Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Российский химико-технологический университет имени Д.И. Менделеева»
Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 9

Выполнил студент группы .....КС-30..... Лихолат Полина Николаевна
Ссылка на репозиторий: .....https://github.com/MUCTR-IKT-CPP/Likholat_algorithms.git

Приняли: .....Пысин Максим Дмитриевич
.....Краснов Дмитрий Олегович
.....Лобанов Алексей Владимирович
.....Крашенинников Роман Сергеевич

Дата сдачи: 16.04.2023

Оглавление

Описание задачи.

Реализовать алгоритм отжига для поиска глобального оптимума (минимума) произвольной функции. В качестве примера взять функцию F(x) = x ^ 2 + 10 - 10 * cos (2 * pi * x) Сам алгоритм выглядит следующим образом:

1. Задать начальное значение (можно выбирать случайно)
2. Изменить значение температуры при помощи заданной функции T(k), где k это номер итерации, получив температуру T[k]
3. Сгенерировать новую точку x[k+1], с которой будет сравниваться текущий вариант(возможна случайная генерация, или использование какой либо функции от температуры)
4. Вычислить значение искомой функции F(x) в точке x[k+1] и вычислить разницу между F(x[k + 1]) - F(x[k]) = dF
5. Проверка решения на вероятность принятий: { 1 при dF < 0 P (x[k] , x[k+1]) = { { exp(-dF / T[k])
6. Проверяем критерий завершения, критерием является некоторая температура окончания.
7. Воспользуемся вариантом быстрого отжига: T(k) = T[0] / k A(x) = x + T * C(0,1), где C это случайно число сгенерированное при помощи распределения коши

После реализации, требуется построить график зависимости времени нахождения решения от значения минимальной температуры, при этом ось x пусть будет инвертированной температурой, полученный график проанализировать.

Описание метода/модели.

Алгори́тм имита́ции о́тжига (англ. *Simulated annealing*) — общий алгоритмический метод решения задачи глобальной оптимизации, особенно дискретной и комбинаторной оптимизации. Один из примеров методов Монте-Карло.

Алгоритм основывается на имитации физического процесса, который происходит при кристаллизации вещества, в том числе при отжиге металлов. Предполагается, что атомы вещества уже почти выстроены в кристаллическую решётку, но ещё допустимы переходы отдельных атомов из одной ячейки в другую. Активность атомов тем больше, чем выше температура, которую постепенно понижают, что приводит к тому, что вероятность переходов в состояния с большей энергией уменьшается. Устойчивая кристаллическая решётка соответствует минимуму энергии атомов, поэтому атом либо переходит в состояние с меньшим уровнем энергии, либо остаётся на месте. (Этот алгоритм также называется алгоритмом Н. Метрополиса, по имени его автора).

Моделирование похожего процесса используется для решения задачи глобальной оптимизации, состоящей в нахождении такой точки или множества точек, на которых достигается минимум некоторой целевой функции F(x) ("энергия системы"), где $x \in X$ (x — "состояние системы", X — множество всех состояний).

Алгоритм поиска минимума методом имитации отжига предполагает свободное задание:

$x_0 \in X$ — начального состояния системы;

оператора $A(x, i) : X \times \mathbb{N} \to X$, случайно генерирующего новое состояние системы после i-ого шага с учётом текущего состояния x (этот оператор, с одной стороны, должен обеспечивать достаточно свободное случайное блуждание по пространству X, а с другой — работать в некоторой степени целенаправленно, обеспечивая быстроту поиска);

$T_i > 0$— убывающей к нулю положительной последовательности, которая задаёт аналог понижающейся температуры в кристалле. Скорость остывания (закон убывания) также может задаваться (и варьироваться) произвольно, что придаёт алгоритму значительной гибкости.

Алгоритм генерирует процесс случайного блуждания по пространству состояний X. Решение ищется последовательным вычислением точек $x_0, x_1, x_2, \ldots$, пространства X; каждая точка, начиная с x_1, «претендует» на то, чтобы лучше предыдущих приближать решение. На каждом шаге алгоритм (который описан ниже) вычисляет новую точку и понижает значение величины (изначально положительной), понимаемой как «температура».

Последовательность этих точек (состояний) получается следующим образом. К точке x_i применяется оператор A, в результате чего получается точка-кандидат $x_i^* = A(x_i, i)$, для которой вычисляется соответствующее изменение "энергии" $\Delta F_i = F(x_i^*) - F(x_i)$. Если энергия понижается ($\Delta F_i \leq 0$), осуществляется переход системы в новое состояние: $x_{i+1} = x_i^*$. Если энергия

повышается ($\Delta F_i > 0$), переход в новое состояние может осуществиться лишь с некоторой вероятностью, зависящей от величины повышения энергии и текущей температуры, в соответствии с законом распределения Гиббса:

$$P(x_i^* \to x_{i+1}|x_i) = \exp(-\frac{\Delta F_i}{T_i})$$

Если переход не произошёл, состояние системы остаётся прежним: $x_{i+1} = x_i$. Алгоритм останавливается по достижении точки, которая оказывается при температуре ноль.

Алгоритм имитации отжига похож на градиентный спуск, но за счёт случайности выбора промежуточной точки и возможности выбираться из локальных минимумов должен реже застревать в локальных, но не глобальных минимумах, чем градиентный спуск. Алгоритм имитации отжига не гарантирует нахождения минимума функции, однако при правильной настройке генерации случайной точки в пространстве X, как правило, происходит улучшение начального приближения.

Выполнение задачи.

Алгоритм отжига работает следующим образом:

1. Задаются начальное значение x, начальная температура T0, минимальная температура Tmin, и максимальное количество итераций kmax.
2. Запускается цикл, в котором на каждой итерации генерируется новая точка x_new, которая получается путем прибавления случайного значения, сгенерированного из нормального распределения с центром в 0 и стандартным отклонением T.
3. Если значение функции в новой точке меньше, чем в текущей, то новая точка принимается.
4. Если значение функции в новой точке больше, чем в текущей, то новая точка принимается с вероятностью, равной exp(-dF/T), где dF - разница между значениями функции в новой и текущей точках, а T - текущая температура.
5. Температура понижается по формуле T = T0 / (1 + alpha*k), где k - номер итерации.
6. Цикл продолжается до тех пор, пока текущая температура T больше минимальной температуры Tmin или количество итераций не превышает максимальное значение kmax.
7. Возвращается точка, в которой достигается минимум функции, и ее значение.

```
import math
import random
import time
import numpy as np
import matplotlib.pyplot as plt

# Определяем функцию, которую нужно минимизировать
def F(x):
    return x**2 + 10 - 10*math.cos(2*math.pi*x)

# Реализуем алгоритм отжига
def annealing(T0, Tmin, kmax):
    """Функция, реализующая алгоритм отжига
    @param T0: Начальная температура
```

```
    @param Tmin: Конечная температура
    @param kmax: Максимальное количество итераций
    @return: Возвращает значение x, значение функции F(x), и списки значений времени работы алгоритма и температуры
    """
    x = random.uniform(-5.0, 5.0) # Начальное значение x
    t = time.time() # Засекаем время
    t_vals = [] # Список для хранения значений времени
    T_vals = [] # Список для хранения значений температуры
    x_vals = [] # Список для хранения значений x
    k = 0 # Номер итерации
    T = T0 # Начальная температура
    while T > Tmin:
        k += 1
        x_new = x + T * np.random.standard_cauchy() # Генерируем новую точку
        dF = F(x_new) - F(x) # Вычисляем разницу значений функции
        if dF < 0 or random.random() < math.exp(-dF/T):
            x = x_new # Принимаем новую точку
        #T = T0 / (k**alpha) # Изменяем температуру
        #T = T0 / (1 + alpha*k)
        T = T0 / k # из задания
        #T = T * alpha
        # Сохраняем текущие значения x, t, T
        x_vals.append(x)
        t_vals.append(time.time() - t)
        T_vals.append(T)
        if k == kmax:
            break
    return x, F(x), t_vals, T_vals

# Задаем параметры алгоритма
T0 = 100 # Начальная температура
Tmin = 0.001 # Минимальная температура
kmax = 10000 # Максимальное число итераций

# Запускаем алгоритм отжига
x_opt, F_opt, t_vals, T_vals = annealing(T0, Tmin, kmax)

# Выводим результаты
print("Параметр X:", x_opt)
print("Значение функции:", F_opt)
```

```
Параметр X: -0.0006038057804218164
Значение функции: 7.232998294526283e-05
```

На рис.1 представлен график зависимости времени нахождения решения от значения минимальной температуры. По нему мы видим, что увеличение значения минимальной температуры приводит к увеличению времени работы алгоритма.

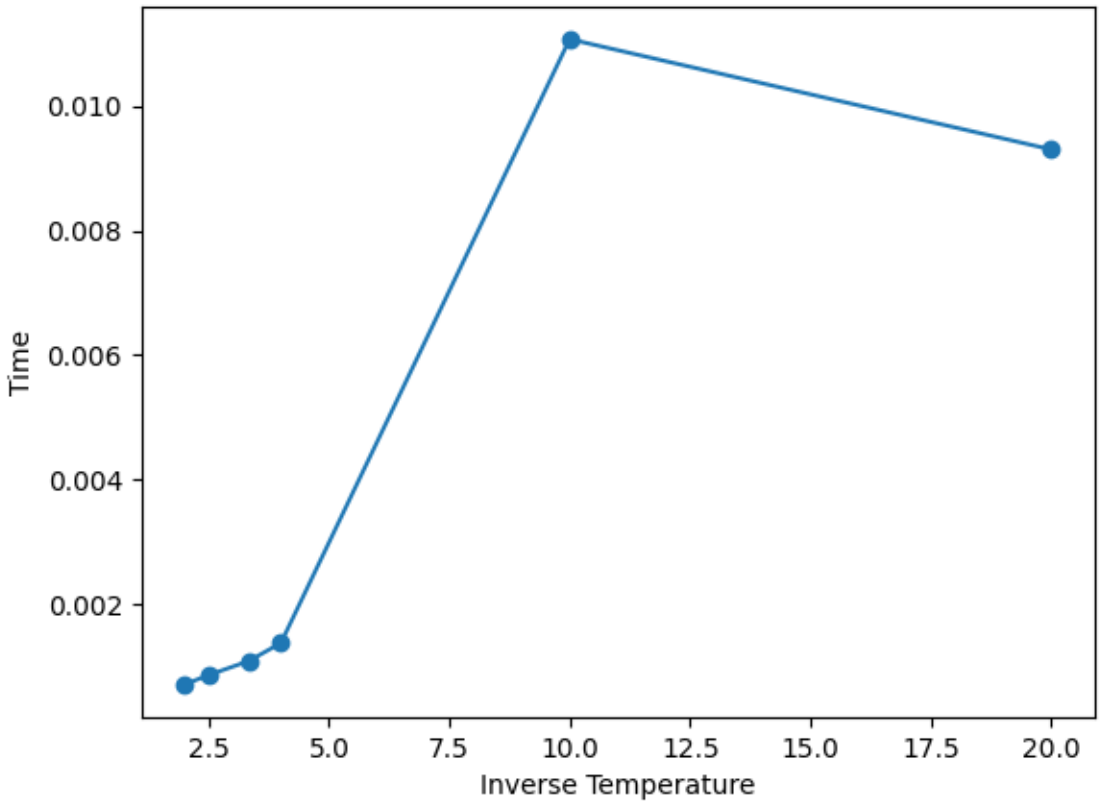

Рис. 1. График зависимости времени от значения минимальной температуры

На рис.2 представлен график данной в задании функции. По нему мы видим, что глобальный минимум функции находится в точке 0, а значит, полученное решение достаточно точно. Так же мы видим, что функция имеет множество локальных минимумов, но с использованием метода отжига мы смогли найти глобальный минимум.

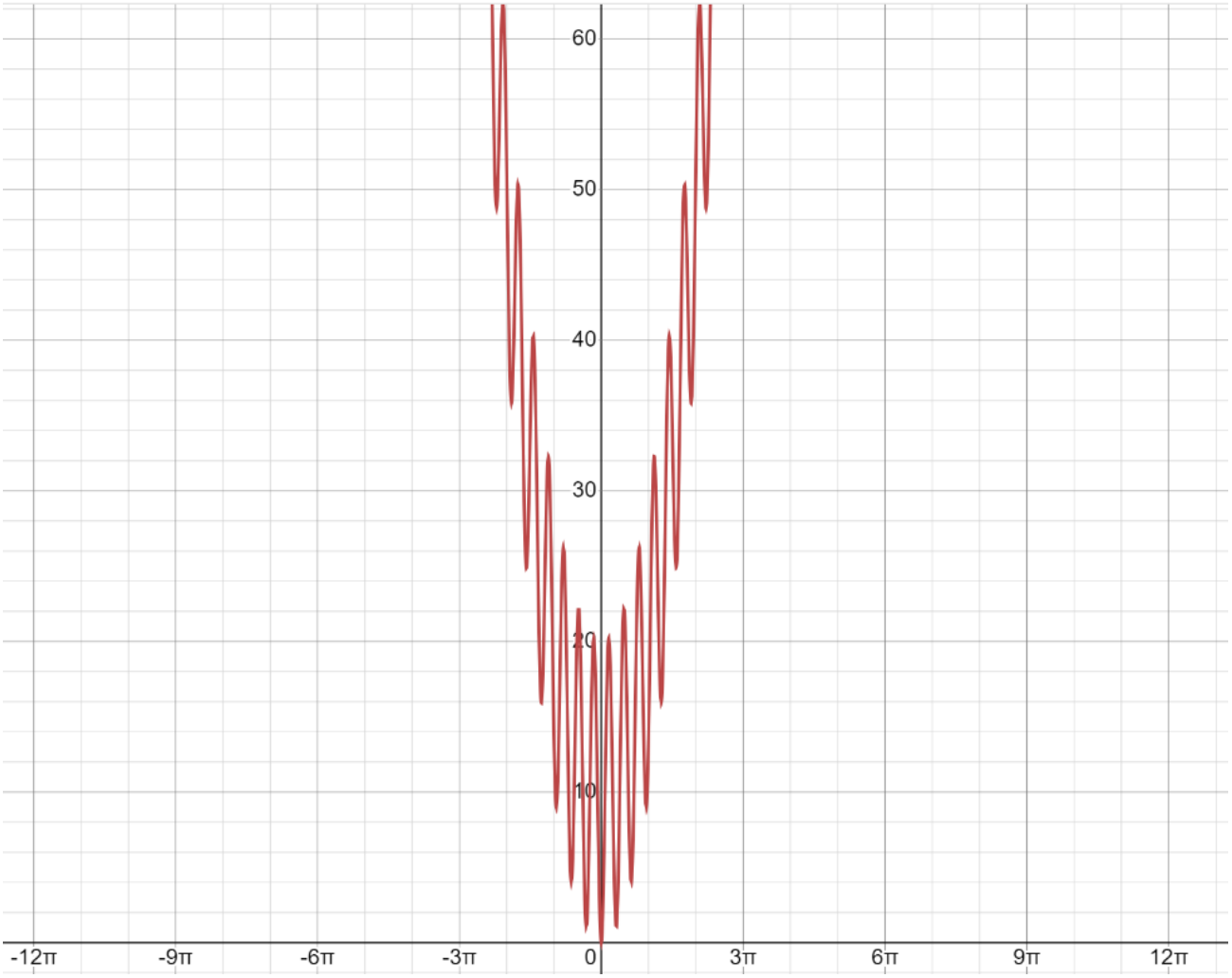

Рис. 2. График заданной функции

Заключение.

Как и любой стохастический метод, имитация отжига не гарантирует нам наилучшего решения. Однако метод дает хороший шанс выбраться из "локального минимума". Еще одним несомненным достоинством данного метода является простота реализации.

Настройка параметров метода отжига играет важную роль в его эффективности. Оптимальные значения параметров могут зависеть от конкретной задачи и функции, которую необходимо оптимизировать.