Derivative-Free Optimization

Zhao Chi

St.Petersburg State University

dandanv5@hotmail.com

November 22, 2019

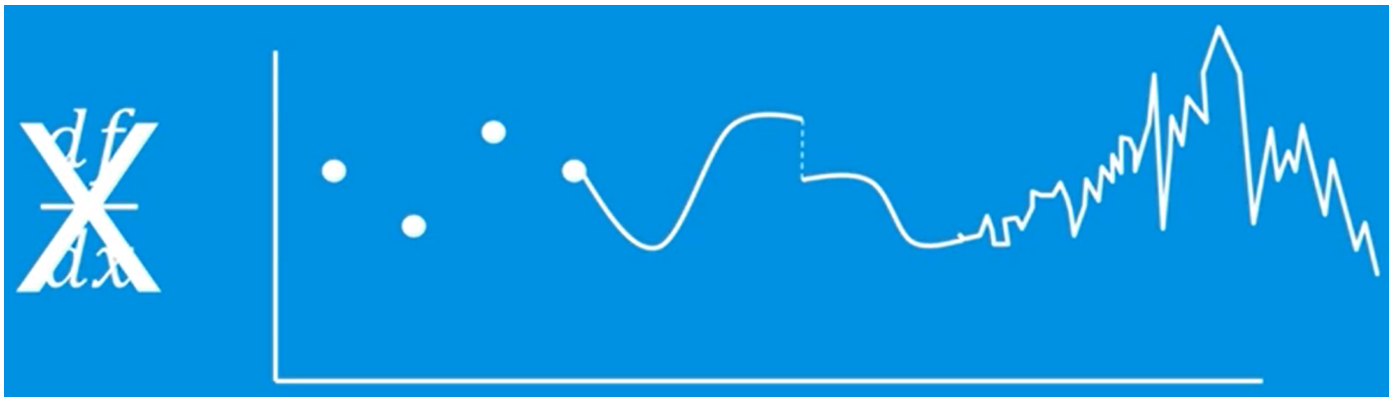
Table of contents slide

Overview in Derivative-Free Optimization

Gradient based algorithms and gradient free algorithms are the two main types of methods for solving optimization problems.

difference

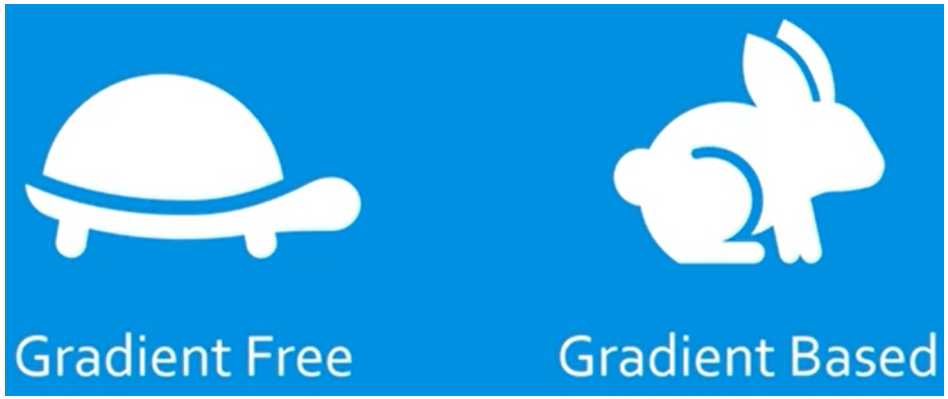
The main **difference** between gradient based and gradient free algorithms is that gradient free algorithms do not require derivatives. This means that they can be used for optimization problems where derivatives can't be obtained or are difficult to obtain. This can include functions that are discrete, discontinuous, or noisy. This makes gradient free algorithms very flexible in the types of problems they can be applied to.



Overview in Derivative-Free Optimization

disadvantage

The major **disadvantage** of gradient free algorithms is that they are generally much slower than gradient based algorithms.

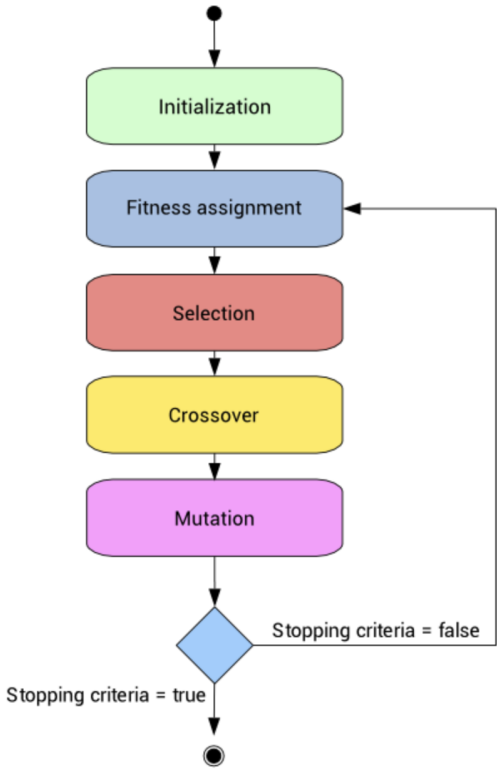


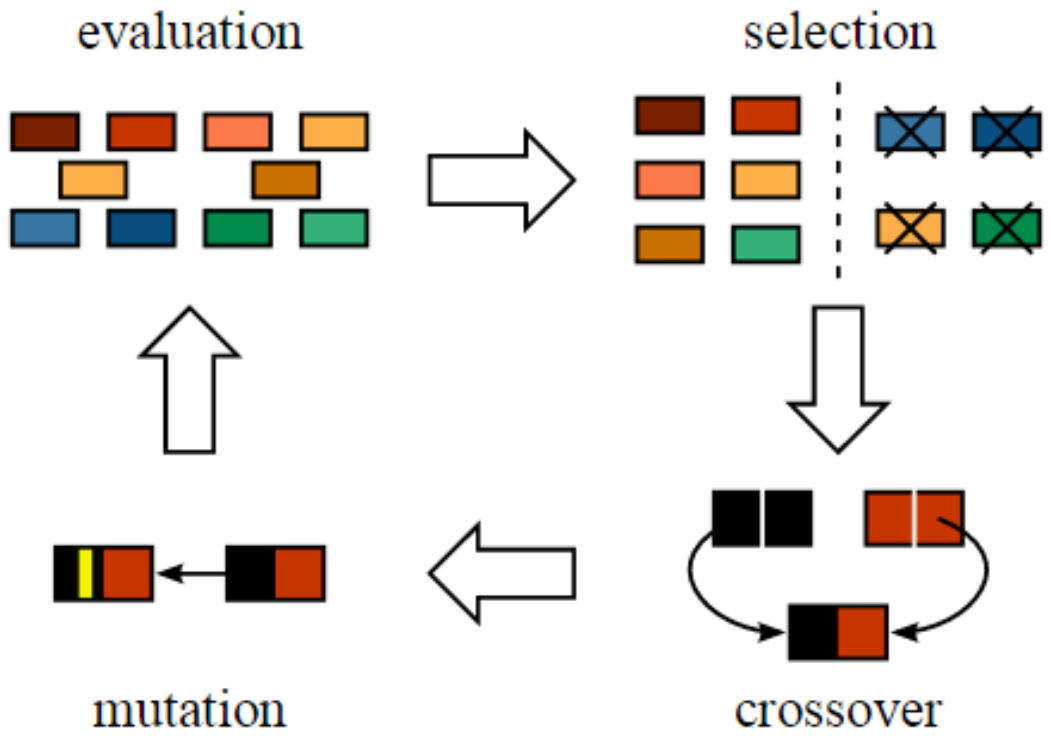
Exhaustive Search

First off, Exhaustive Search. The simplest, and most inefficient gradient free optimization method is to try every possible solution and pick the best answer. While this approach may work for very small problems, with larger problems it quickly becomes impossible.

Genetic Algorithms

Genetic algorithms are another type of gradient free optimization algorithm. Genetic algorithms are based on the ideas of biology and evolution. Instead of proposing just a single solution to an optimization problem, a genetic algorithm generates many possible solutions that form a "population".

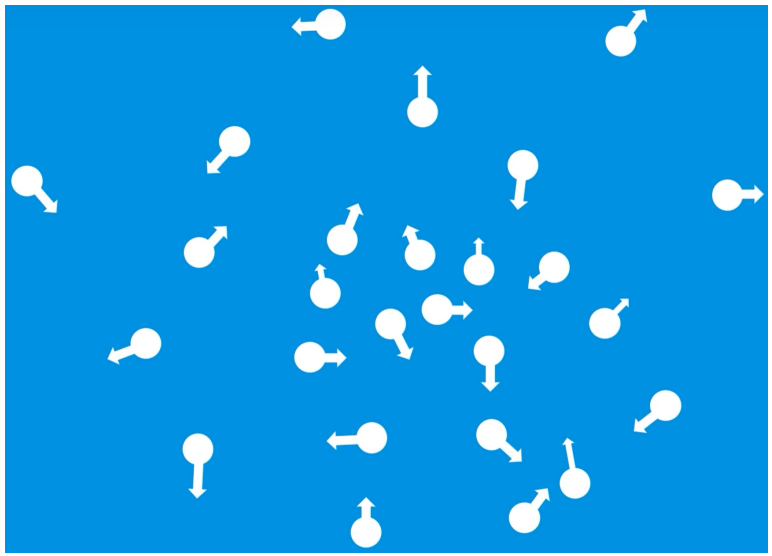




Particle Swarm

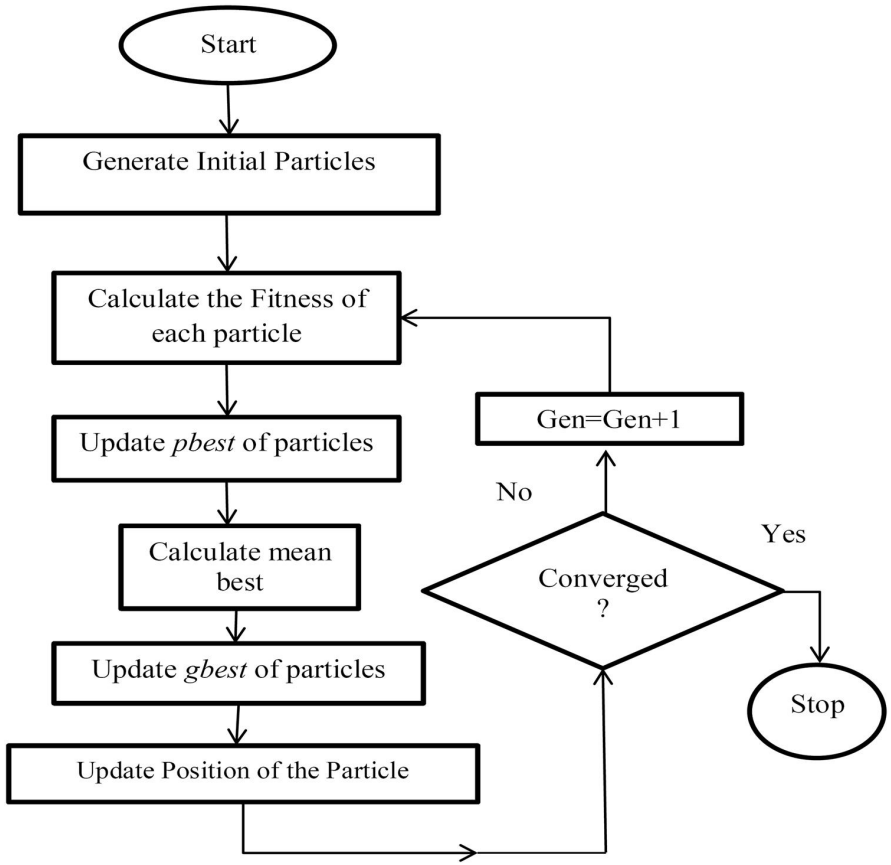
Particle Swarm

Particle swarm is similar to a genetic algorithm in that it creates a population, or in this case a swarm of possible solutions at each iteration. Each solution, or particle in the swarm has a direction and velocity.



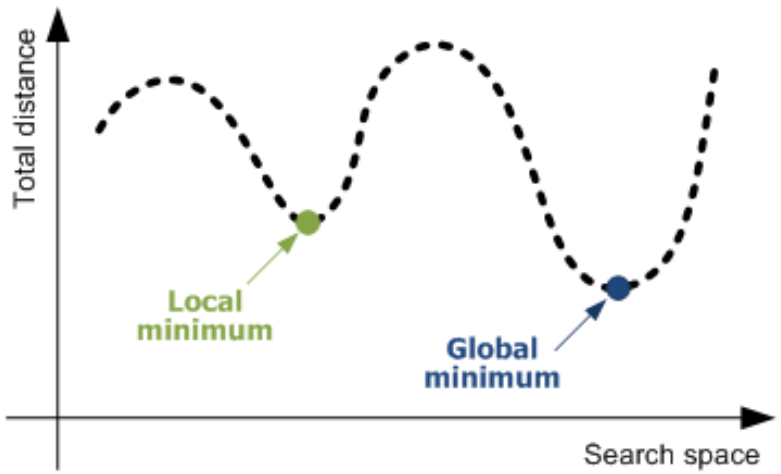
Particle Swarm

At each iteration, the movement of the particle is determined by a mixture of the direction it is currently moving, the direction of the best point it has found in the past, and the direction of the best point the whole swarm has discovered.



Simulated Annealing

Annealing is a process in which metal or glass is heated, and then allowed to slowly cool at a controlled rate. Annealing changes the properties of a metal, making it more ductile and workable. Simulated annealing uses the idea of slowly cooling metal in an optimization algorithm. An initial guess is taken. Then another solution is randomly guessed near the previous solution. This new solution is evaluated to see if it is better than the previous solution.



Simulated Annealing

The figure below is the algorithm of Simulated Annealing, in the line 13, Why do we need to accept it with a certain probability when we are far from the optimal solution?

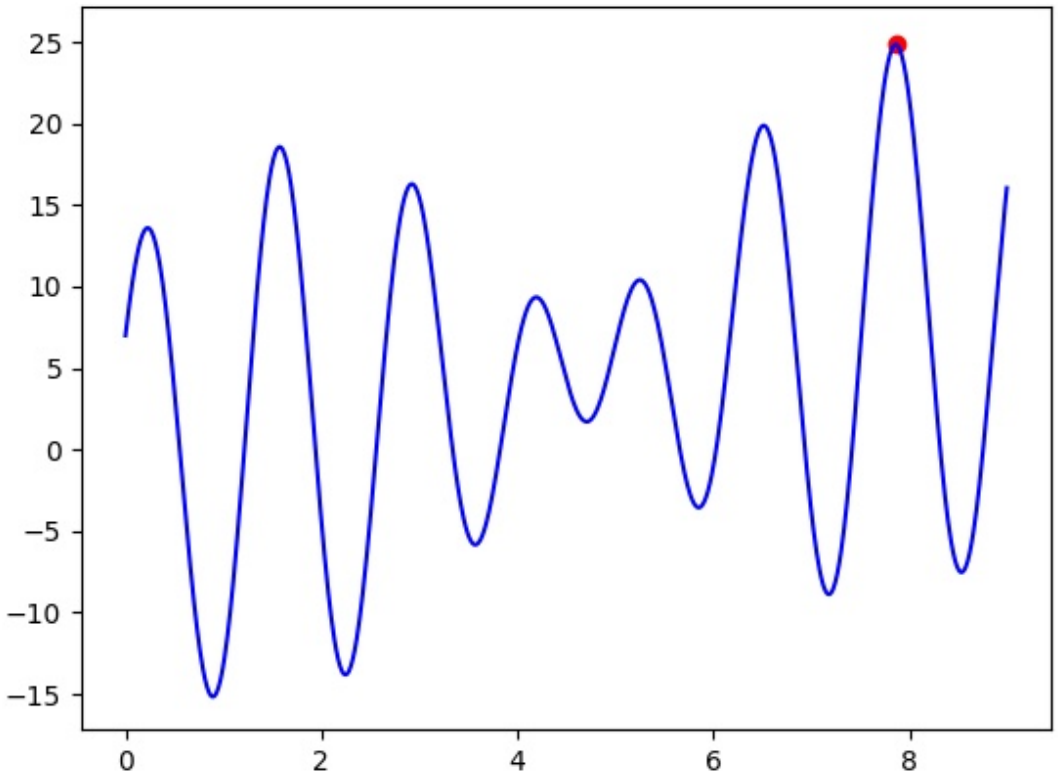
Data: Cooling ratio r and length L
Result: approximate solution S

```
1  Initialize solution S;
2  Initialize temperature T > 0;
3  while not yet frozen do
4  |  for i ← 1 to L do
5  |  |  Pick a random neighbor S' of S;
6  |  |  Δ ← (cost(S') − cost(S));
7  |  |  if Δ ⩽ 0 // downhill move
8  |  |  then
9  |  |  |  S ← S';
10 |  |  end
11 |  |  if Δ ≥ 0 // uphill move
12 |  |  then
13 |  |  |  S ← S' with probability e^{−Δ/T};
14 |  |  end
15 |  end
16 |  T ← rT (reduce temperature);
17 end
```

Simulated Annealing

equation

$$y = x + 10sin(5x) + 7cos(4x) \tag{1}$$



Nelder–Mead method

We are trying to minimize the function $f(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^n$. Our current test points are $\mathbf{x}_1, \ldots, \mathbf{x}_{n+1}$.

1. Order according to the values at the vertices:

$$f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \leq \cdots \leq f(\mathbf{x}_{n+1}).$$

Check whether method should stop. We can set the maximum number of iteration. And Nelder and Mead used the sample standard deviation of the function values of the current simplex. If these fall below some tolerance, then the cycle is stopped and the lowest point in the simplex returned as a proposed optimum. Sometimes inappropriately called "convergence". Sometimes inappropriately called "convergence".

2. Calculate $\mathbf{x}_o$, the centroid of all points except $\mathbf{x}_{n+1}$.

3. Reflection

Compute reflected point $\mathbf{x}_r = \mathbf{x}_o + \alpha(\mathbf{x}_o - \mathbf{x}_{n+1})$ with $\alpha > 0$.

If the reflected point is better than the second worst, but not better than the best, i.e. $f(\mathbf{x}_1) \leq f(\mathbf{x}_r) < f(\mathbf{x}_n)$,

then obtain a new simplex by replacing the worst point $\mathbf{x}_{n+1}$ with the reflected point $\mathbf{x}_r$, and go to step 1.

Nelder–Mead method

4. Expansion

If the reflected point is the best point so far, $f(\mathbf{x}_r) < f(\mathbf{x}_1)$,

then compute the expanded point $\mathbf{x}_e = \mathbf{x}_o + \gamma(\mathbf{x}_r - \mathbf{x}_o)$ with $\gamma > 1$.

If the expanded point is better than the reflected point, $f(\mathbf{x}_e) < f(\mathbf{x}_r)$,

then obtain a new simplex by replacing the worst point $\mathbf{x}_{n+1}$ with the expanded point $\mathbf{x}_e$ and go to step 1;

else obtain a new simplex by replacing the worst point $\mathbf{x}_{n+1}$ with the reflected point $\mathbf{x}_r$ and go to step 1.

5. Contraction

Here it is certain that $f(\mathbf{x}_r) \geq f(\mathbf{x}_n)$. (Note that $\mathbf{x}_n$ is second or "next" to highest.)

Compute contracted point $\mathbf{x}_c = \mathbf{x}_o + \rho(\mathbf{x}_{n+1} - \mathbf{x}_o)$ with $0 < \rho \leq 0.5$.

If the contracted point is better than the worst point, i.e. $f(\mathbf{x}_c) < f(\mathbf{x}_{n+1})$,

then obtain a new simplex by replacing the worst point $\mathbf{x}_{n+1}$ with the contracted point $\mathbf{x}_c$ and go to step 1;

6. Shrink

Replace all points except the best ($\mathbf{x}_1$) with

$\mathbf{x}_i = \mathbf{x}_1 + \sigma(\mathbf{x}_i - \mathbf{x}_1)$ and go to step 1.

Note: α, γ, ρ and σ are respectively the reflection, expansion, contraction and shrink coefficients. Standard values are $\alpha = 1$, $\gamma = 2$, $\rho = 1/2$ and $\sigma = 1/2$.

Conclusion

In summary, a large variety of gradient free optimization methods exist, each with their own strengths and weaknesses. Some of the most commonly used methods are genetic algorithms, particle swarm, simulated annealing, and the Nelder-Mead simplex algorithm.

Conclusion

In general, gradient free methods are easier to implement, and have the advantage of not requiring derivatives. This means that they can be applied to problems that are discrete, discontinuous, or otherwise non-differentiable. Many of the algorithms also do a good job of searching for a global solution rather than a local solution.

Conclusion

On the downside, gradient free algorithms tend to be very slow for large problems. Many of the algorithms are also stochastic, meaning that they are based on chance, and will not always find the same solution. Finally, there is no guarantee that these algorithms will return an optimal solution, meaning that while the solution found might be better that what you started with, we would not know if it is the best solution possible.

The End
Thank You!
Correct me if I'm wrong!

Author: Zhao Chi