An Introduction to Differential Evolution

Kelly Fleetwood

Synopsis

- Introduction
- Basic Algorithm
- Example
- Performance
- Applications

The Basics of Differential Evolution

- Stochastic, population-based optimisation algorithm
- Introduced by Storn and Price in 1996
- Developed to optimise real parameter, real valued functions
- General problem formulation is:

  For an objective function $f : X \subseteq \mathbb{R}^D \rightarrow \mathbb{R}$ where the feasible region $X \neq \emptyset$, the minimisation problem is to find

$$x^* \in X \text{ such that } f(x^*) \leq f(x) \, \forall x \in X$$

  where:

$$f(x^*) \neq -\infty$$

Why use Differential Evolution?

- Global optimisation is necessary in fields such as engineering, statistics and finance
- But many practical problems have objective functions that are non-differentiable, non-continuous, non-linear, noisy, flat, multi-dimensional or have many local minima, constraints or stochasticity
- Such problems are difficult if not impossible to solve analytically
- DE can be used to find approximate solutions to such problems

Evolutionary Algorithms

- DE is an Evolutionary Algorithm
- This class also includes Genetic Algorithms, Evolutionary Strategies and Evolutionary Programming

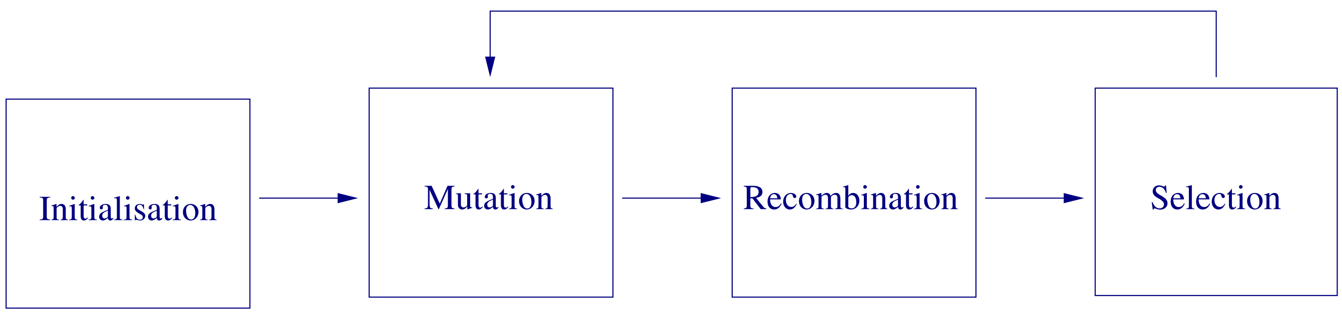


Figure 1: General Evolutionary Algorithm Procedure

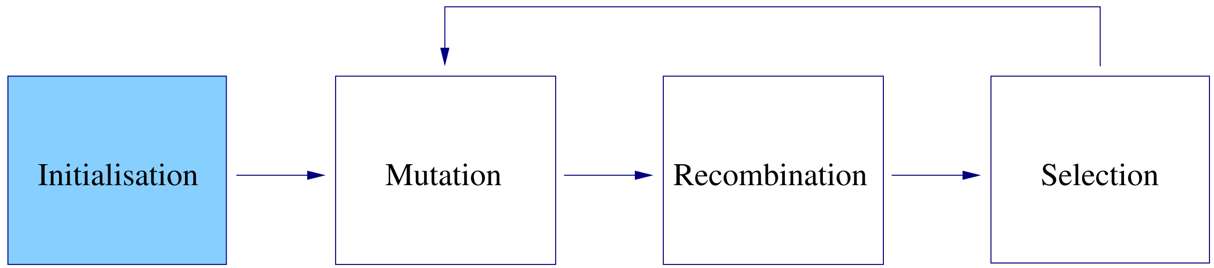
Notation

- Suppose we want to optimise a function with D real parameters
- We must select the size of the population N (it must be at least 4)
- The parameter vectors have the form:

$$x_{i,G} = [x_{1,i,G}, x_{2,i,G}, \dots x_{D,i,G}] \; i = 1, 2, \dots, N.$$

where G is the generation number.

Initialisation

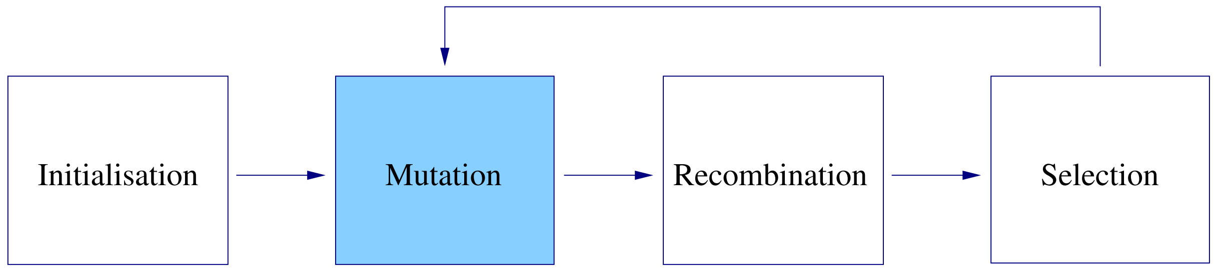


- Define upper and lower bounds for each parameter:

$$x_j^L \leq x_{j,i,1} \leq x_j^U$$

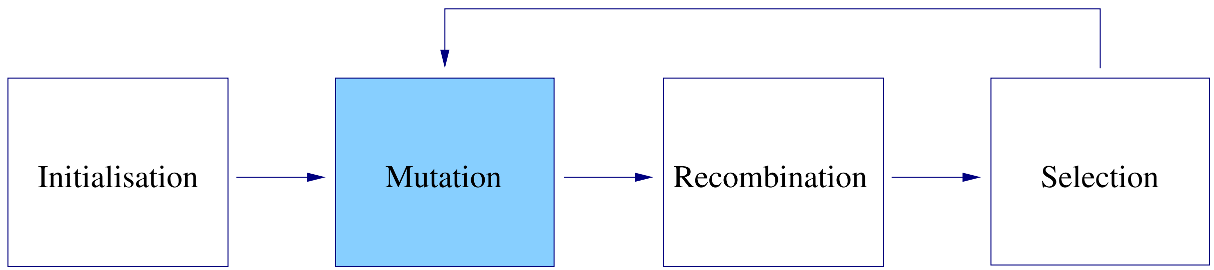
- Randomly select the initial parameter values uniformly on the intervals $[x_j^L, x_j^U]$

Mutation



- Each of the N parameter vectors undergoes mutation, recombination and selection
- Mutation expands the search space
- For a given parameter vector $x_{i,G}$ randomly select three vectors $x_{r1,G}$, $x_{r2,G}$ and $x_{r3,G}$ such that the indices $i, r1, r2$ and $r3$ are distinct

Mutation

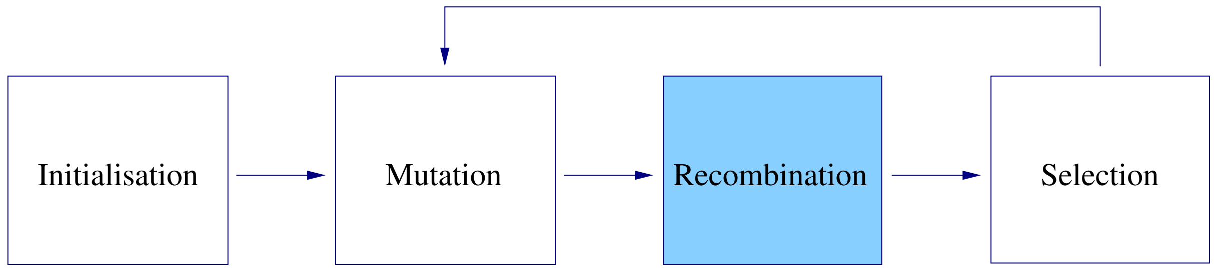


- Add the weighted difference of two of the vectors to the third

$$v_{i,G+1} = x_{r1,G} + F(x_{r2,G} - x_{r3,G})$$

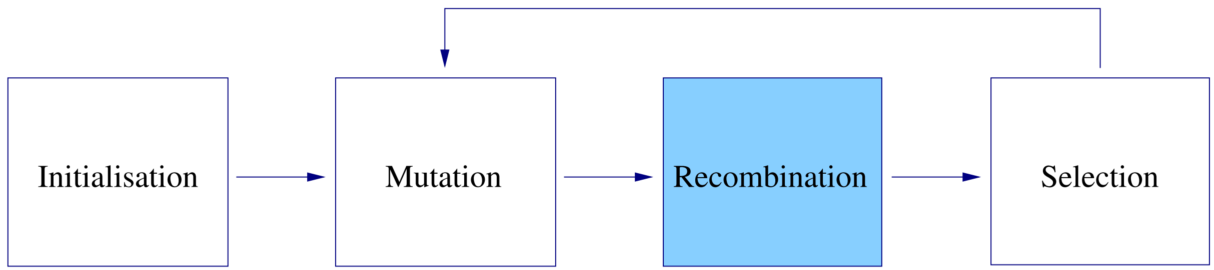
- The mutation factor F is a constant from $[0, 2]$
- $v_{i,G+1}$ is called the donor vector

Recombination



- Recombination incorporates successful solutions from the previous generation
- The trial vector $u_{i,G+1}$ is developed from the elements of the target vector, $x_{i,G}$, and the elements of the donor vector, $v_{i,G+1}$
- Elements of the donor vector enter the trial vector with probability CR

Recombination

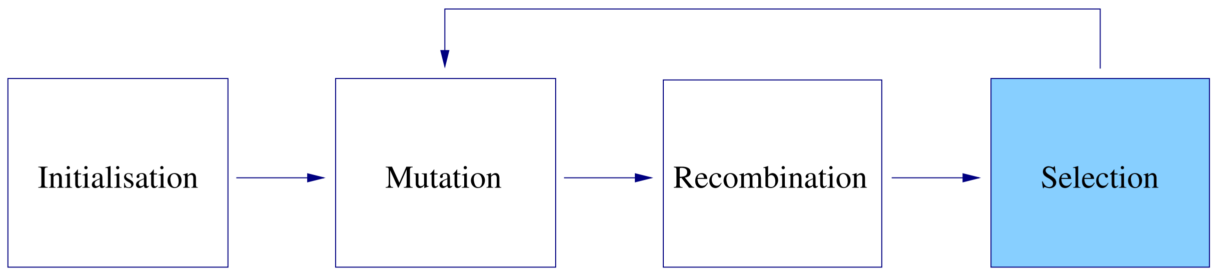


$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & \text{if } \mathrm{rand}_{j,i} \leq CR \text{ or } j = \mathrm{I}_{\mathrm{rand}} \\ x_{j,i,G} & \text{if } \mathrm{rand}_{j,i} > CR \text{ and } j \neq \mathrm{I}_{\mathrm{rand}} \end{cases}$$

$$i = 1, 2, \ldots, N; \; j = 1, 2, \ldots, D$$

- $\mathrm{rand}_{j,i} \sim U[0,1]$, $\mathrm{I}_{\mathrm{rand}}$ is a random integer from $[1, 2, ..., D]$
- $\mathrm{I}_{\mathrm{rand}}$ ensures that $v_{i,G+1} \neq x_{i,G}$

Selection



- The target vector $x_{i,G}$ is compared with the trial vector $v_{i,G+1}$ and the one with the lowest function value is admitted to the next generation

$$x_{i,G+1} = \begin{cases} u_{i,G+1} & \text{if } f(u_{i,G+1}) \leq f(x_{i,G}) \\ x_{i,G} & \text{otherwise} \end{cases} \qquad i = 1, 2, \ldots, N$$

- Mutation, recombination and selection continue until some stopping criterion is reached

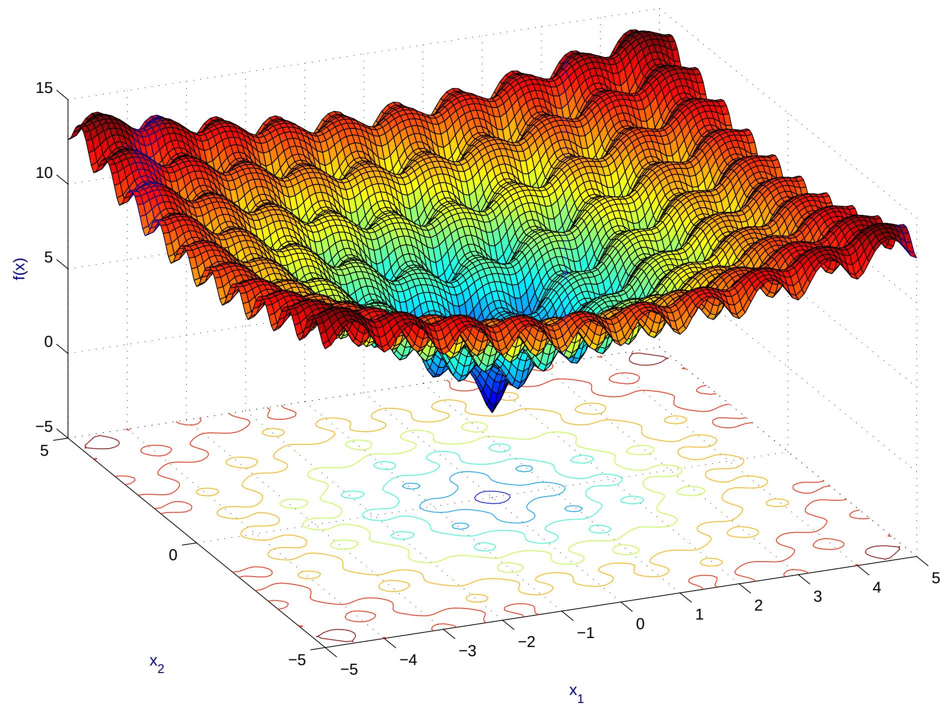
Example: Ackley's function

- DE with $N = 10$, $F = 0.5$ and $CR = 0.1$
- Ackley's function

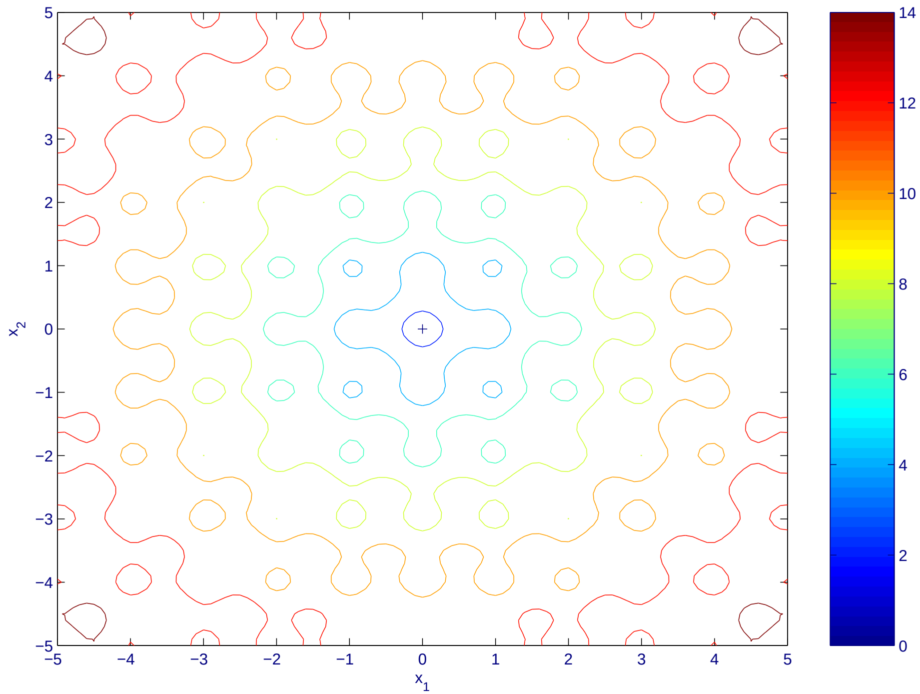
$$f(x_1, x_2) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{n}(x_1^2 + x_2^2)}\right) - \exp\left(\frac{1}{n}\left(\cos(2\pi x_1) + \cos(2\pi x_2)\right)\right)$$

- Find $x^* \in [-5, 5]$ such that $f(x^*) \leq f(x) \, \forall x \in [-5, 5]$
- $f(x^*) = 0;\ x^* = (0, 0)$

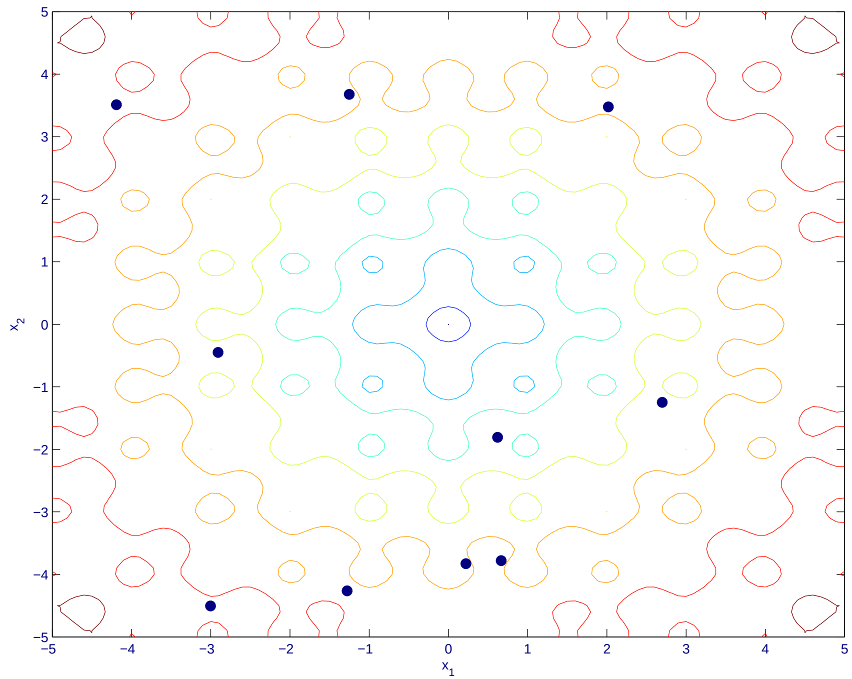
Example: Ackley's function



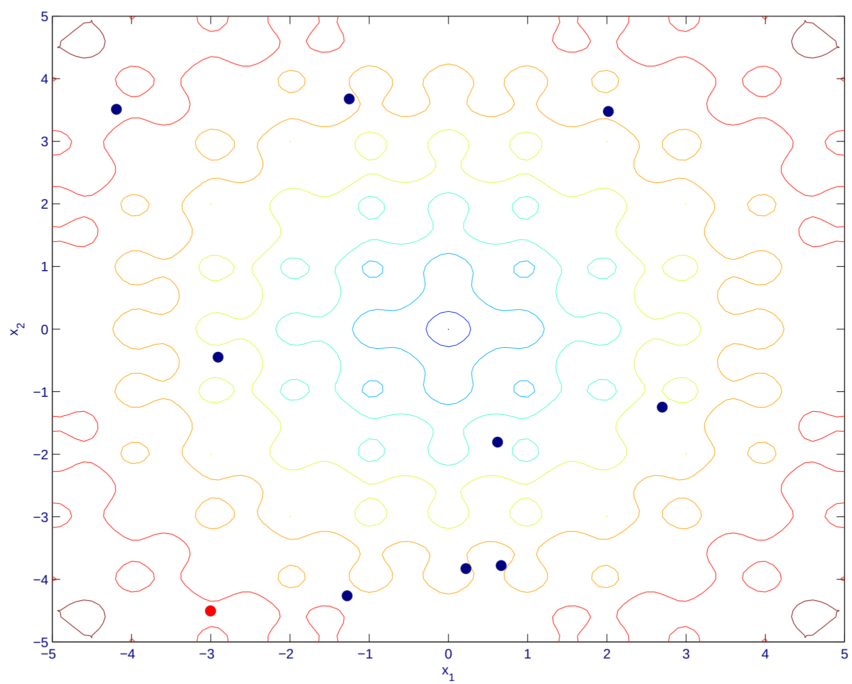
Example: Ackley's function



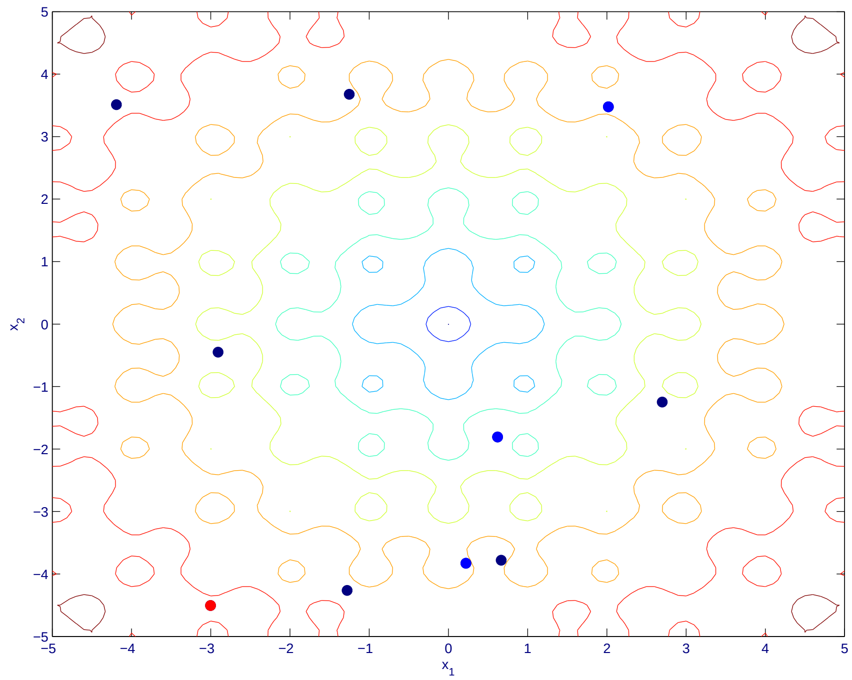
Example: Initialisation



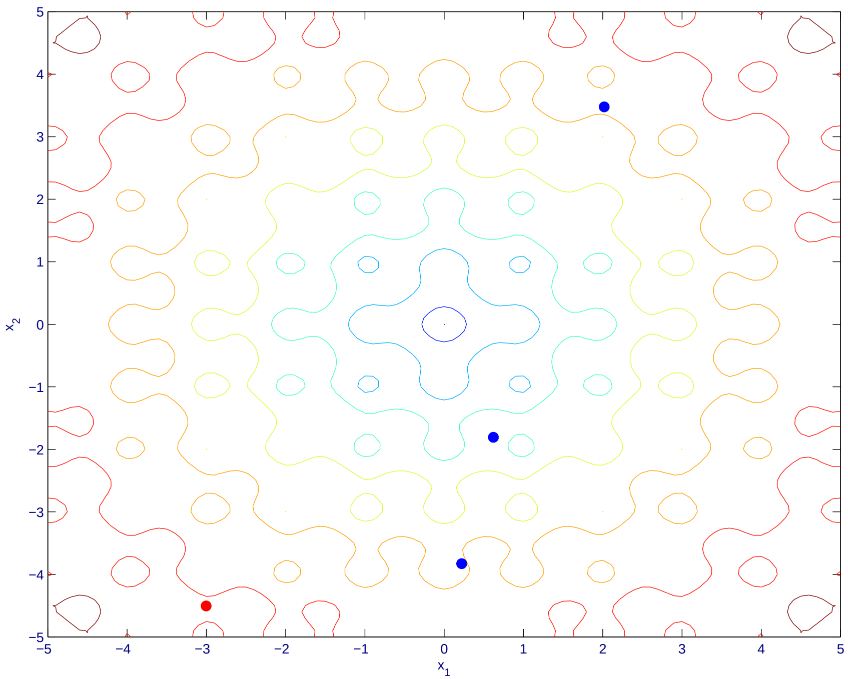
Example: Mutation



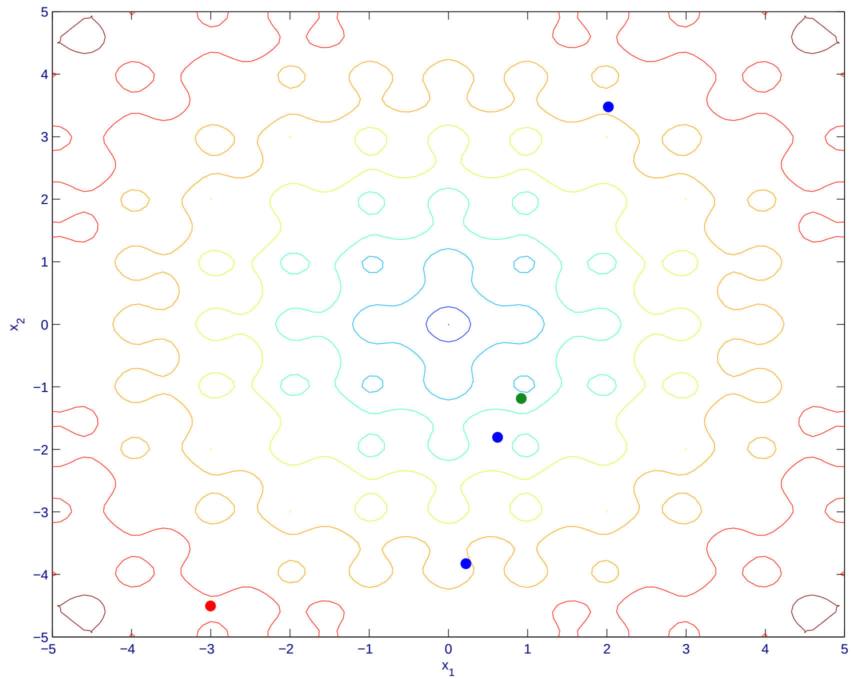
Example: Mutation



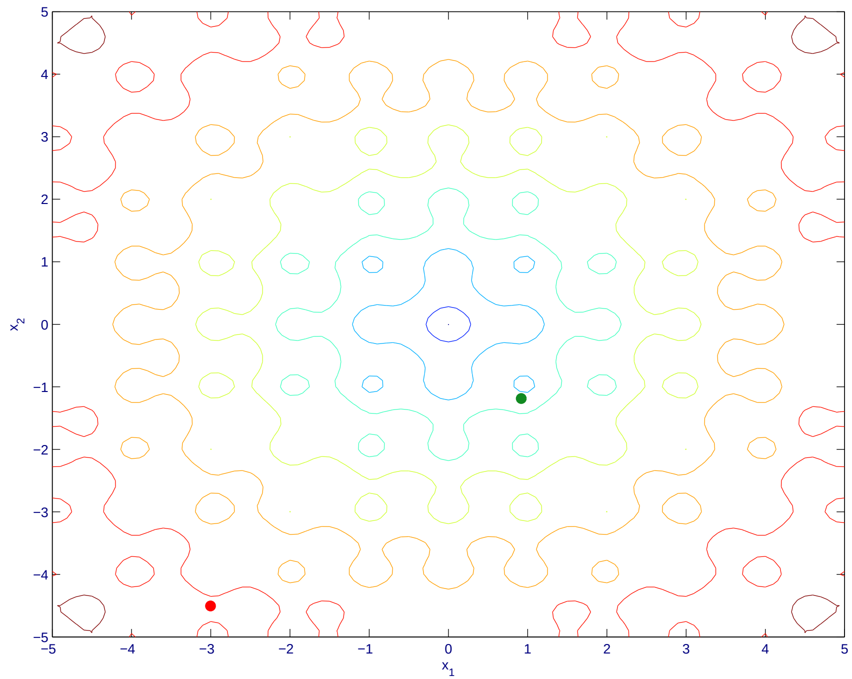
Example: Mutation



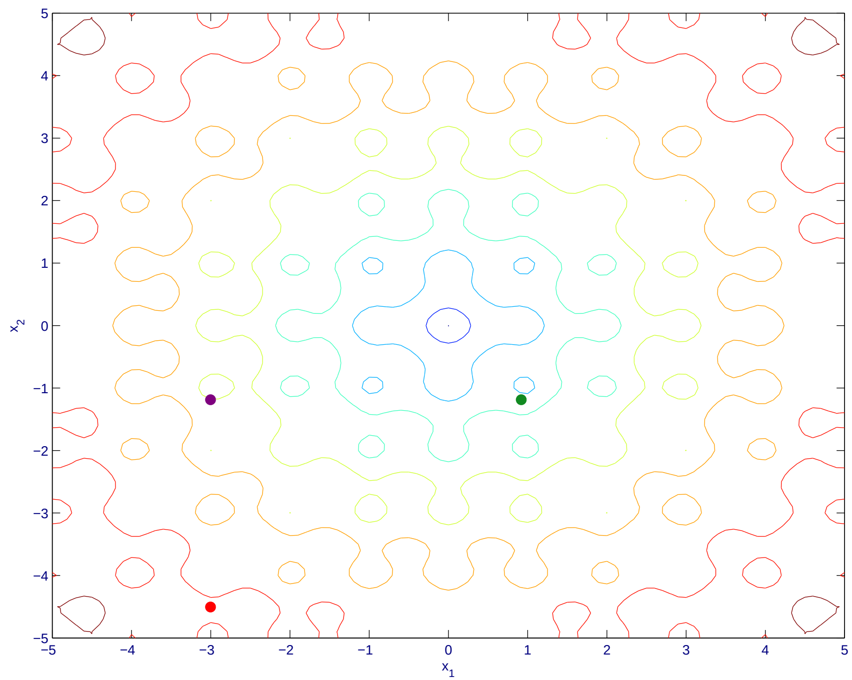
Example: Mutation



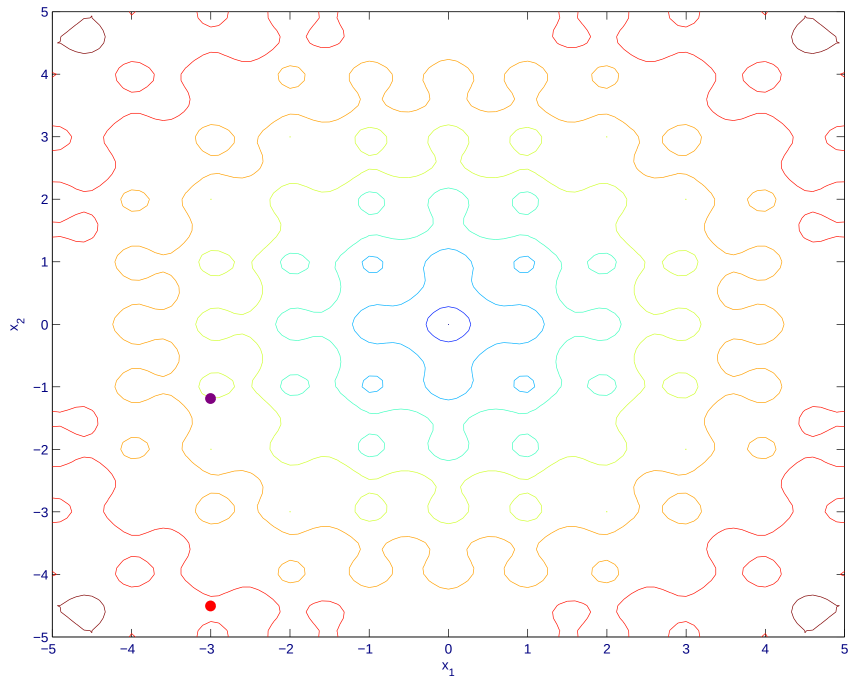
Example: Mutation



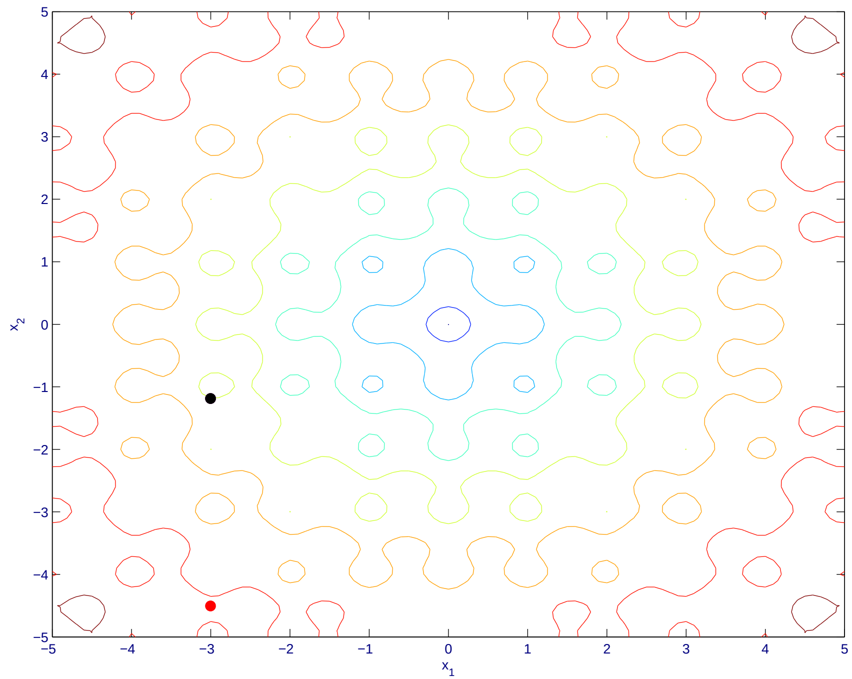
Example: Recombination



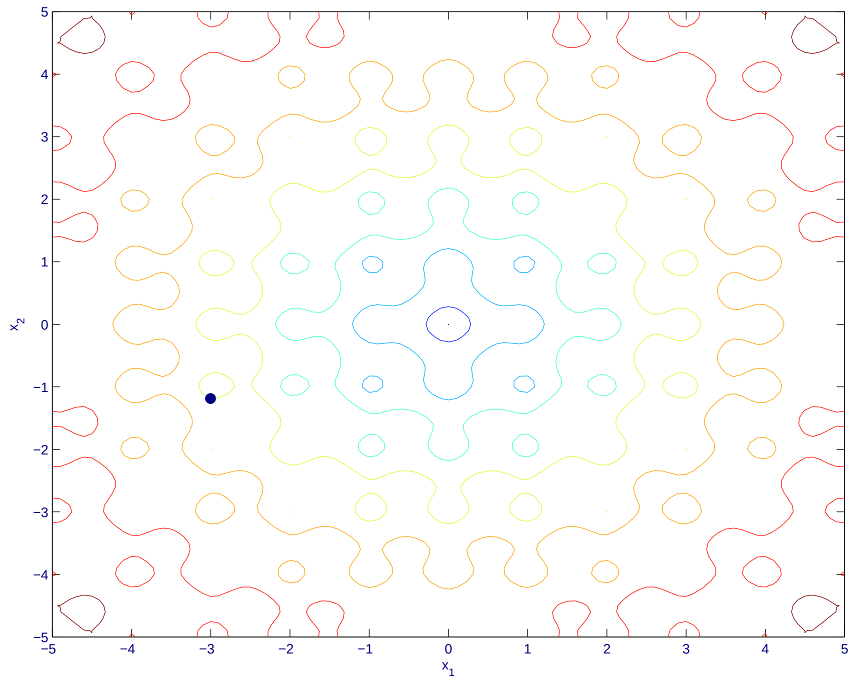
Example: Selection



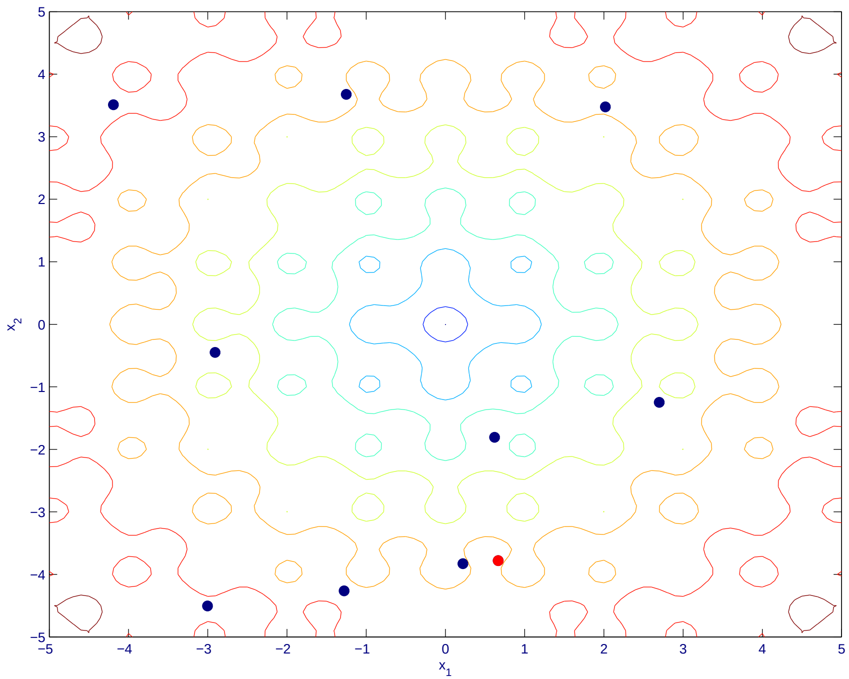
Example: Selection



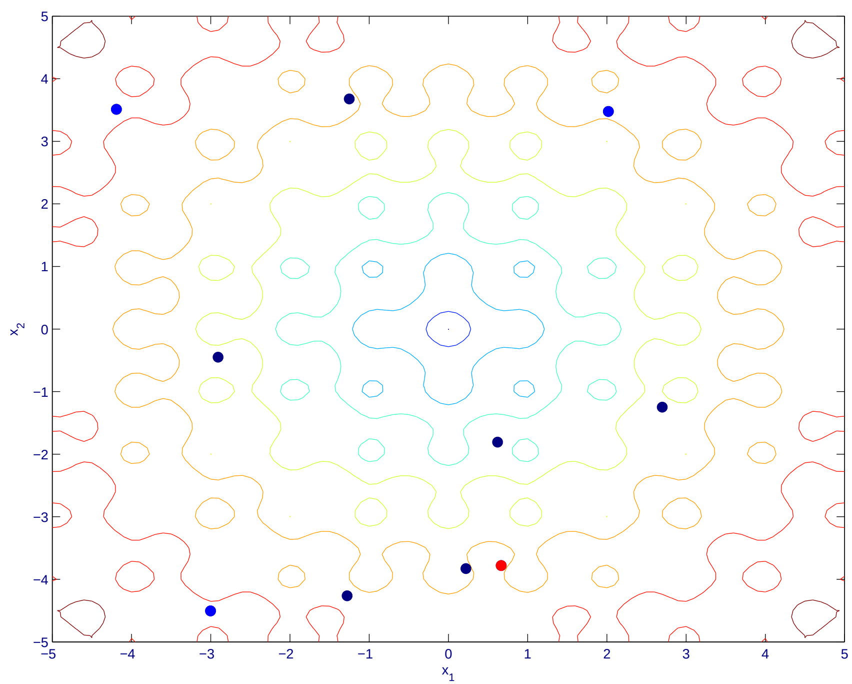
Example: Selection



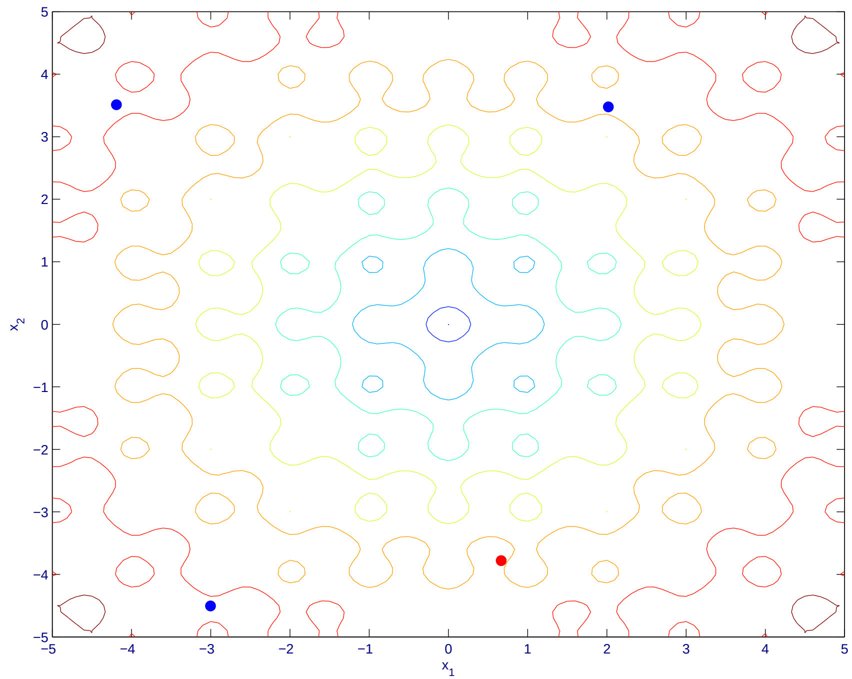
Example: Mutation



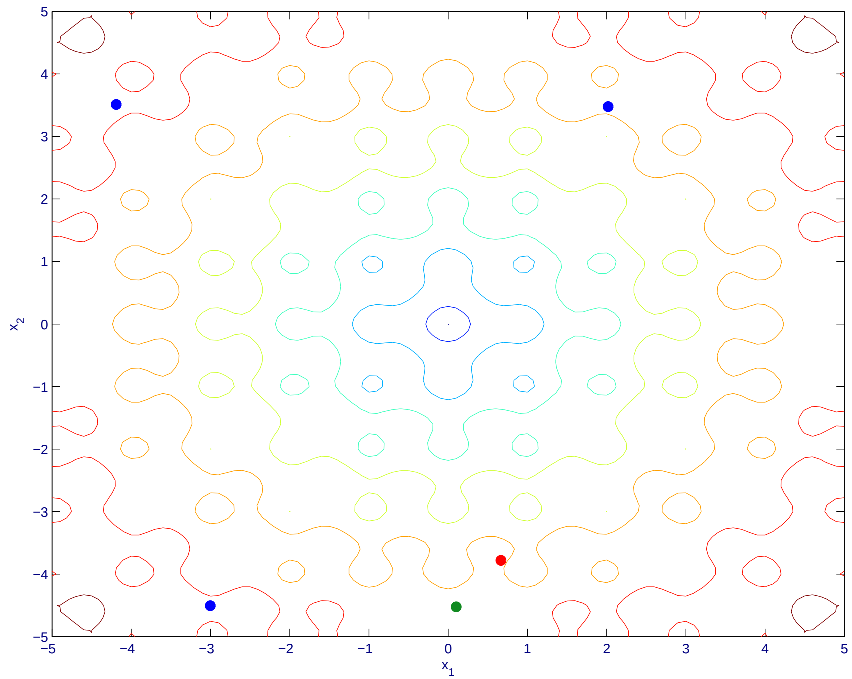
Example: Mutation



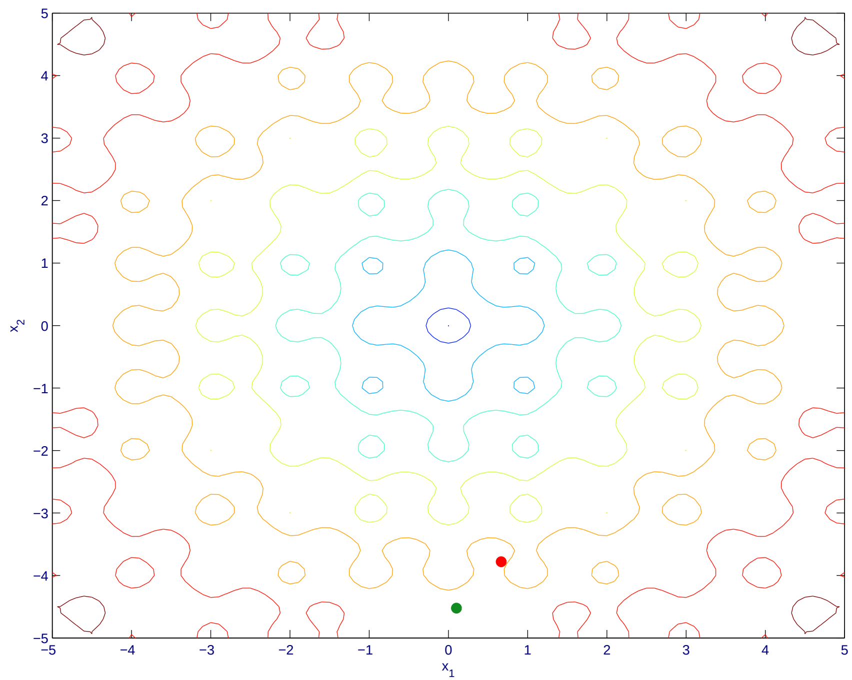
Example: Mutation



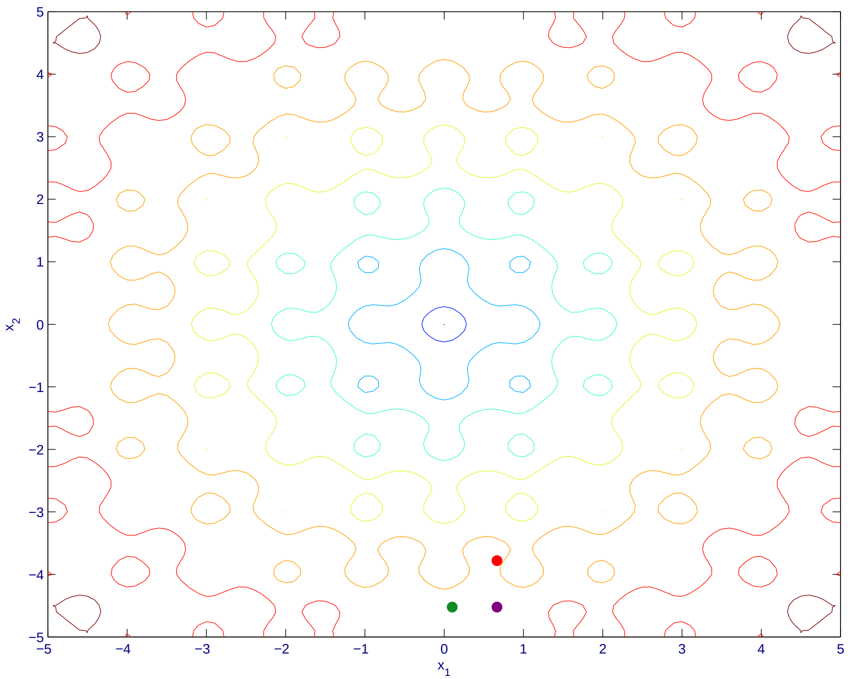
Example: Mutation



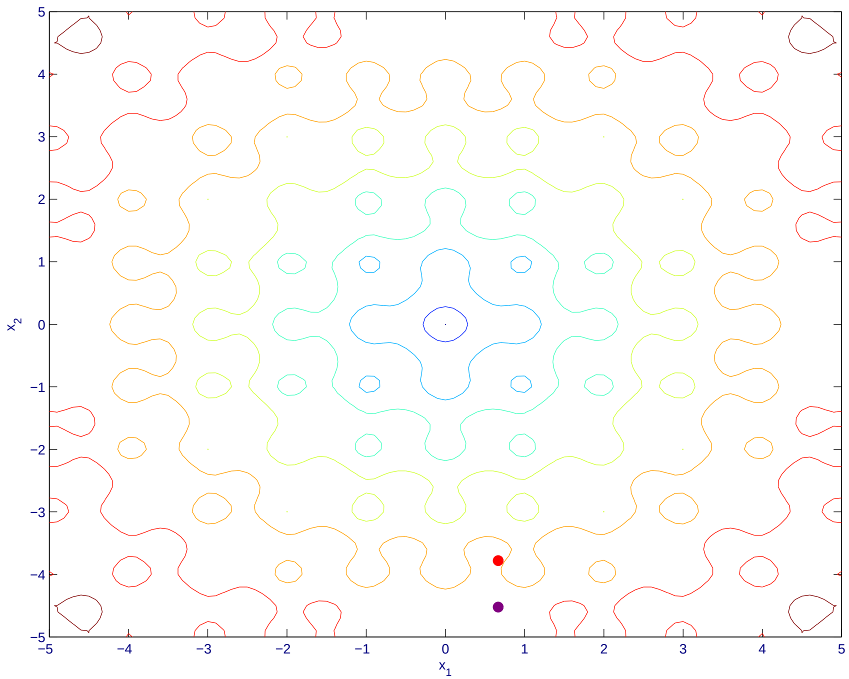
Example: Mutation



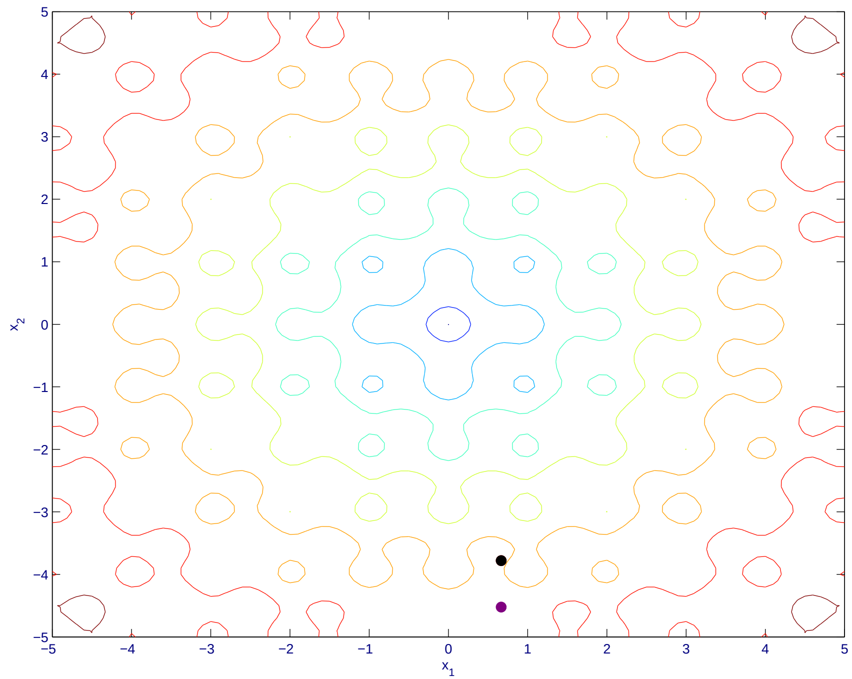
Example: Recombination



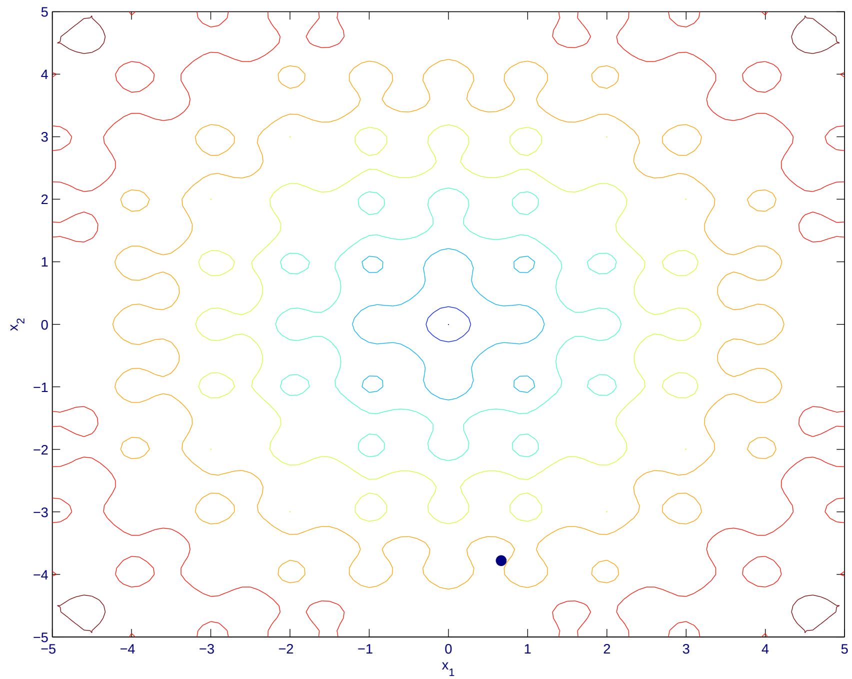
Example: Recombination



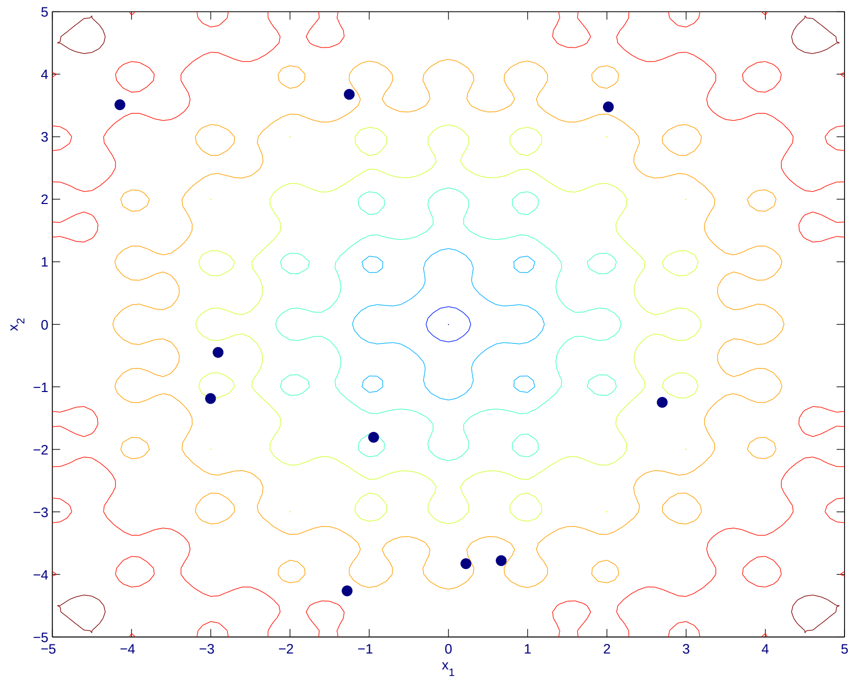
Example: Selection



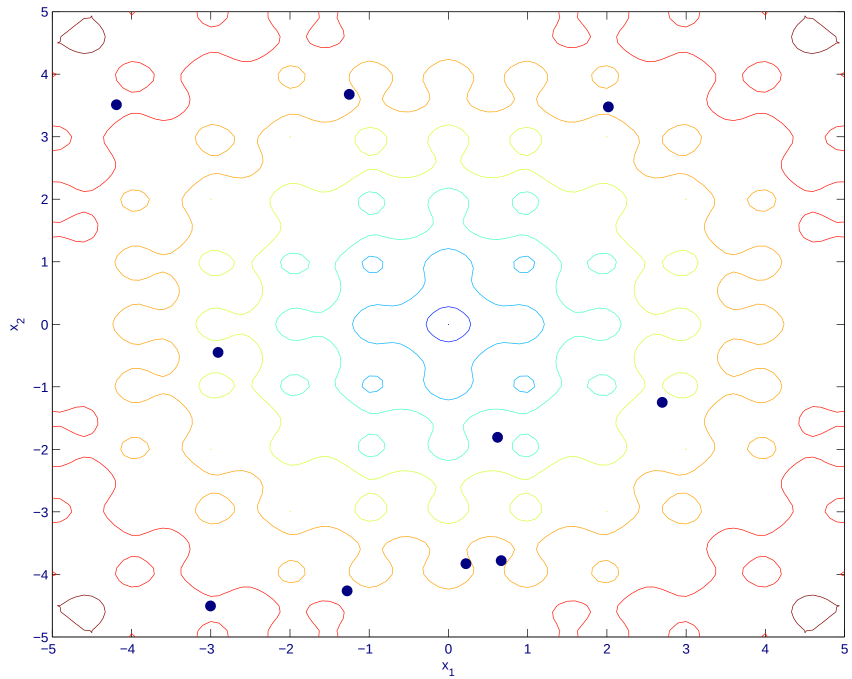
Example: Selection



Example: Generation 2



Example: Generation 1



Example: Movie

- Thirty generations of DE
- $N = 10$, $F = 0.5$ and $CR = 0.1$
- Ackley's function

Performance

- There is no proof of convergence for DE
- However it has been shown to be effective on a large range of classic optimisation problems
- In a comparison by Storn and Price in 1997 DE was more efficient than simulated annealing and genetic algorithms
- Ali and Törn (2004) found that DE was both more accurate and more efficient than controlled random search and another genetic algorithm
- In 2004 Lampinen and Storn demonstrated that DE was more accurate than several other optimisation methods including four genetic algorithms, simulated annealing and evolutionary programming

Recent Applications

- Design of digital filters
- Optimisation of strategies for checkers
- Maximisation of profit in a model of a beef property
- Optimisation of fermentation of alcohol

Further Reading

- Price, K.V. (1999), ‘An Introduction to Differential Evolution’ in Corne, D., Dorigo, M. and Glover, F. (eds), *New Ideas in Optimization*, McGraw-Hill, London.
- Storn, R. and Price, K. (1997), ‘Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces’, *Journal of Global Optimization*, 11, pp. 341–359.

Five most frequently used DE mutation schemes

"DE/rand/1": $\vec{V}_i(t) = \vec{X}_{r_1^i}(t) + F \cdot (\vec{X}_{r_2^i}(t) - \vec{X}_{r_3^i}(t)).$

"DE/best/1": $\vec{V}_i(t) = \vec{X}_{best}(t) + F.(\vec{X}_{r_1^i}(t) - \vec{X}_{r_2^i}(t)).$

"DE/target-to-best/1": $\vec{V}_i(t) = \vec{X}_i(t) + F.(\vec{X}_{best}(t) - \vec{X}_i(t)) + F.(\vec{X}_{r_1^i}(t) - \vec{X}_{r_2^i}(t)),$

"DE/best/2": $\vec{V}_i(t) = \vec{X}_{best}(t) + F.(\vec{X}_{r_1^i}(t) - \vec{X}_{r_2^i}(t)) + F.(\vec{X}_{r_3^i}(t) - \vec{X}_{r_4^i}(t)).$

"DE/rand/2": $\vec{V}_i(t) = \vec{X}_{r_1^i}(t) + F_1.(\vec{X}_{r_2^i}(t) - \vec{X}_{r_3^i}(t)) + F_2.(\vec{X}_{r_4^i}(t) - \vec{X}_{r_5^i}(t)).$

The general convention used for naming the various mutation strategies is DE/x/y/z, where DE stands for Differential Evolution, x represents a string denoting the vector to be perturbed, y is the number of difference vectors considered for perturbation of x, and z stands for the type of crossover being used (exp: exponential; bin: binomial)

GSL http://www.gnu.org/software/gsl/

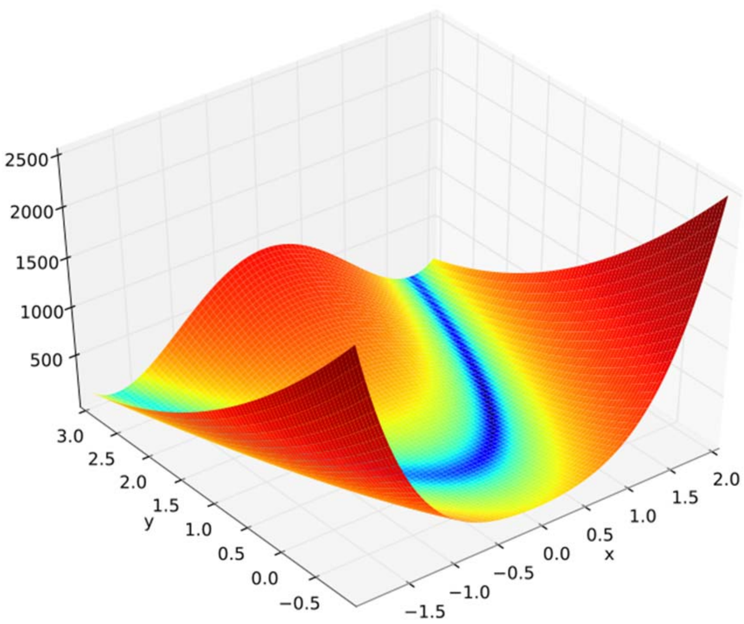
```
#include "gsl/gsl_rng.h"

gsl_rng *r; // global

    const gsl_rng_type *T;
    gsl_rng_env_setup();
    T = gsl_rng_default;
    r = gsl_rng_alloc(T);
    gsl_rng_uniform(r)
    gsl_rng_free(r);
```

Complex Numbers
Special Functions
Permutations
BLAS Support
Eigensystems
Quadrature
Quasi-Random Sequences
Statistics
N-Tuples
Simulated Annealing
Interpolation
Chebyshev Approximation
Discrete Hankel Transforms
Minimization
Physical Constants
Discrete Wavelet Transforms

Roots of Polynomials
Vectors and Matrices
Sorting
Linear Algebra
Fast Fourier Transforms
Random Numbers
Random Distributions
Histograms
Monte Carlo Integration
Differential Equations
Numerical Differentiation
Series Acceleration
Root-Finding
Least-Squares Fitting
IEEE Floating-Point
Basis splines

Exercício (para casa)

- Otimizar a função Rosenbrock



$$f(\boldsymbol{x}) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$$

$$-\infty \leq x_i \leq \infty$$

$$1 \leq i \leq n$$

$$\text{Min} = \begin{cases} n = 2 & \rightarrow \quad f(1,1) = 0, \\ n = 3 & \rightarrow \quad f(1,1,1) = 0, \\ n > 3 & \rightarrow \quad f\left(\underbrace{1,\ldots,1}_{(n)\ \text{times}}\right) = 0 \end{cases}$$