Введение в генетические алгоритмы и реализация их на языках C# и Rust

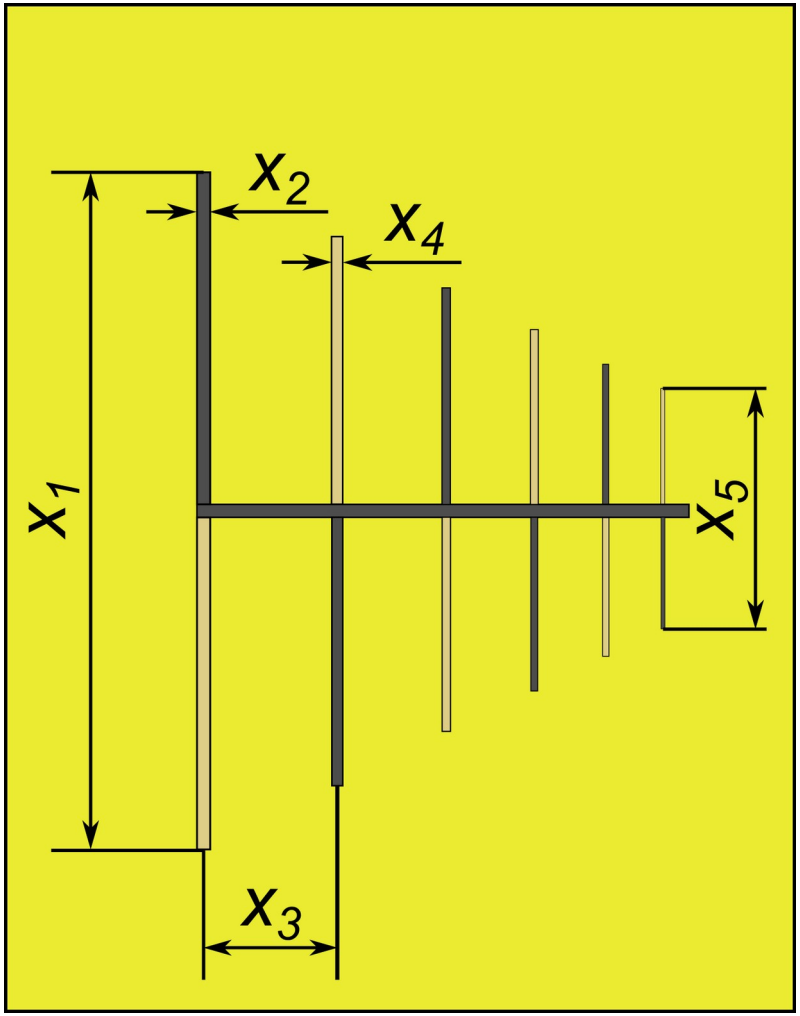
Евгений Ильин

МАИ, Институт №4 «Радиоэлектроника, инфокоммуникации и информационная безопасность»

Оптимизация

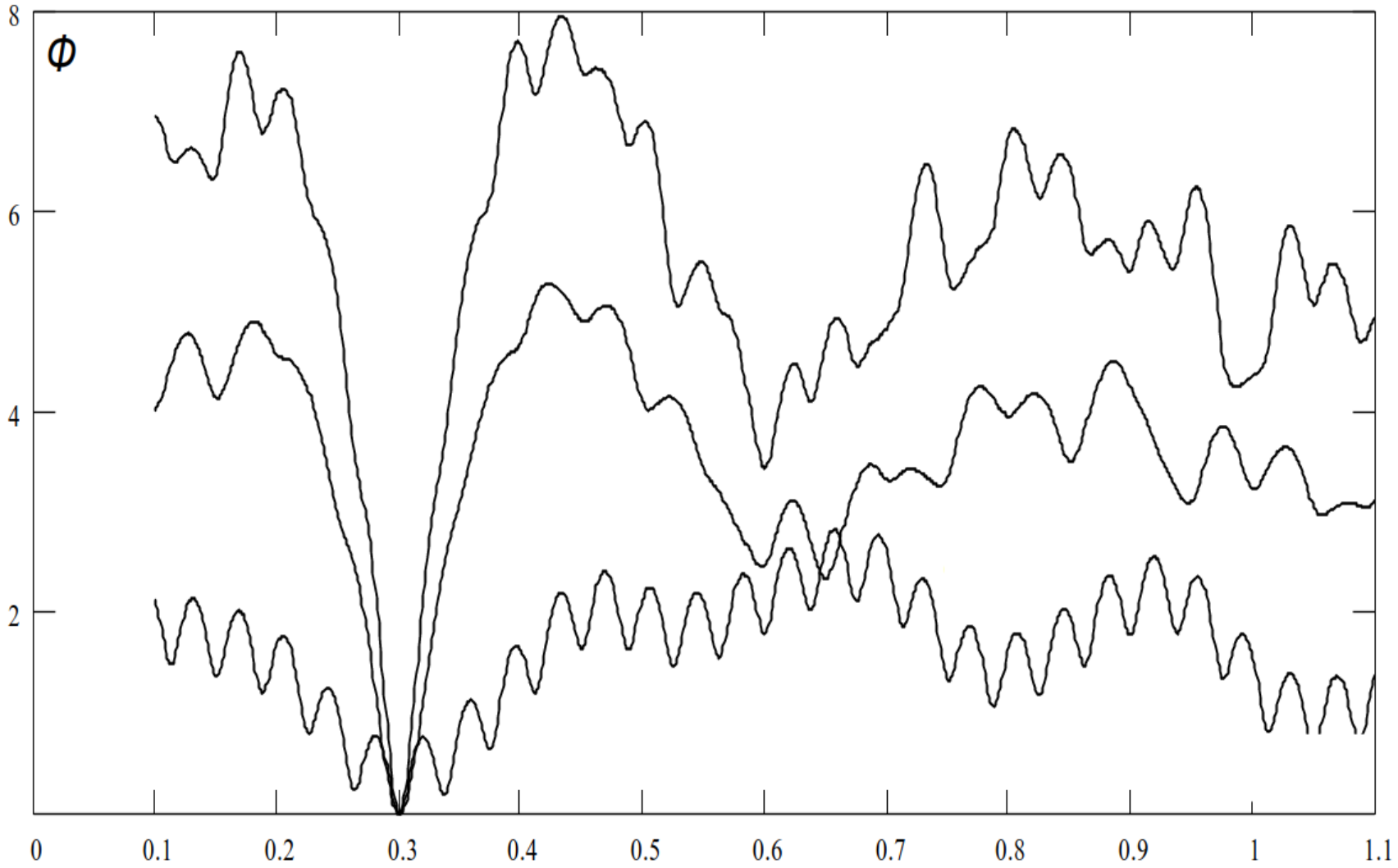
Оптимизация — (от лат. optimus — наилучший) задача нахождения экстремума (максимума или минимума) целевой функции в некоторой области конечномерного векторного пространства, ограниченной набором линейных и/или нелинейных равенств и/или неравенств.

Оптимизация параметров антенны с помощью генетического алгоритма

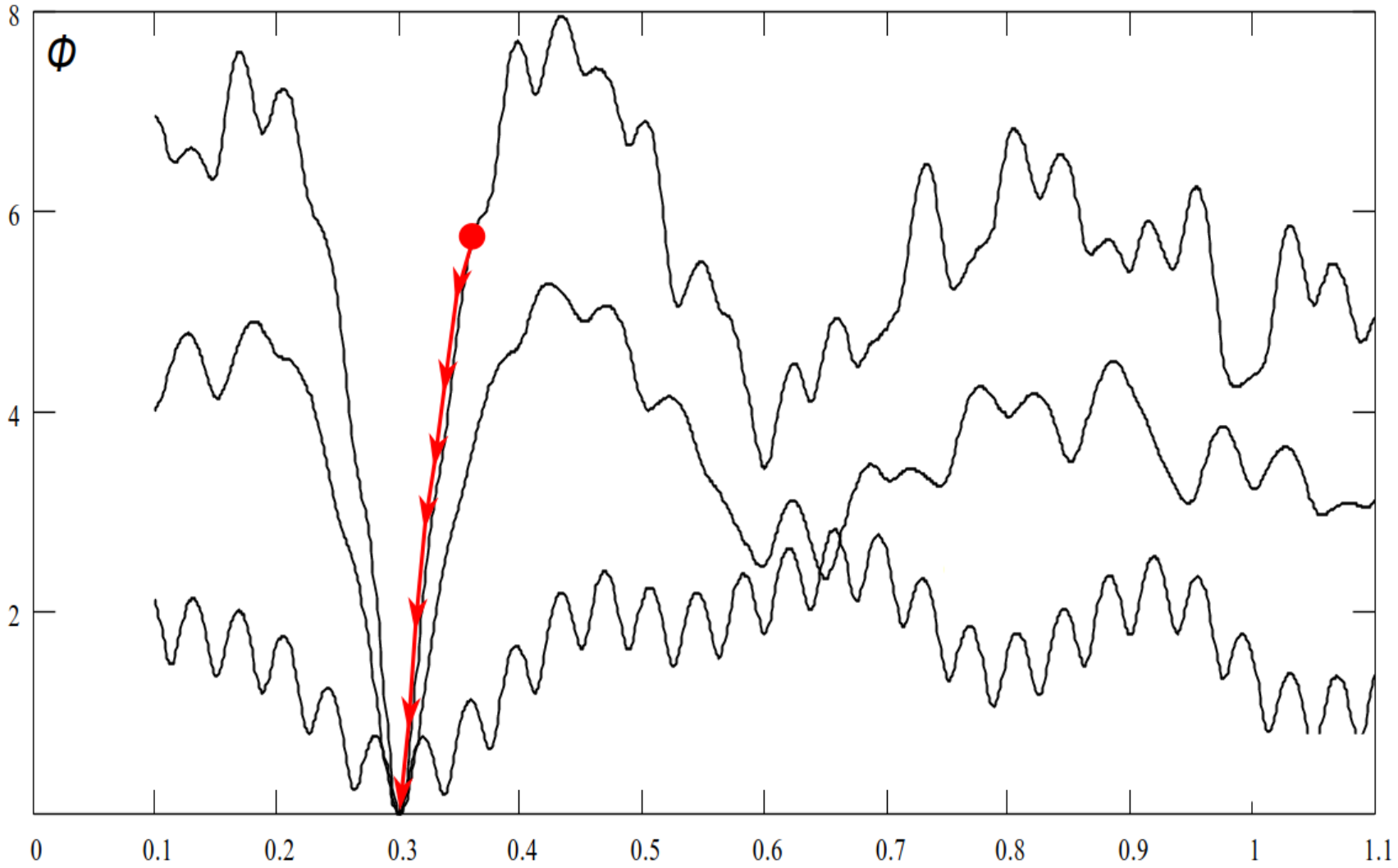


f = max(КСВ) в заданном диапазоне частот

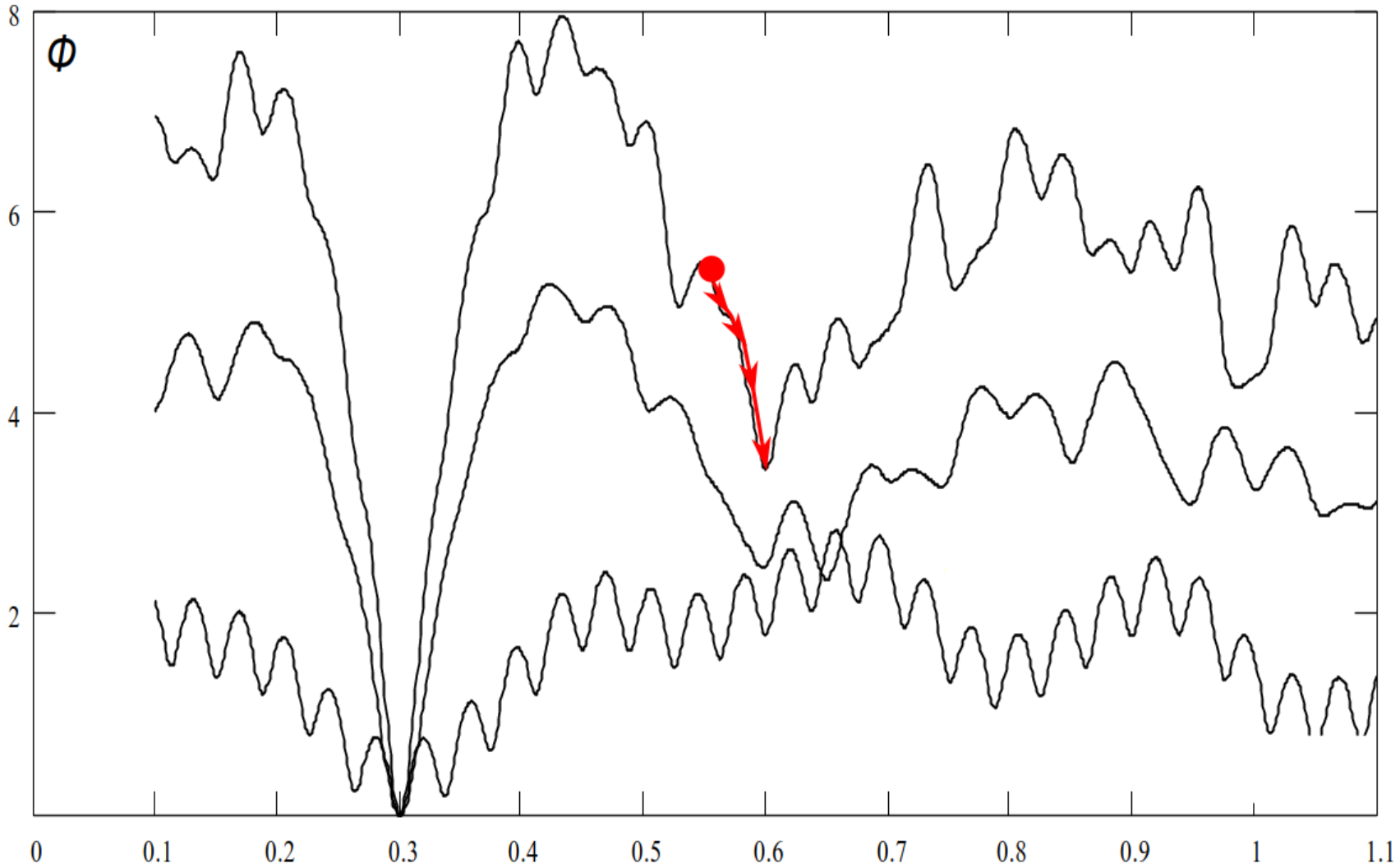
Примеры целевых функций



Сходимость градиентного метода



Сходимость градиентного метода



Алгоритмы глобальной оптимизации

- **Генетический алгоритм**
- Метод Нелдера-Мида
- Алгоритм случайного поиска
- Алгоритм имитации отжига
- Алгоритм роя частиц
- Алгоритм пчел
- И многие другие

Генетический алгоритм

J. H. Holland. “Adaptation in natural and artificial systems”.

University of Michigan Press, Ann Arbor, 1975.

Генетический алгоритм. Основные понятия

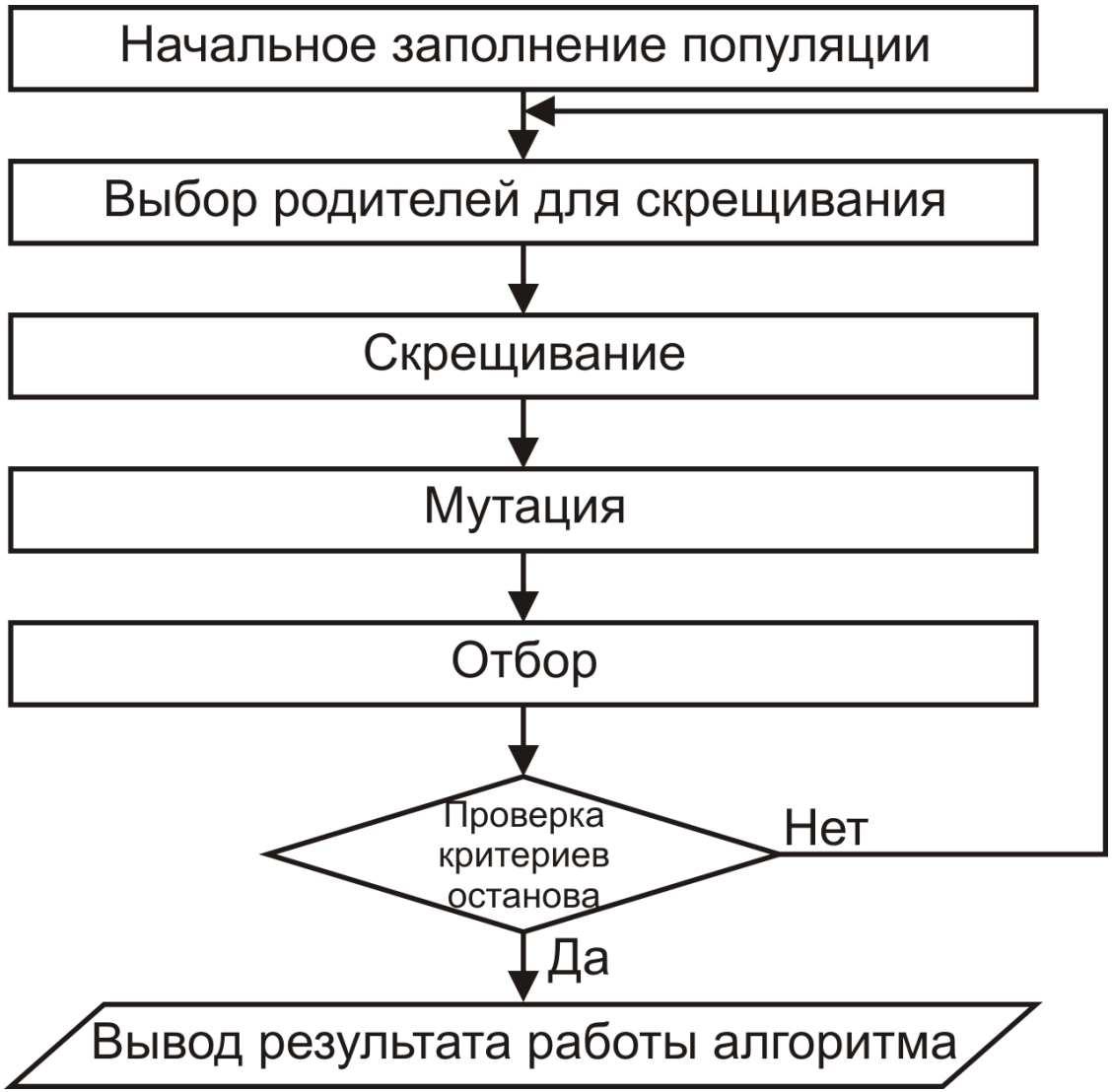
- **Хромосома** – значение одного из искомых параметров.

- **Особь** – одно из возможных решений (набор хромосом).

- **Популяция** – набор решений (особей).

- **Функция приспособленности** – минимизируемая (целевая) функция.

Генетический алгоритм



Начальное заполнение популяции

- Случайное распределение особей.
- Равномерное распределение особей.

Выбор родителей для скрещивания

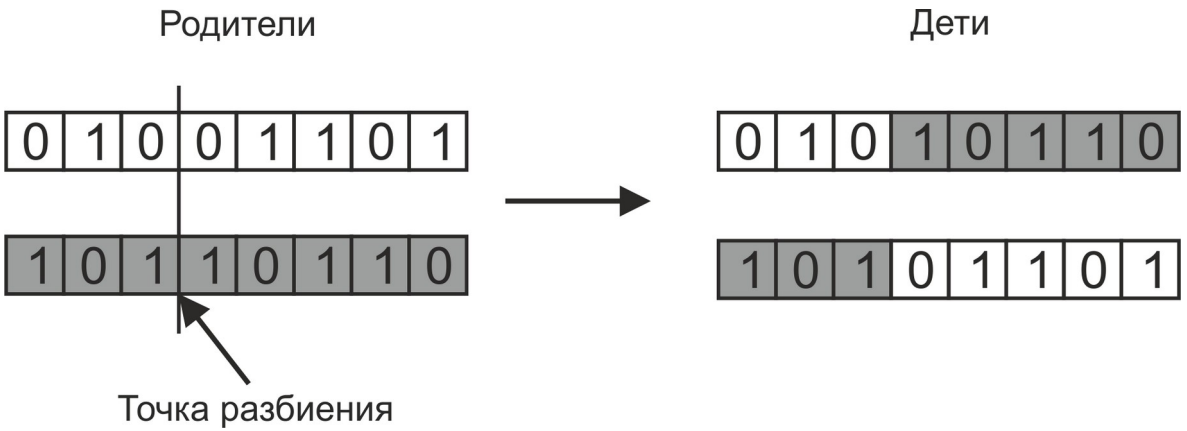
- Панмиксия.
- Инбридинг.
- Аутбридинг.
- Метод турнира.
- Метод элиты.

Операторы скрещивания

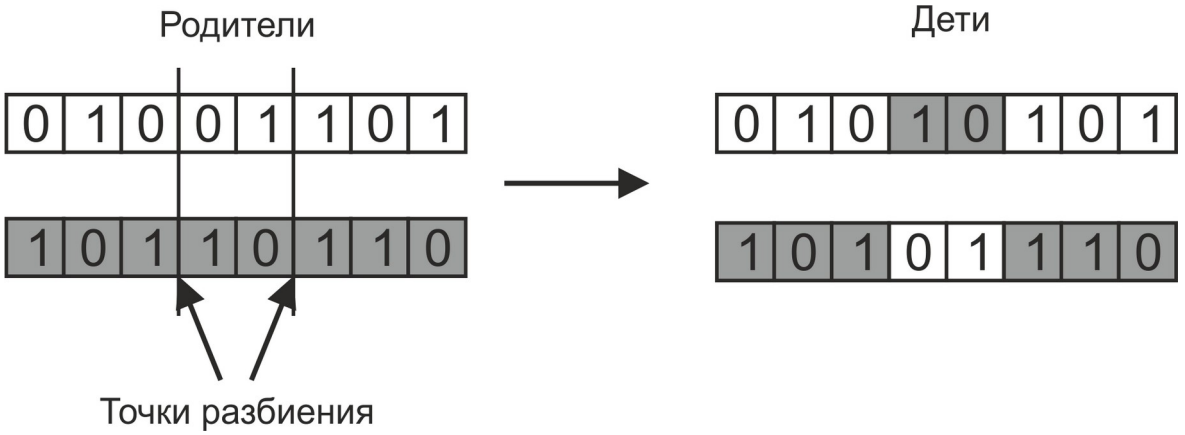
- Расчет среднего арифметического.
- Расчет среднего геометрического.
- Побитовое скрещивание.

Битовые операторы скрещивания

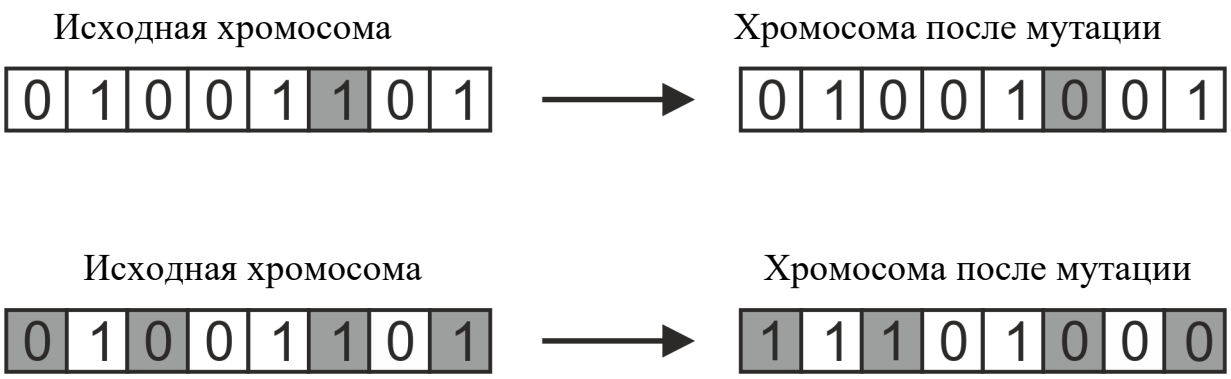
Одноточечное скрещивание



Многоточечное скрещивание



Операторы мутации



Другие алгоритмы мутации

- Добавление к хромосоме небольшой случайной величины
- Инвертирование всех битов хромосомы
- Замена хромосомы на случайное число

Операторы отбора

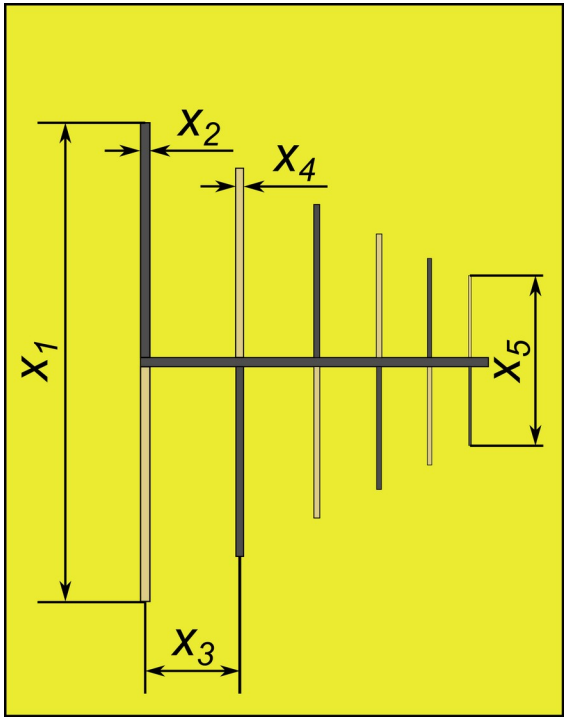
- Метод поддержания постоянного размера популяции
- Метод вероятностного отбора
- Метод турнира
- Метод элиты

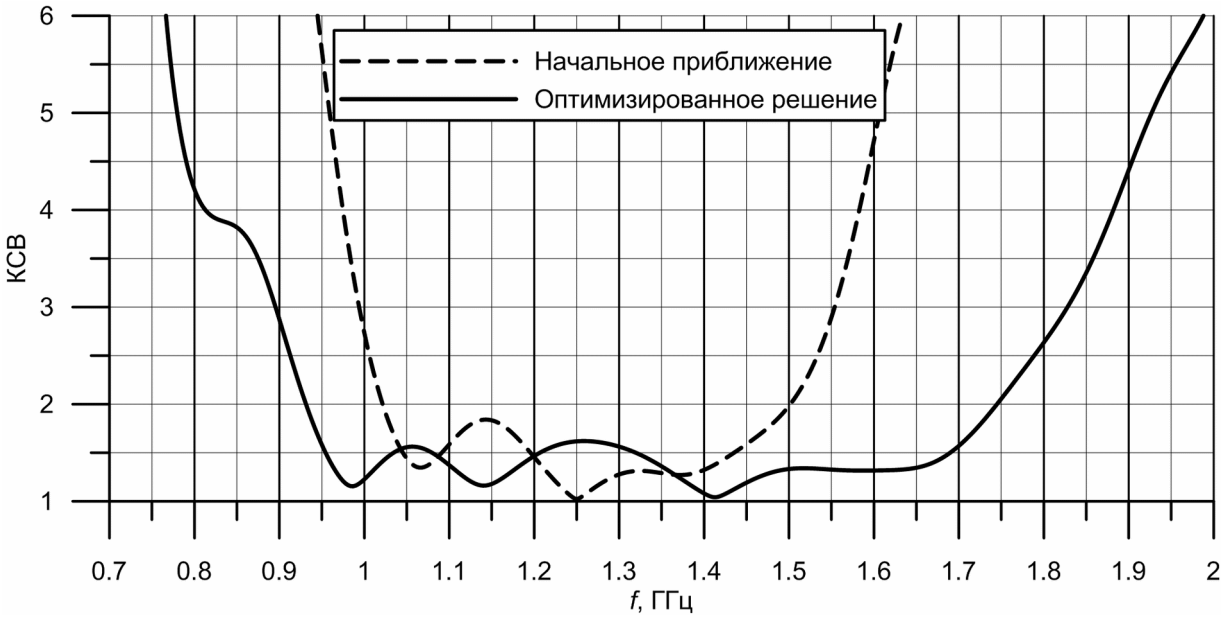
Критерии останова

- Постоянство целевой функции.
- Достижение заданного значения целевой функции.
- Достижение определенного номера поколения.
- Вырождение популяции.

Примеры использования

Оптимизация параметров антенны с помощью генетического алгоритма

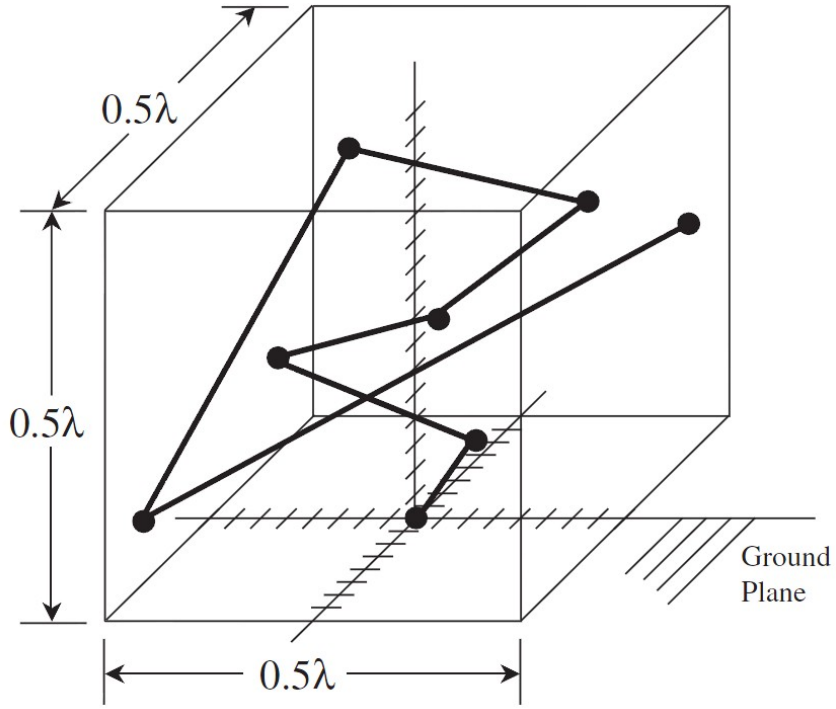


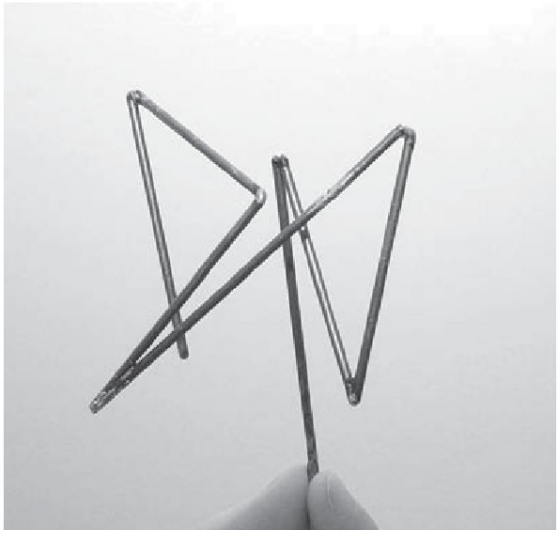


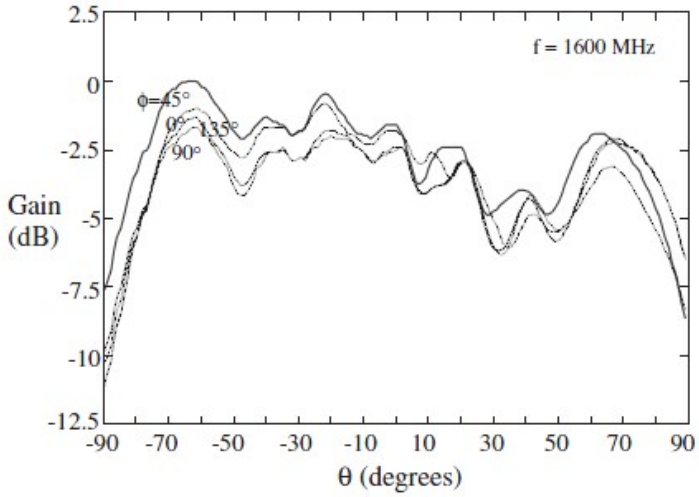
Параметры алгоритма:

- Минимизация КСВ в заданном диапазоне частот
- Количество оптимизируемых параметров: 3
- Размер популяции: 12 особей
- Вероятность мутации: 60%
- Количество поколений: 30
- Общее количество моделирований: 187
- Время одного расчета: 3 минуты

Пример применения генетического алгоритма



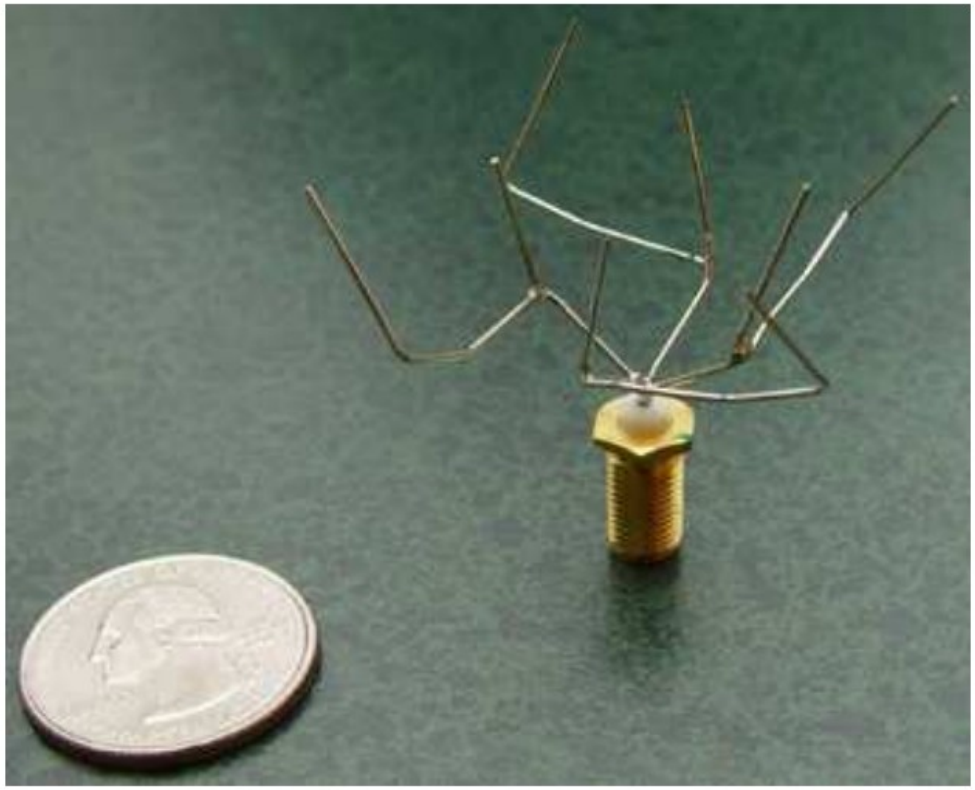


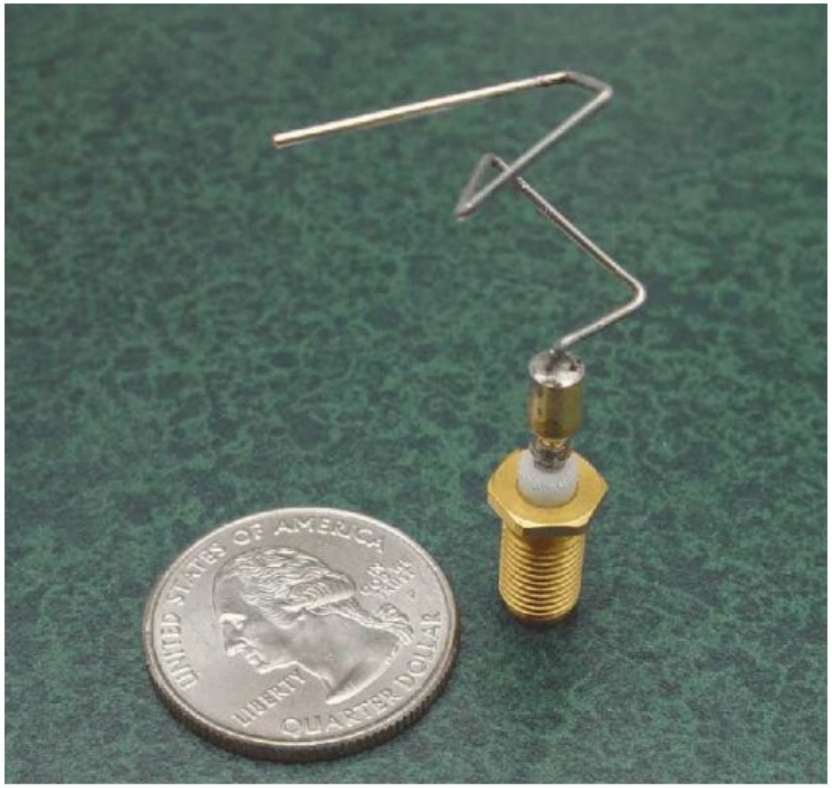


$$cost = \sum_{\text{over all } \theta,\, \phi} [gain(\theta, \phi) - average\ gain]^2$$

Stefanie Alki Delichatsios. GA Optimization for RFID Broadband Antenna Applications. MAS.862 May 22, 2006

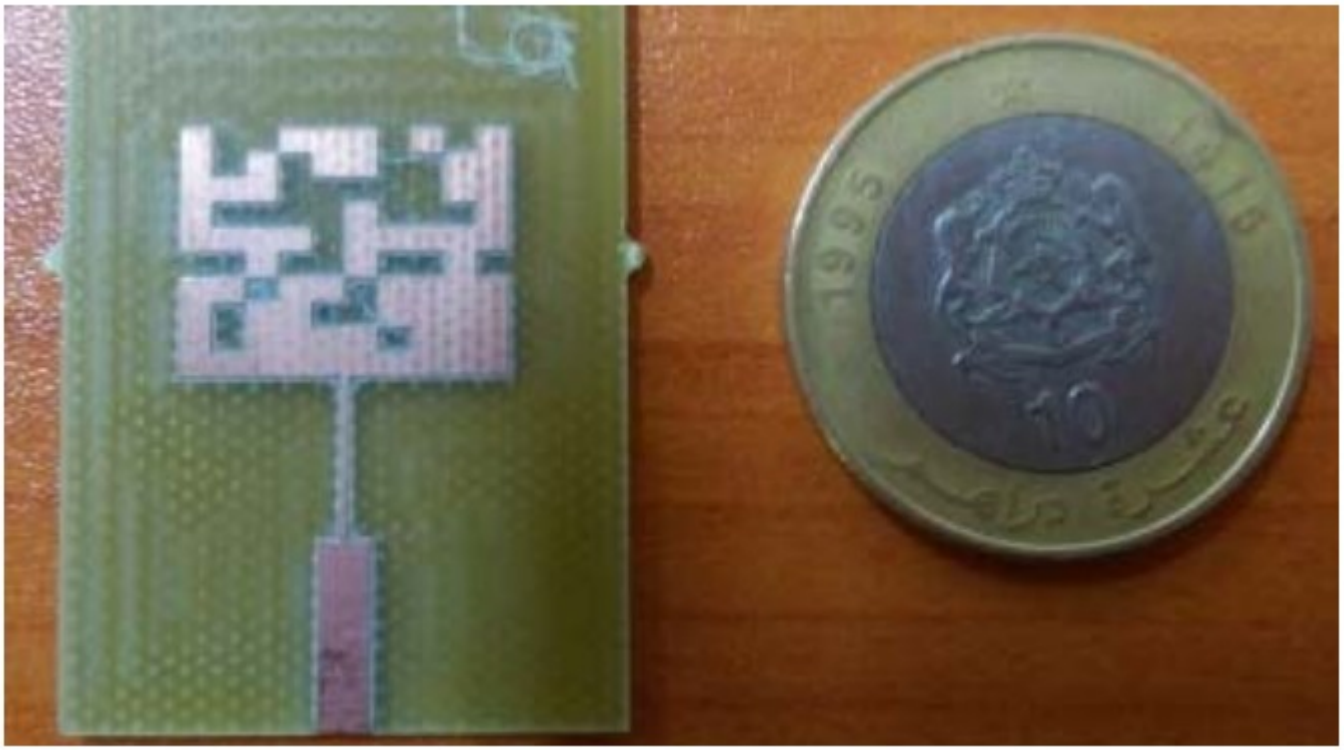
Пример применения генетического алгоритма





Hornby, Greg & Globus, Al & Linden, Derek & Lohn, Jason. (2006). Automated Antenna Design with Evolutionary Algorithms. Collection of Technical Papers - Space 2006 Conference. 1. 10.2514/6.2006-7242.

Пример применения генетического алгоритма



Mohammed Lamsalli, Abdelouahab El Hamichi, Mohamed Boussouis, Naima A. Touhami, and Taj-eddin Elhamadi. Genetic Algorithm Optimization for Microstrip Patch Antenna Miniaturization. Progress In Electromagnetics Research Letters, Vol. 60, 113–120, 2016

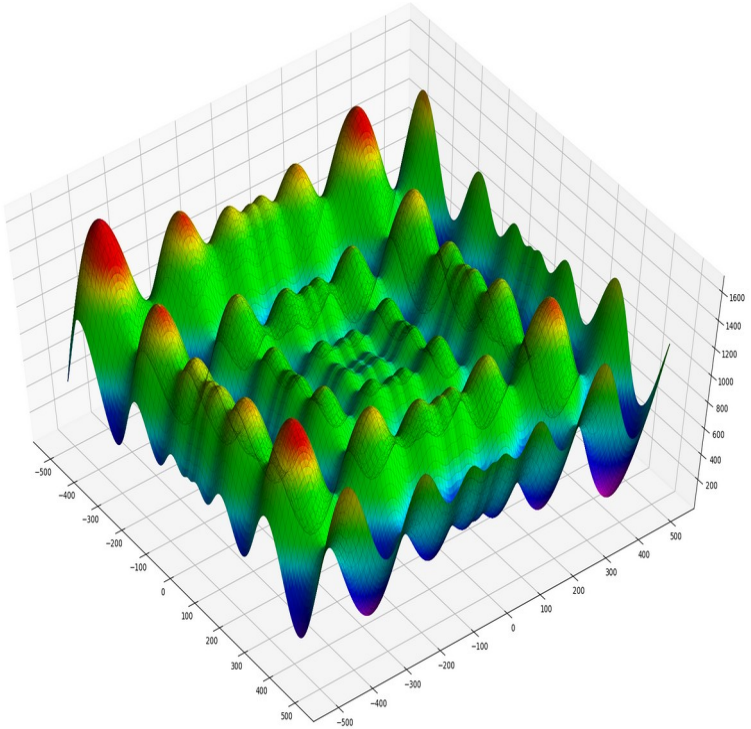
Реализация генетического алгоритма на C#

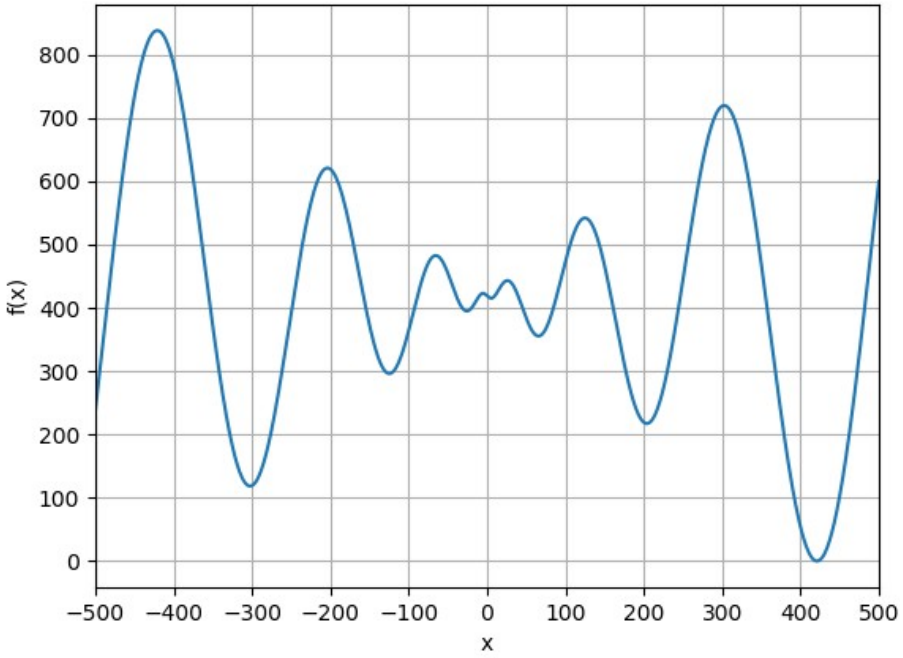
https://jenyay.net/Programming/Genetic

Тестовые функции. Функция Швефеля (Schwefel function)

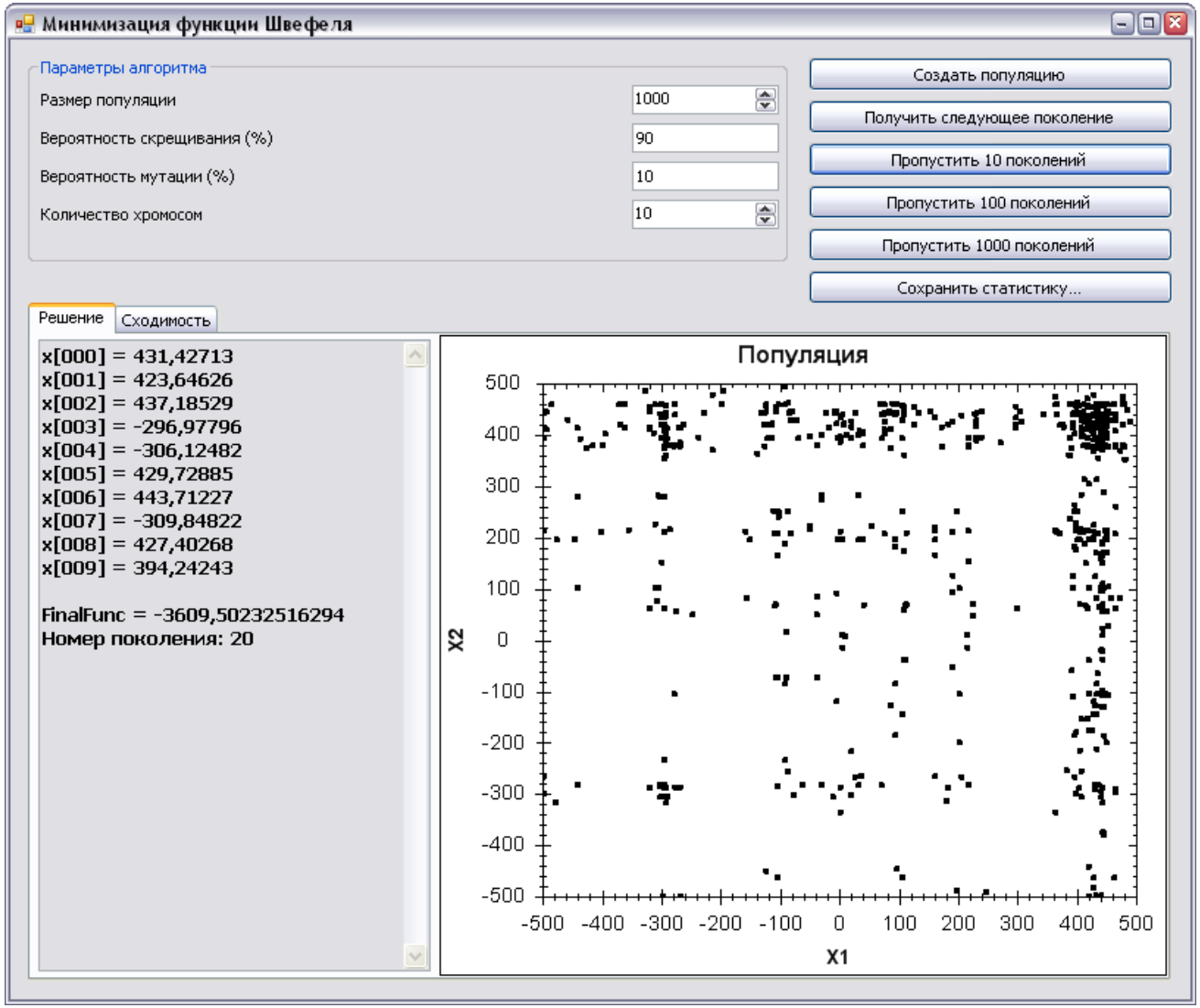
$$f(\mathbf{x}) = 418.9829n + \sum_{i=1}^{n} \left(-x_i \sin\left(\sqrt{|x_i|}\right)\right)$$

Глобальный минимум: $f(\mathbf{x}) = 0$ при
$x_i = 420.9687$, $i = 1, \ldots, n$; $-500 \le x_i \le 500$

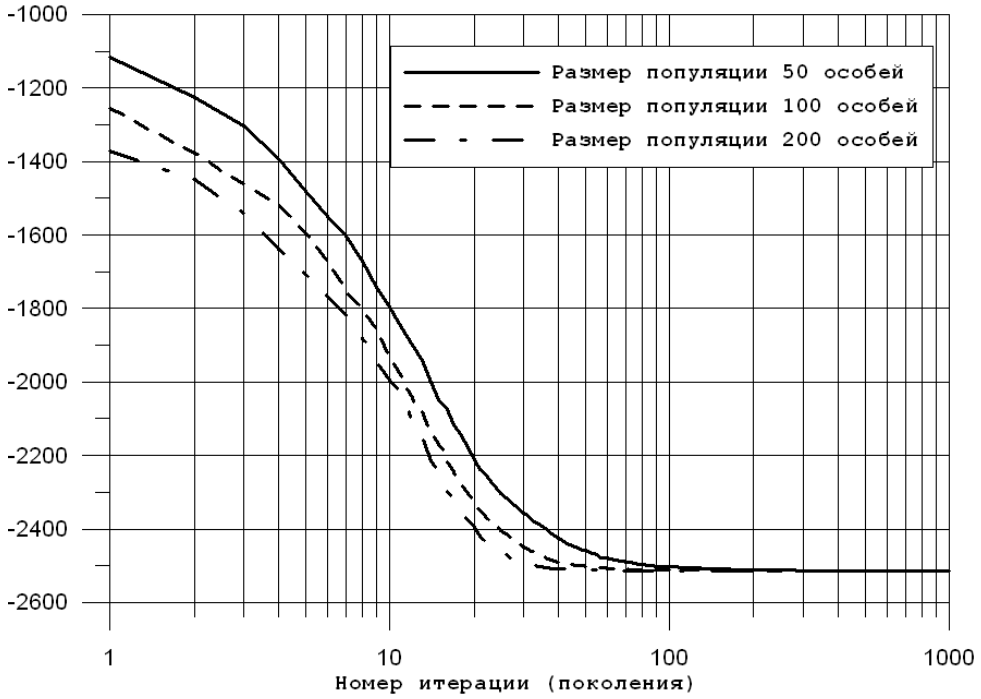




Демонстрация работы генетического алгоритма



Сходимость генетического алгоритма



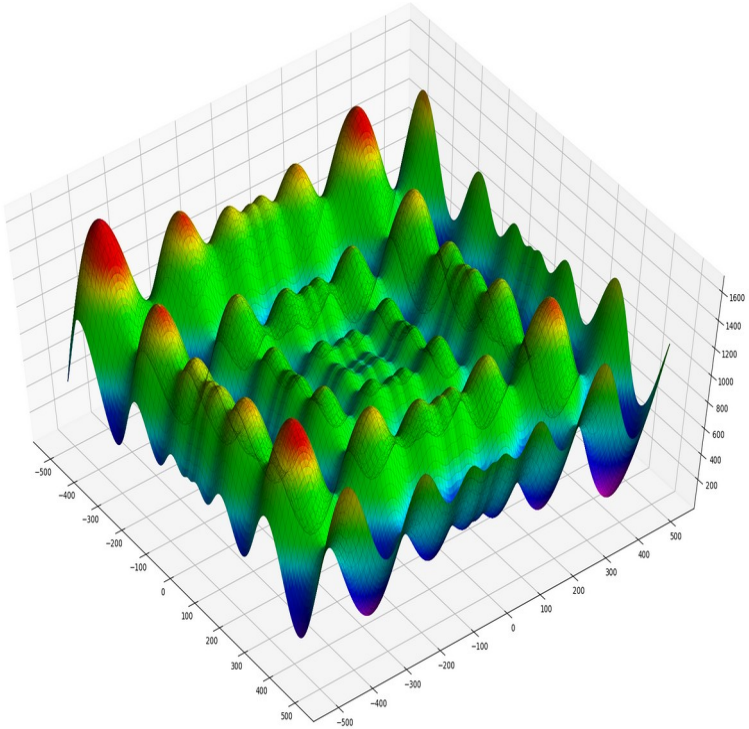
Реализация генетического алгоритма на Rust

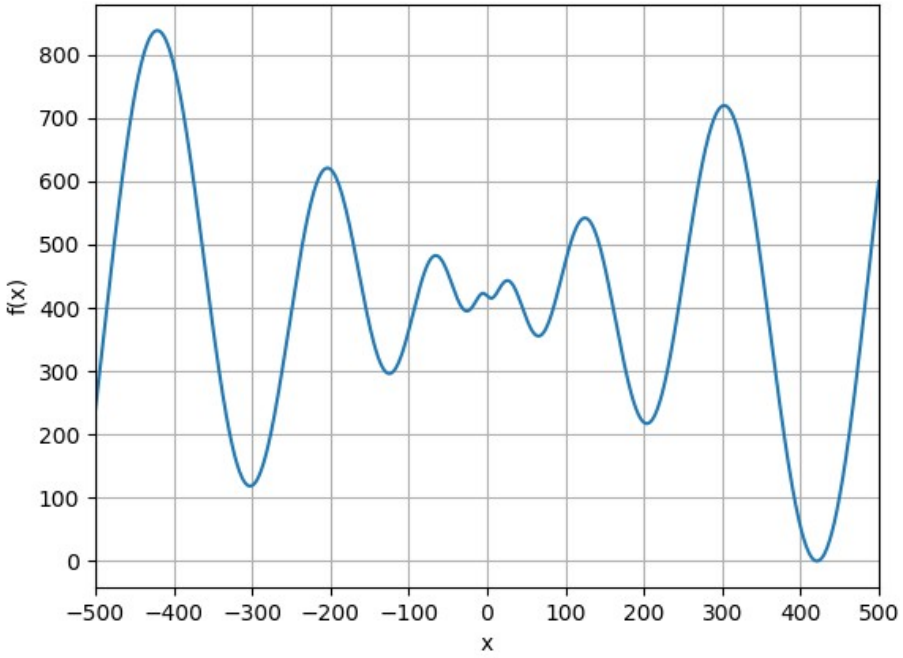
https://jenyay.net/Programming/OptlibGenetic

Тестовые функции. Функция Швефеля (Schwefel function)

$$f(\mathbf{x}) = 418.9829n + \sum_{i=1}^{n} \left(-x_i \sin\left(\sqrt{|x_i|}\right)\right)$$

Глобальный минимум: $f(\mathbf{x}) = 0$ при
$x_i = 420.9687$, $i = 1, \ldots, n$; $-500 \le x_i \le 500$





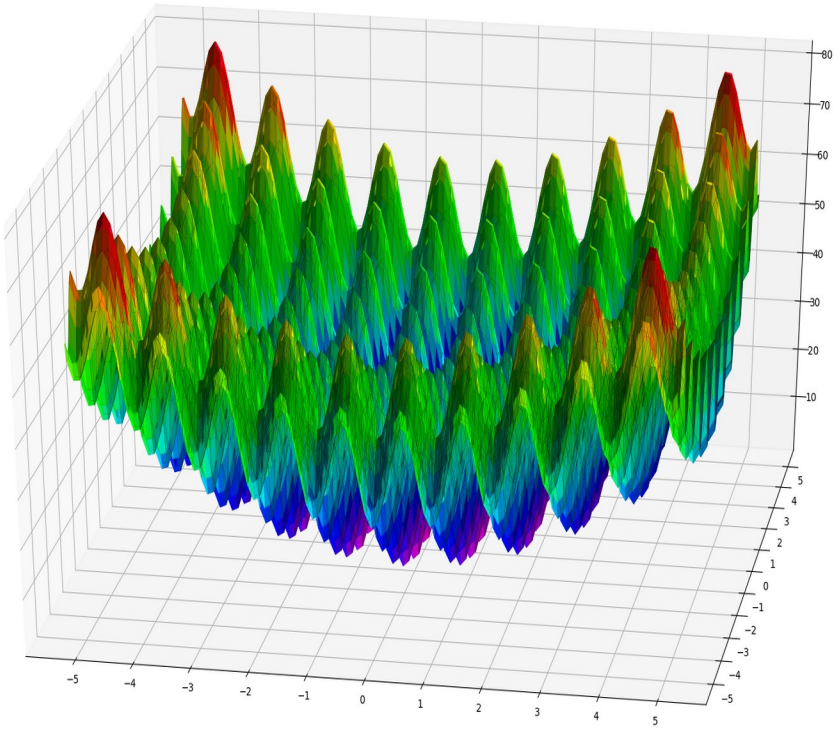
Тестовые функции. Функция Растригина

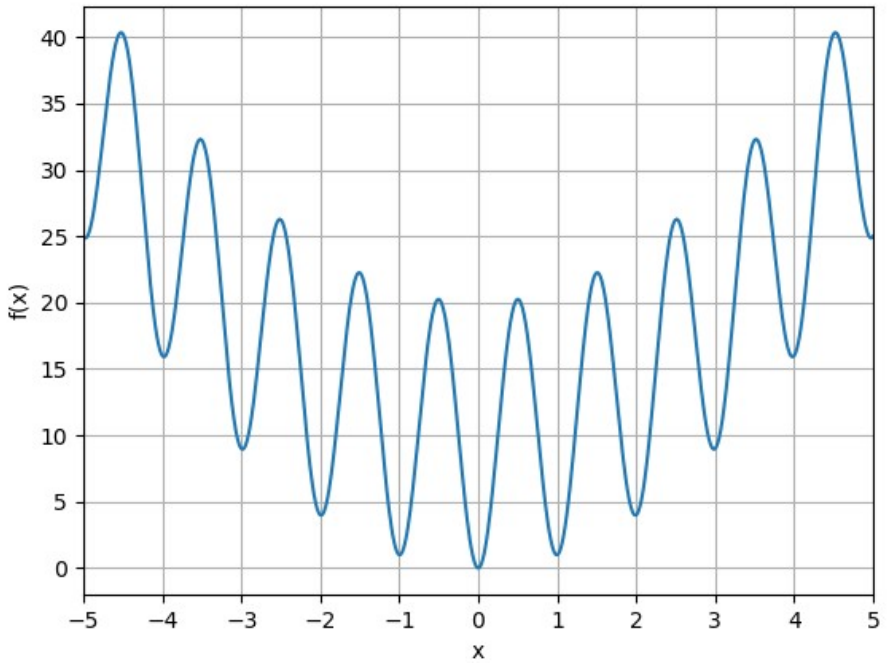
$$f(\mathbf{x}) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) \right)$$

Глобальный минимум: $f(\mathbf{x}) = 0$ при

$x_i = 0,\ i = 1, \ldots, n,$

$-5.12 \le x_i \le 5.12$





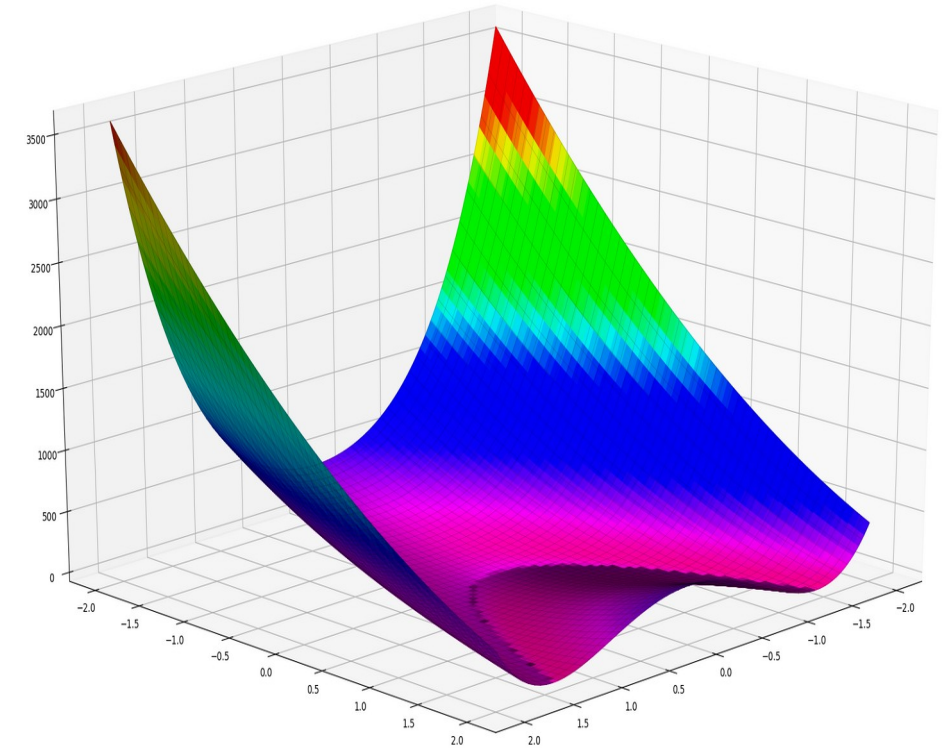
Тестовые функции. Функция Розенброка (Rosenbrock Function)

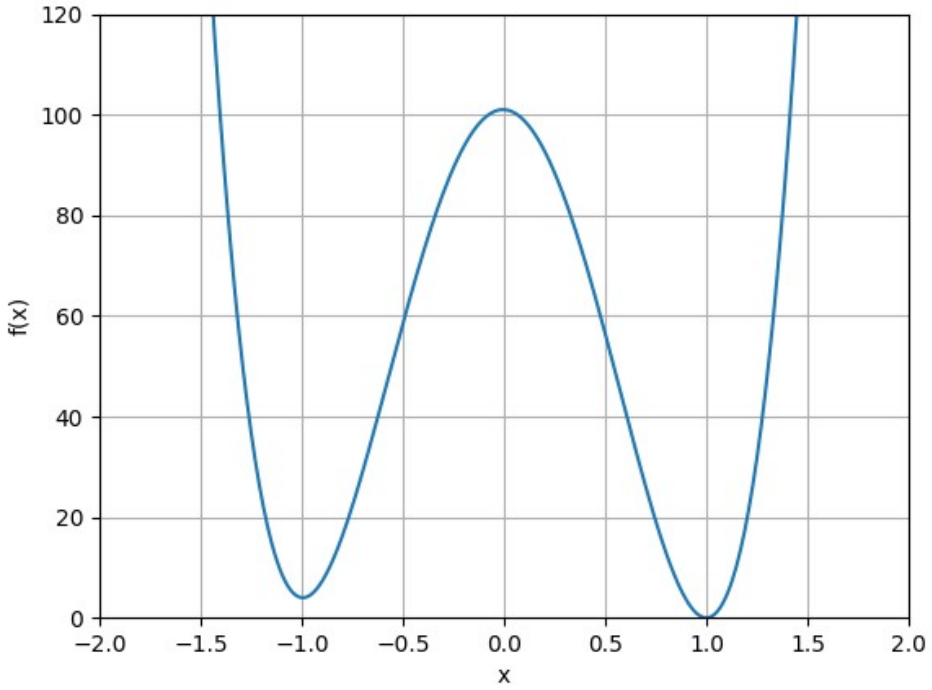
$$f(\mathbf{x}) = \sum_{i=1}^{d-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + \left(x_i - 1 \right)^2 \right]$$

Глобальный минимум: $f(\mathbf{x}) = 0$ при

$x_i = 1, i = 1, \ldots, n,$

$-2 \le x_i \le 2$





Реализация скрещивания на языке Rust

```rust
/// ```
/// use optlib::genetic::cross;
///
/// assert_eq!(cross::cross_u16(0b_0000_0000_0000_0000, 0b_1111_1111_1111_1111, 8), 0b_0000_0000_1111_1111);
/// ```
pub fn cross_u16(parent_1: u16, parent_2: u16, pos: usize) -> u16 {
    let size = mem::size_of::<u16>() * 8;
    let mask_parent_1 = !0u16 << pos;
    let mask_parent_2 = !0u16 >> (size - pos);
    (parent_1 & mask_parent_1) | (parent_2 & mask_parent_2)
}
```

```rust
/// ```
/// use optlib::genetic::cross;
///
/// assert_eq!(cross::cross_f32(0f32, f32::from_bits(std::u32::MAX), 4), f32::from_bits(0b_1111));
/// ```
pub fn cross_f32(parent_1: f32, parent_2: f32, pos: usize) -> f32 {
    let parent_1_bits = parent_1.to_bits();
    let parent_2_bits = parent_2.to_bits();

    let child_bits = cross_u32(parent_1_bits, parent_2_bits, pos);
    f32::from_bits(child_bits)
}
```

Реализация скрещивания на языке Rust

```rust
impl<T: Float> Cross<T> for FloatCrossExp {
    fn cross(&mut self, parents_genes: &[&T]) -> Vec<T> {
        assert_eq!(parents_genes.len(), 2);
        // mantissa: u64, exponent: i16, sign: i8
        let (mantissa_1, exponent_1, sign_1) = parents_genes[0].integer_decode();
        let (mantissa_2, exponent_2, sign_2) = parents_genes[1].integer_decode();

        let mantissa_size = mem::size_of_val(&mantissa_1) * 8;
        let exponent_size = mem::size_of_val(&exponent_1) * 8;

        let mantissa_between = Uniform::new(1, mantissa_size);
        let exponent_between = Uniform::new(1, exponent_size);

        let mantissa_pos = mantissa_between.sample(&mut self.random);
        let exponent_pos = exponent_between.sample(&mut self.random);

        let mantissa_child = cross_u64(mantissa_1, mantissa_2, mantissa_pos);
        let exponent_child = cross_i16(exponent_1, exponent_2, exponent_pos);

        let sign_child = match Uniform::new_inclusive(0i8, 1i8).sample(&mut self.random) {
            0 => sign_1,
            1 => sign_2,
            _ => panic!("Invalid random value in FloatCrossExp"),
        };

        vec![
            T::from(sign_child).unwrap()
                * T::from(mantissa_child).unwrap()
                * T::from(exponent_child).unwrap().exp2(),
        ]
    }
}
```

Реализация мутации на языке Rust

```rust
impl Mutation<f64> for BitwiseMutation {
    fn mutation(&mut self, gene: &f64) -> f64 {
        let size = mem::size_of::<f64>() * 8;
        let between = Uniform::new(0, size);

        let mut bit_value = gene.to_bits();
        for _ in 0..self.change_gene_count {
            let pos = between.sample(&mut self.random);
            bit_value ^= 1 << pos;
        }
        f64::from_bits(bit_value)
    }
}
```

Недостатки генетического алгоритма

- Не гарантируется нахождение глобального экстремума.
- Использование вероятностных операций.
- Большое количество вычислений целевой функции.
- Большое количество параметров алгоритма.

Модификации генетического алгоритма

- Совместное применение с итерационными алгоритмами.
- Создание нескольких независимых популяций.
- Адаптация параметров алгоритма во время выполнения.
- Использование ГА для определения оптимальных параметров другого ГА.

Контакты

Евгений Ильин.

Введение в генетические алгоритмы
и реализация их на языках C# и Rust

E-mail: jenyay.ilin@gmail.com
Сайт: https://jenyay.net
Github: https://github.com/jenyay
Телеграм: @jenyay
VK: https://vk.com/jenyay

https://jenyay.net/Programming/Genetic
https://jenyay.net/Programming/OptlibGenetic