Решение задач невыпуклой оптимизации с использованием мультистартов

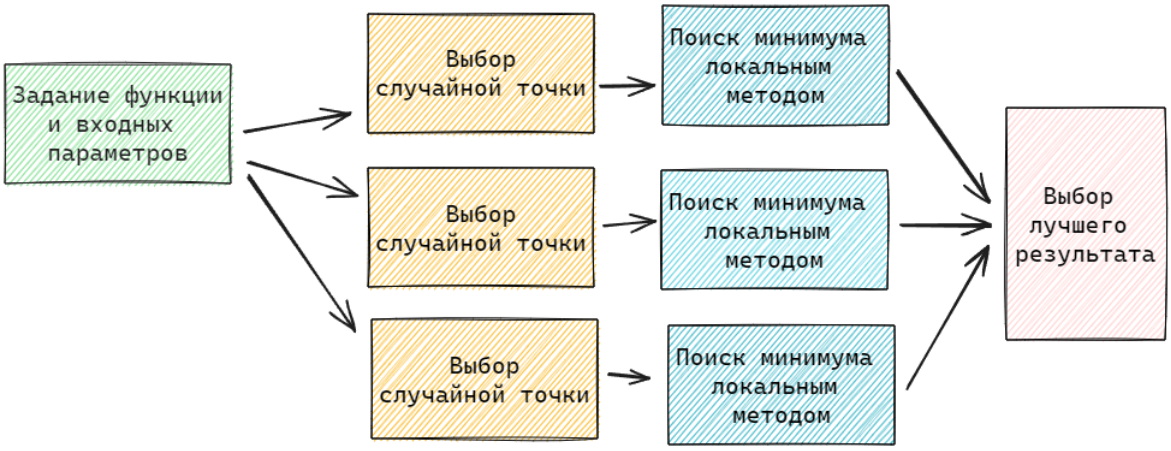
Задача

Известно, что задача оптимизации за пределами класса выпуклых функций является NP-трудной. Существуют разные подходы к решению этой задачи и многие из них основаны на использовании рандомизированных алгоритмов. В данной лабораторной работе реализован и протестирован один из таких подходов.

Существет масса быстрых алгоритмов поисков экстремумов выпуклых функций. Но применение таких методов к невыпуклым функция приводит к тому, что метод сходится лишь к локальному экстремуму. Наша цель найти глобальный экстремум.

Метод мультистартовзаключается случайном выборе стартовых точек в области поиска экстремума и применении методов локального поиска из этих начальных приближений. В результате мы имеем набор найденных нами локальных минимумов или максимумов и дальше мы должны выбрать самый экстремальный из них. Разумеется этот метод не даёт гарантии, что мы нашли лучший результат, но так задача NP-трудная, мы понимаем что полиномиальных точных алгоритмов для неё быть не может.

Схема поиска алгоритма нахождения результата:



Программная реализация выполнена на языке python с использование технологии MPI. В качестве метода локального поиска использовался алгоритм ralgb5 - это модификация субградиентного спуска с растяжением пространства по направлению разности двух последовательных субградиентов. Реализация доступна в репозитории:

https://github.com/ZvyaginMA/MpiMultistart

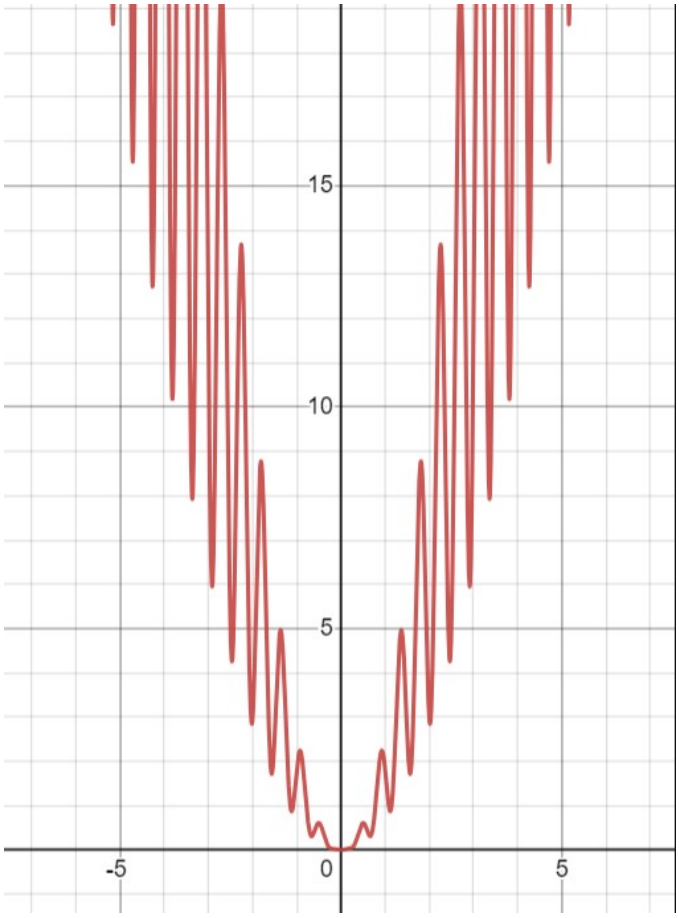
Расчеты проводились на компьютере с процессором AMD Ryzen 5 5500U with Radeon Graphics 2.10 GHz, 16.0 ГБ, 6 ядер 12 логических процессоров.

Пример 1

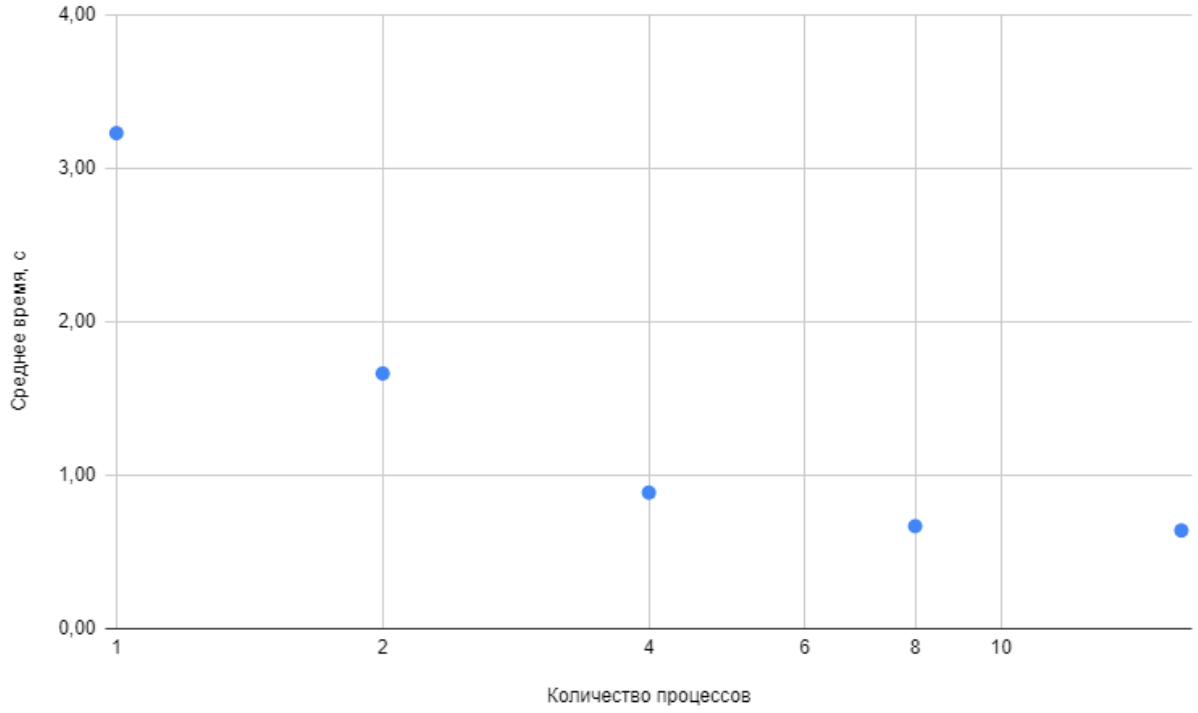
В этом примере мы будем искать глобальный минимум функции:

$$f(x) = x^2 \cdot \cos(14x) + 1.7x^2$$

График этой функции:



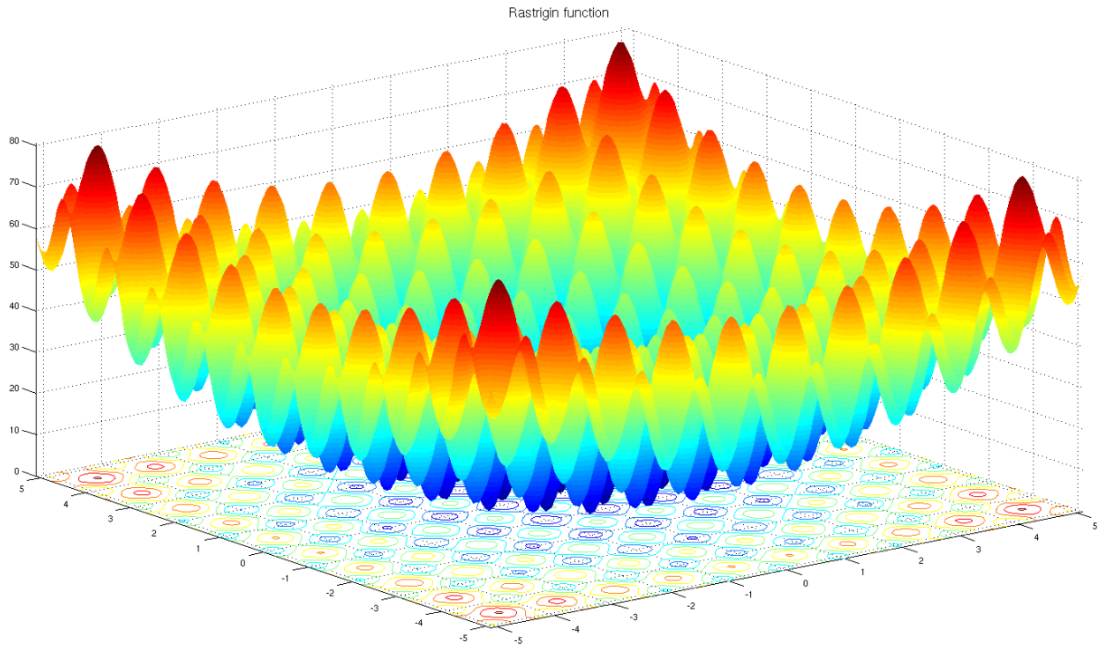
Зависимость времени расчетов от количества процессов



Пример 2 Функция Растригина

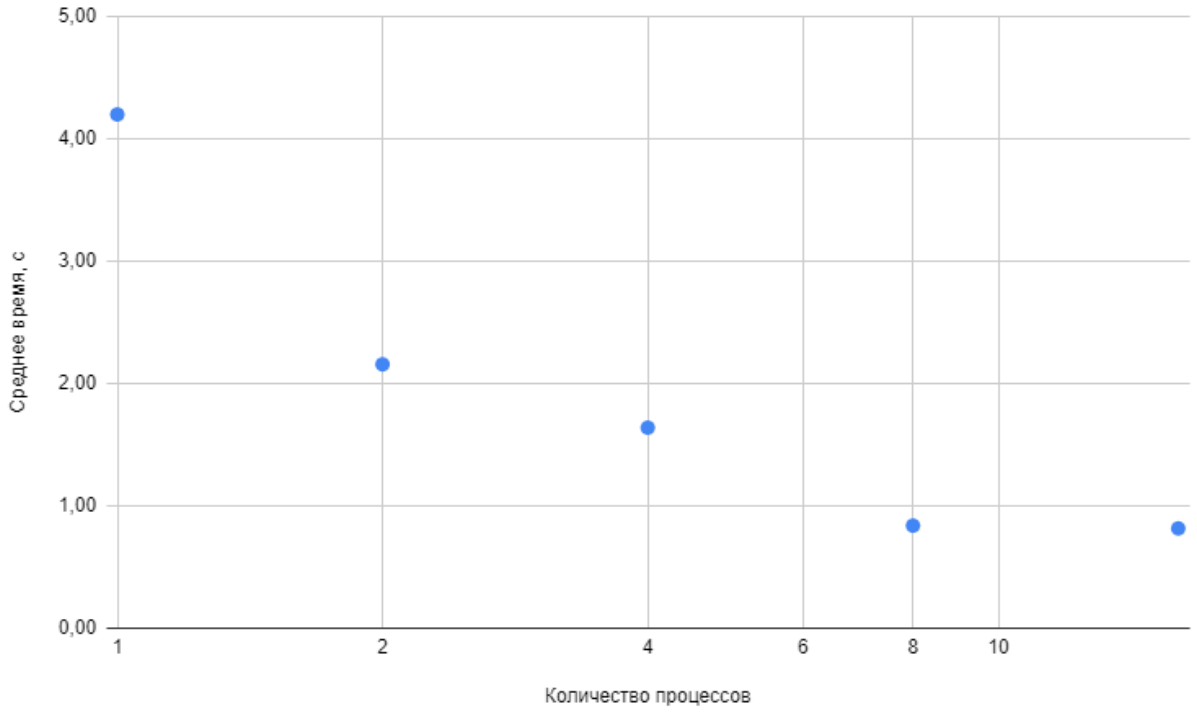
$$f(x) = 10 \cdot n + \sum_{i=1}^{n}(x_i^2 - 10 \cdot \cos(2\pi x_i))$$

При n = 2 график функции представляет собой:

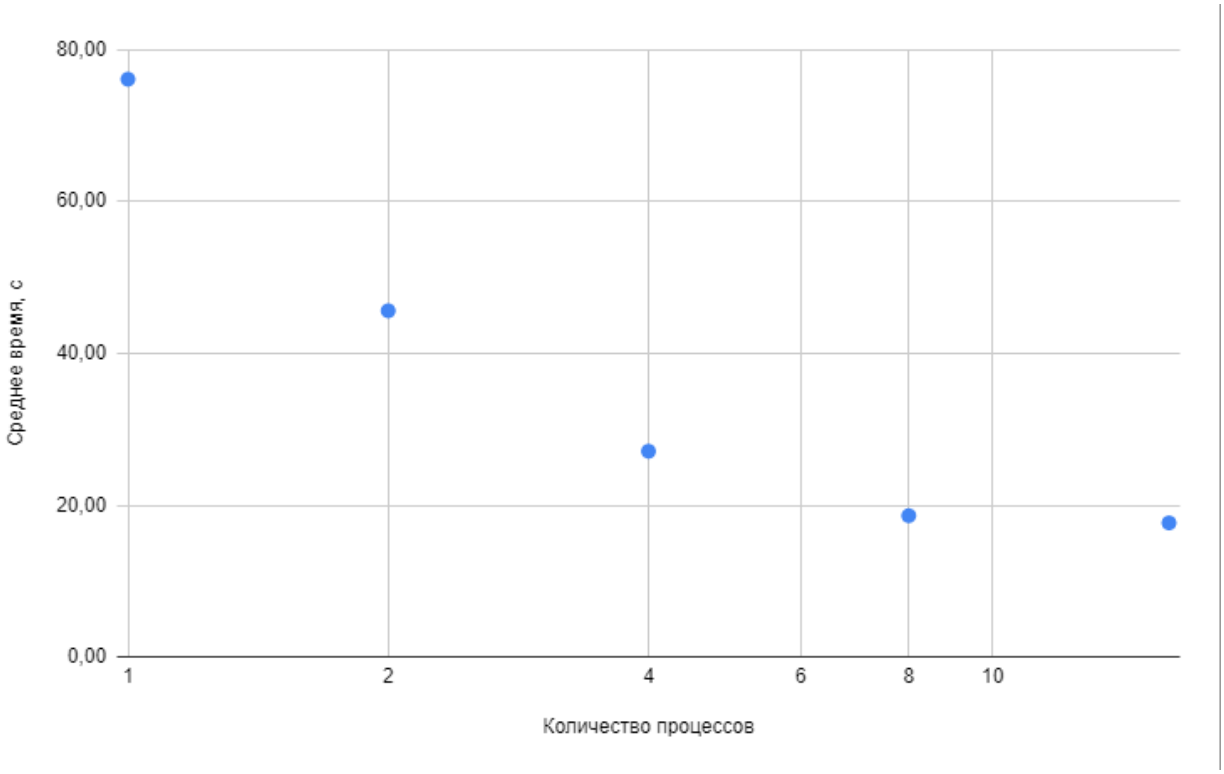


На данных графиках представлена зависимость времени от количества процессов.

n = 2, 1000 итераций.



n = 4, 10000 итераций.



Как видим начиная с 8 процессов быстродействие программы останавливается, это связано с ограниченностью ресурсов компьютера.