty**: add a penalty for each violated constraint, maybe also dependent on the magnitude of violation
- $\rightarrow$ can be put in the objective function, main DE-algorithm does not have to be changed

Resetting schemes

- **Random reinitialization**: reinitialize the violated parameter with a random number from the allowed range
- **Bounce back**: reinitialize the violated parameter with a value between the violated bound and the base vector

Inequality Constraints: Penalty functions

Change the objective function $f(\boldsymbol{x})$ to:

$$f'(x) = f(x) + \sum_{m=1}^{M} w_m * p_m(x)$$

where w_m are weights and $p_m(x)$ are penalty functions

One possibility to steer the solution away from infeasible regions:

$$p_m(x) = \begin{cases} \gamma_m^2(x) & \text{if } \gamma_m(x) > 0 \\ 0 & \text{otherwise} \end{cases}$$

This adds values to the objective function iff the constraint is violated. The penalty is proportional to the violation, i.e. stronger violations lead to higher penalties.

Inequality Constraints: Penalty functions

Pros and cons:

+ easy, no modification of the main DE algorithm needed
+ for few constraints it often works very well
- weights have to be chosen carefully
  - `underpenalization` might result in slow convergence towards feasible solutions, or feasible solutions are not found at all, if low objective value outweights the penalty
  - `overpenalization` leads to fast convergence upon feasible solutions, but often prematurely
  - $\rightarrow$ for a high number of constraints, finding the right weights can be an optimization problem on its own

Direct constraint handling (Lampinen [2002])

Only modifies the *selection* part of the algorithm

Select the trial vector:

- if both target and trial vector are feasible solutions and the trial vector has lower or equal objective value
- if the trial vector is feasible and the target vector is not
- if both are infeasible, but trial vector has lower or equal constraint violations for all constraints (pareto domination)

otherwise, keep the target vector

Direct constraint handling

+ only compares objective values of two vectors when both are feasible
+ for infeasible population members: selective pressure only towards feasibility, independent of the value of the objective function
+ no pressure to over-satisfy constraints
+ no additional parameters

Complex optimization environments

Large-Scale Optimization

DE-algorithms perform quite well for dimensions of 30–100 but can't deal with dimensions of over 500:

1. Complexity increases with size of problem
2. Solution space increases exponentially with problem size

$\rightarrow$ Optimal solution might not be found or search is not feasible

Large-Scale Optimization

Solution: Coevolution with a divide-and-conquer strategy:

1. Problem decomposition
2. Optimize subcomponents
3. Cooperative combinations

Adaptable for non-separable high-dimensional functions.

Dynamic and Uncertain Environments

Uncertainties in optimization problems can arise from:

1. Noisy fitness functions
2. Design variables or parameters might change after optimization
3. Approximated fitness functions
4. The optimum might change over time

A robust optimal solution is needed

Example: Price fluctuations

Moving peak benchmark

Fitness function changing slightly over time Branke [2012]

Some parameters are:

1. Number, position and height of peaks
2. Severity of movement and change of height and width

Moving peak benchmark

Function : $f(\mathbf{x}) = \frac{h}{1+w\sqrt{\sum_{i=1}^{N}(x_i - p_i)^2}}$

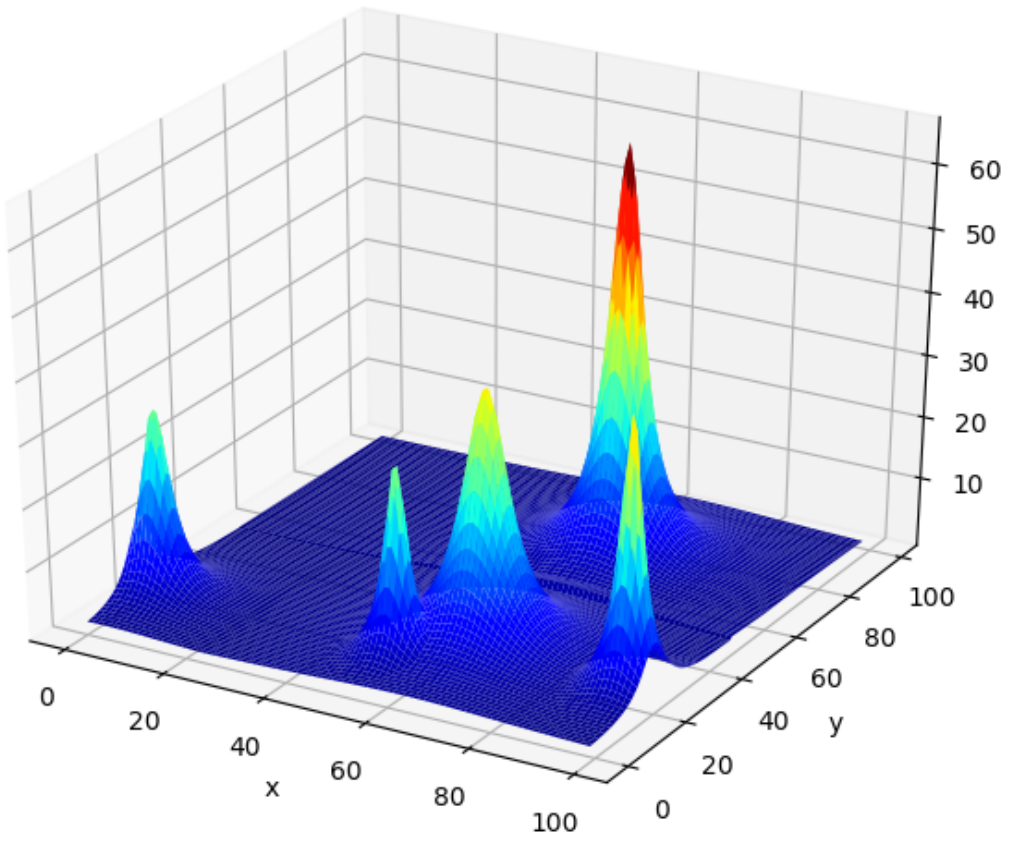


Figure 1: moving peaks, 2 dimensions, 5 peaks (Fortin et al. [2012])

Moving peak benchmark

Function : $f(\mathbf{x}) = \frac{h}{1+w\sqrt{\sum_{i=1}^{N}(x_i - p_i)^2}}$

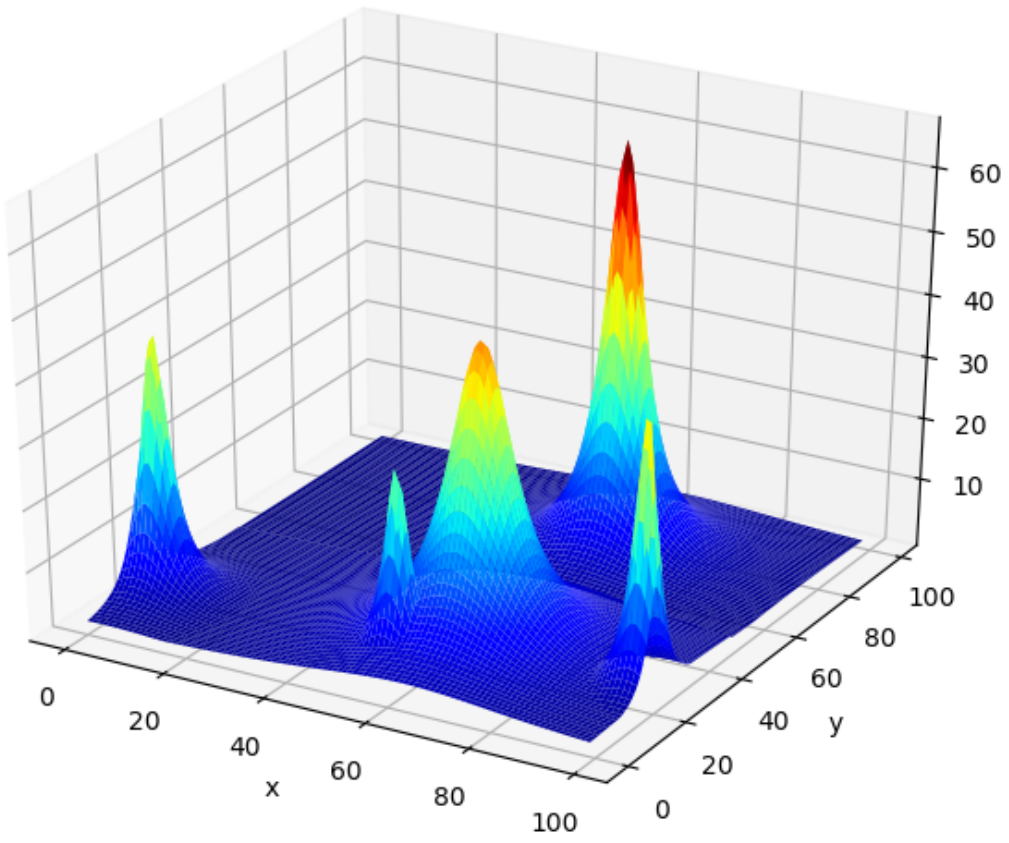


Figure 2: moving peaks, 2 dimensions, 5 peaks (Fortin et al. [2012])

Dynamic and Uncertain Environments

DynDE (Mendes and Mohais [2005])

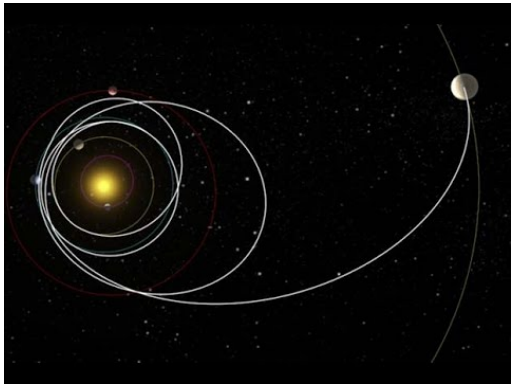
1. Several populations in parallel
2. uniform dither for F [0,1] and Cr [0,1]
3. Maintain the diversity of the populations by:
   - 3.1 Reinitializing population if best individuals of two populations get too close; absolute best is kept, other one is reinitialized
   - 3.2 Randomize populations with Brownian individuals: replace weakest members of population by best individuals with a small random number added

Gives good results for the moving peak benchmark using a DE/best/2/bin scheme

Applications

JUpiter ICy moons Explorer (JUICE)

European
Space Agency will explore Jupiter's moons in 2022 Finding the optimal trajectories for the search is constrained optimization problem with the following objectives:



JUICE mission (European Space Agency [2017])

1. minimum fuel consumption
2. satisfy mission design constraints
   - 2.1 maximum thrust level
   - 2.2 minimize distance to planet

DE with co-evolution and constraint handling can find feasible solutions Labroquere et al. [2014]

Determining Earthquake hypo-centers

The dimension of the model space is 3 coordinates of the hypocenter + origin time. (4 parameters)

Model vector $\mathbf{m} = (X0, Y0, Z0, T0)$

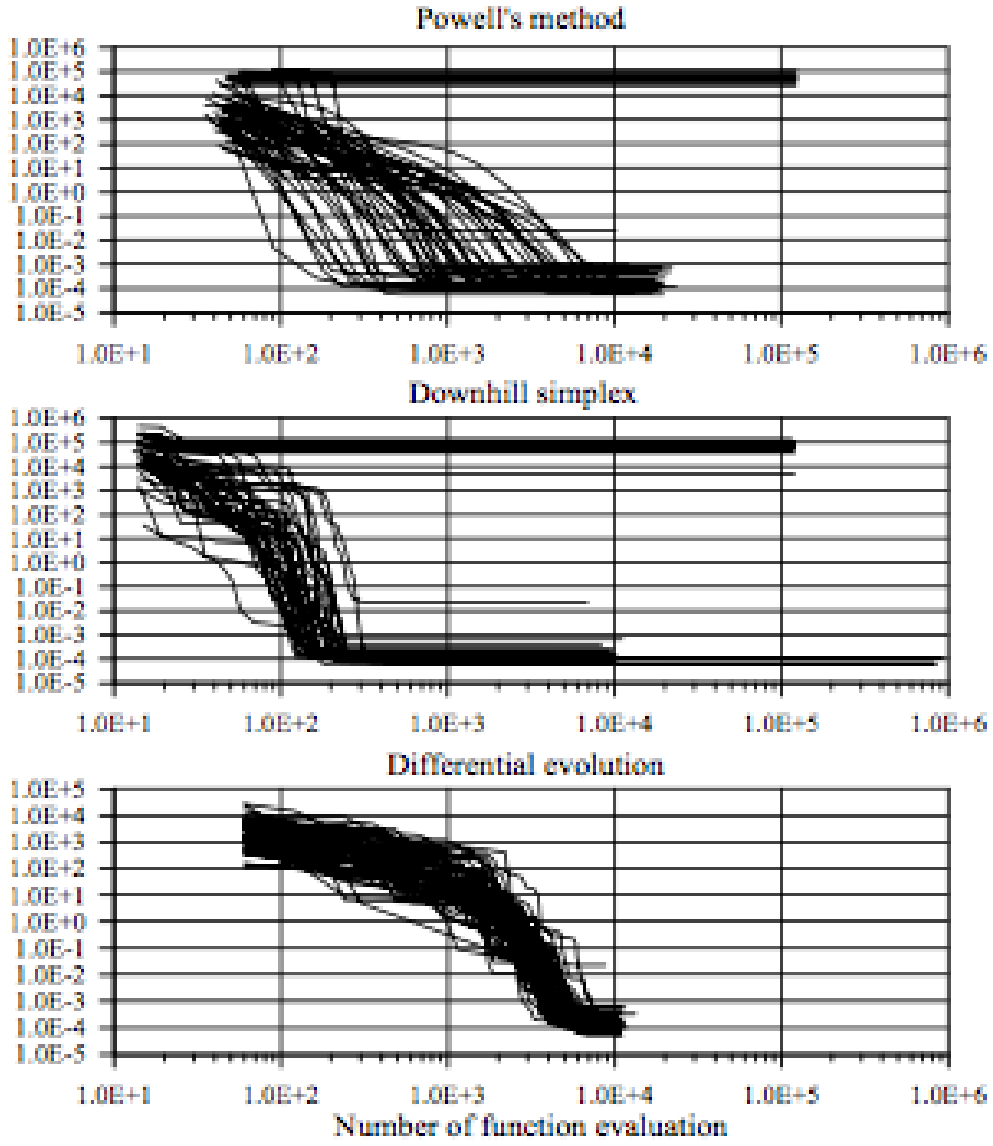
Low dimension but non linear.

Study conducted using 56 synthetic earthquake hypocenter location tasks.

The standard solution is based on minimizing the time residuals that is differences between observed and computed arrivals of seismic waves.

data vector $\mathbf{d}$=$\mathbf{t}$(o) - $\mathbf{t}$(c)

The following minimization represents the search for the optimum solution : $\mathbf{d}^T\mathbf{C}^{-1}\mathbf{d} = min(\mathbf{m})$,where $\mathbf{C}$ is the data covariance matrix.


Powell's method
1.0E+6
1.0E+5
1.0E+4
1.0E+3
1.0E+2
1.0E+1
1.0E+0
1.0E-1
1.0E-2
1.0E-3
1.0E-4
1.0E-5
1.0E+1
1.0E+2
1.0E+3
1.0E+4
1.0E+5
1.0E+6
Downhill simplex
1.0E+6
1.0E+5
1.0E+4
1.0E+3
1.0E+2
1.0E+1
1.0E+0
1.0E-1
1.0E-2
1.0E-3
1.0E-4
1.0E-5
1.0E+1
1.0E+2
1.0E+3
1.0E+4
1.0E+5
1.0E+6
Differential evolution
1.0E+5
1.0E+4
1.0E+3
1.0E+2
1.0E+1
1.0E+0
1.0E-1
1.0E-2
1.0E-3
1.0E-4
1.0E-5
1.0E+1
1.0E+2
1.0E+3
1.0E+4
1.0E+5
1.0E+6
Number of function evaluation

Determining Earthquake hypo-centers

The DE turned out to be best among other optimizers for solution. DE was 100 % reliable.

Best accuracy of all competing algorithms.

Does not insist on evaluations of derivatives.

Works with continuous parameters which need not be bound and their starting positions are not necessary.

Disadvantage : Higher computational effort

Demonstration: fitting a polynomial

A note on notation

We simplified some of the notation for our presentation. In the literature you will find slightly different notation for the vectors and parameters.

Term	Our notation	Price et al. [2006]
Population size	N	Np
Target vector	t	$\mathbf{x}_i$
Base vector	b	$\mathbf{x}_{r0}$
Donor vector	d	$\mathbf{v}$
Trial vector	z	$\mathbf{u}$
Population vectors	$x_1, x_2, \ldots$	$\mathbf{x}_{r1}, \mathbf{x}_{r2}, \ldots$

References I

Jürgen Branke. *Evolutionary optimization in dynamic environments*, volume 3. Springer Science & Business Media, 2012.

Swagatam Das and Ponnuthurai Nagaratnam Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE transactions on evolutionary computation*, 15(1):4–31, 2011.

European Space Agency. 5 gravity-assist flybys will get ESA's JUICE to Jupiter. `https://www.space.com/36127-5-gravity-assist-flybys-will-get-esas-juice-to-jupiter-video.html`, 2017. Accessed: 2010-12-06.

Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.

References II

Jérémie Labroquere, Aurélie Héritier, Annalisa Riccardi, and Dario Izzo. Evolutionary constrained optimization for a Jupiter capture. In *International Conference on Parallel Problem Solving from Nature*, pages 262–271. Springer, 2014.

Jouni Lampinen. A constraint handling approach for the differential evolution algorithm. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 2, pages 1468–1473. IEEE, 2002.

Rui Mendes and Arvind S Mohais. DynDE: a differential evolution for dynamic optimization problems. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 3, pages 2808–2815. IEEE, 2005.

Pablo R. Mier. A tutorial on differential evolution with python. `https://pablormier.github.io/2017/09/05/a-tutorial-on-differential-evolution-with-python/`. Accessed: 2017-11-30.

References III

Kenneth Price, Rainer M Storn, and Jouni A Lampinen. *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

Daniela Zaharie. Critical values for the control parameters of differential evolution algorithms. In *Proc. of MENDEL 2002, 8th Int. Conf. on Soft Computing*, pages 62–67, 2002.