Тема доклада:
"Поиск оптимальных упаковок кругов при помощи алгоритмов оптимизации пакета PyTorch"

Выполнил:
Студент 2 курса факультета математики и компьютерных наук
группа 2ПМ, Никитенко В.К.

Научный руководитель:
Кандидат технических наук, доцент Воронов В.А.

Майкоп
2021

Введение

Курсовая работа посвящена применению современных методов оптимизации, в частности стохастического градиентного спуска, к задаче поиска оптимальной упаковки кругов.

Цель работы – применить алгоритм стохастического градиентного спуска к задаче об оптимальной упаковке кругов.

Задачи:

- Научиться работать с фреймворком PyTorch на базовом уровне;
- Улучшить скорость работы алгоритма оптимизации за счет выбора параметров;
- Найти процент оптимальных конфигураций, вычисляемых алгоритмом оптимизации;
- Максимизировать получаемые значения;
- Собрать и проанализировать получаемые данные.

Фреймворк PyTorch

PyTorch — современная библиотека глубокого обучения, разработанная и развивающаяся компанией Facebook. Его разработка началась в 2012 году, но открытым и доступным широкой публике PyTorch стал лишь в 2017 году. С этого момента фреймворк очень быстро набирает популярность и привлекает внимание всё большего числа исследователей.

Тензорные вычисления — основа PyTorch, каркас, вокруг которого наращивается вся остальная функциональность.

Постановка задачи

Задачи упаковки — это класс задач оптимизации в математике, в которых пытаются упаковать объекты в контейнеры. Цель упаковки — либо упаковать отдельный контейнер как можно плотнее, либо упаковать все объекты, использовав как можно меньше контейнеров, либо нахождение конфигурации, которая упаковывает один контейнер с максимальной плотностью. При этом объекты не должны пересекаться и объекты не должны пересекать стены контейнера.

Многие из таких задач могут относиться к упаковке предметов в реальной жизни, вопросам складирования и транспортировки.

Градиентный спуск

Градиентный спуск — это эвристический алгоритм, который выбирает случайную точку, рассчитывает направление скорейшего убывания/возрастания функции (пользуясь градиентом функции в данной точке), а затем пошагово рассчитывает новые значения функции, двигаясь в выбранную сторону. Если убывание/возрастание значения функции становится слишком медленным, алгоритм останавливается и говорит, что нашел минимум.

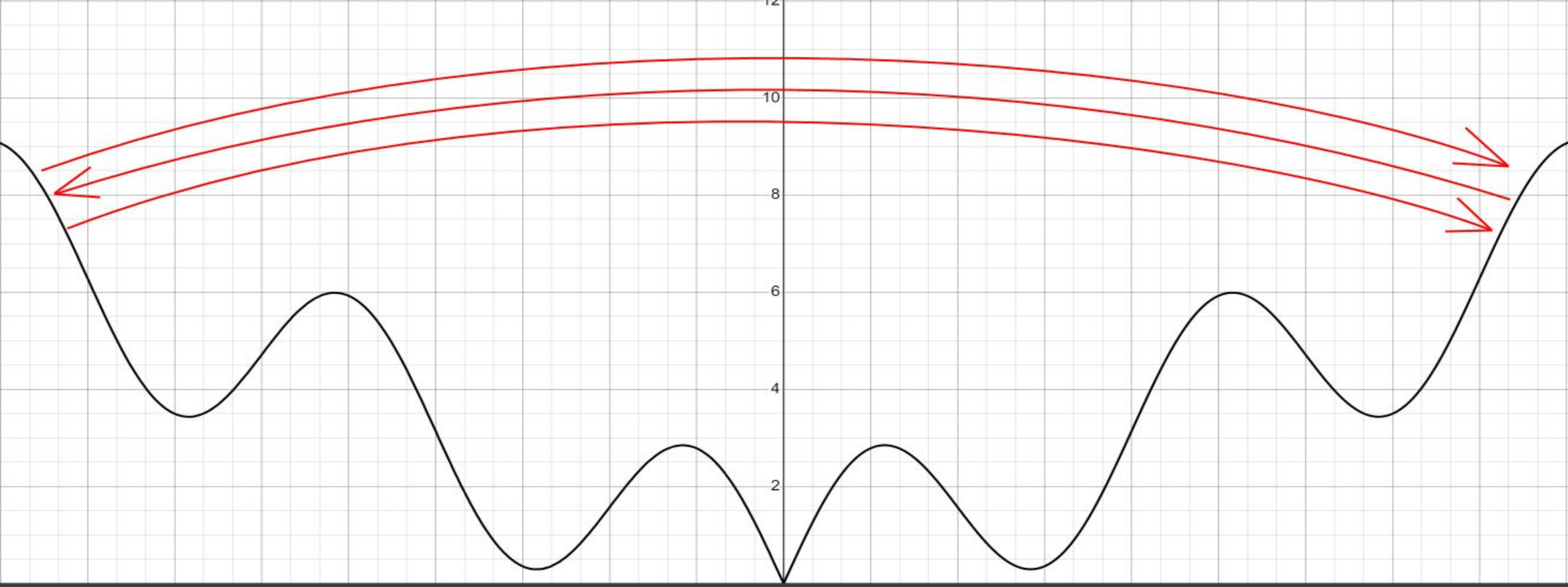
Градиентный спуск выбирает случайную точку, находит направление самого быстрого убывания функции и двигается до ближайшего минимума вдоль этого направления. Если на каком-то этапе разность между старой точкой (до шага) и новой снижается ниже предела, считается, что минимум найден, алгоритм завершен.

Скорость обучения

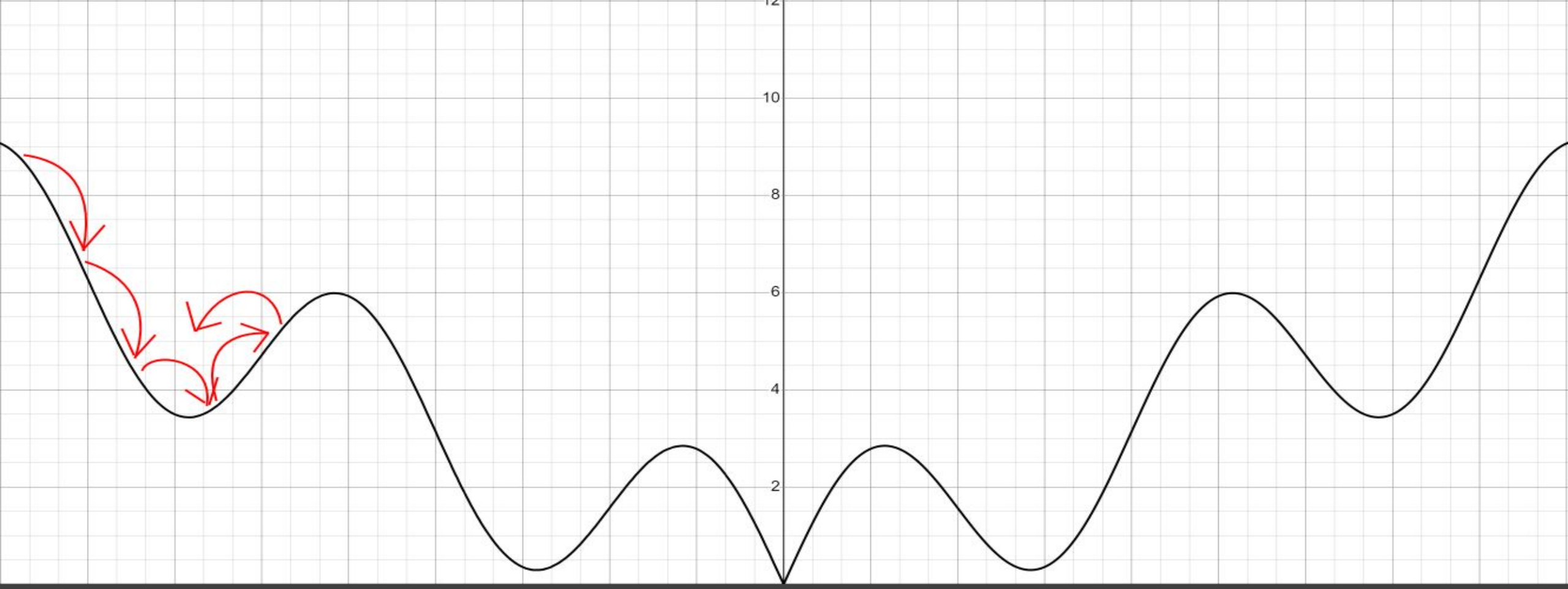
Размер шага алгоритма определяет, насколько мы собираемся двигать точку на графике функции потерь, и этот параметр называется «скоростью обучения».

Скорость обучения стоит воспринимать как ширину шагов. В некоторых случаях бывает так, что слишком широкие шаги вообще не позволяют достичь минимума, и машина бесконечно перешагивает через него, затем градиент «разворачивает» ее обратно, и алгоритм снова перескакивает через минимум.

Слишком маленькая скорость обучения тоже может не дать нужного результата, так как можно попасть в локальный минимум, из которого уже не получится выйти.



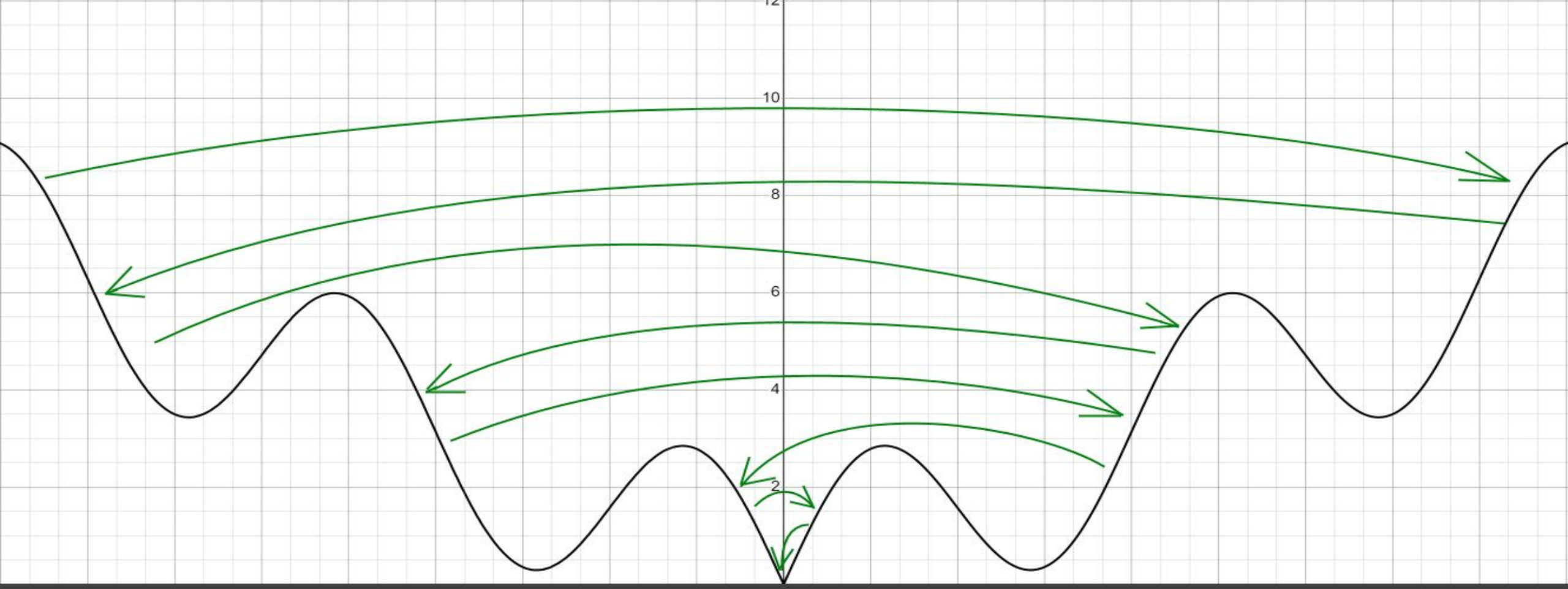
СКОРОСТЬ ОБУЧЕНИЕ СЛИКШОМ ВЕЛИКА



СКОРОСТЬ ОБУЧЕНИЕ СЛИКШОМ МАЛА

Затухающая скорость обучения

Программа начинает работать с большой скоростью обучения, когда разность между старой точкой и новой перестает меняться или меняется слишком незначительно указанное количество шагов (значение patience), скорость обучения понижается. Таким образом, постепенно доходим до минимума функции.



ЗАТУХАЮЩАЯ СКОРОСТЬ ОБУЧЕНИЯ

Значение patience

При затухающей скорости обучения важно знать, в какой момент нужно уменьшить скорость. Для этого используется значение patience.

Чтобы определить, какое значение даст наилучший результат, было решено провести большое количество запусков программы при разных значениях, при этом подсчитывая среднее арифметическое радиусов кругов и записывая всю полученную информацию в документ. Таким образом получилось выяснить примерные лучшие варианты.

```
 20_10: (100, 14.419323831796646, 0.14419323831796646)
 30_10: (100, 14.557886108756065, 0.14557886108756066)
 40_10: (100, 14.594097211956978, 0.14594097211956977)
 50_10: (100, 14.652108877897263, 0.14652108877897263)
  60_10: (100, 14.65576159954071, 0.1465576159954071)
 70_10: (100, 14.666099473834038, 0.14666099473834038)
 80_10: (100, 14.678517773747444, 0.14678517773747443)
 90_10: (100, 14.695189774036407, 0.14695189774036407)
100_10: (100, 14.747388735413551, 0.14747388735413552)
110_10: (100, 14.727175563573837, 0.14727175563573838)
120_10: (100, 14.719042524695396, 0.14719042524695397)
130_10: (100, 14.752179980278015, 0.14752179980278016)
140_10: (100, 14.742900416254997, 0.14742900416254998)
150_10: (100, 14.759741485118866, 0.14759741485118866)
160_10: (100, 14.738642171025276, 0.14738642171025276)
170_10: (100, 14.739636912941933, 0.14739636912941934)
180_10: (100, 14.766204684972763, 0.14766204684972764)
190_10: (100, 14.741886541247368, 0.14741886541247368)
 200_10: (100, 14.74767418205738, 0.1474767418205738)
210_10: (100, 14.759666308760643, 0.14759666308760644)
220_10: (100, 14.758980199694633, 0.14758980199694632)
 230_10: (100, 14.75654511153698, 0.14756545111536978)
240_10: (100, 14.767590835690498, 0.14767590835690497)
250_10: (100, 14.766855612397194, 0.14766855612397195)
260_10: (100, 14.757829666137695, 0.14757829666137695)
270_10: (100, 14.743579804897308, 0.14743579804897308)
280_10: (100, 14.758969902992249, 0.14758969902992247)
290_10: (100, 14.758339539170265, 0.14758339539170265)
300_10: (100, 14.746790930628777, 0.14746790930628775)
```

ПРИМЕР ПОЛУЧЕННЫХ ДАННЫХ

Результаты испытаний

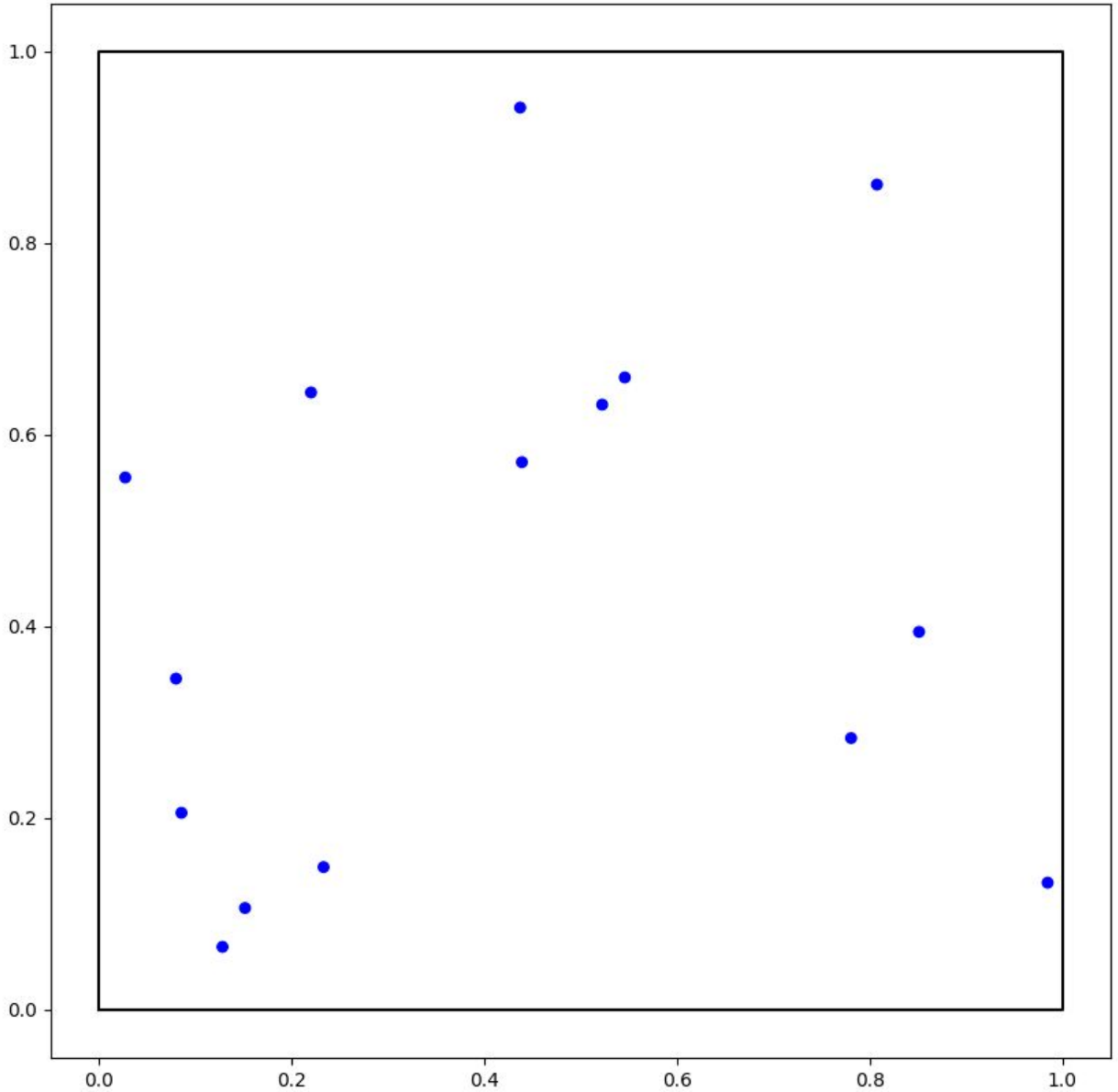
Количество кругов	Лучшее значение patience
10	300
15	320
20	370
25	380
30	390

По полученным результатам можно сказать, что с ростом количества кругов лучшие результаты получаются на больших значениях patience, но разница между ними уменьшается, поэтому для дальнейших вычислений было выбрано среднее из лучших значений – 320.

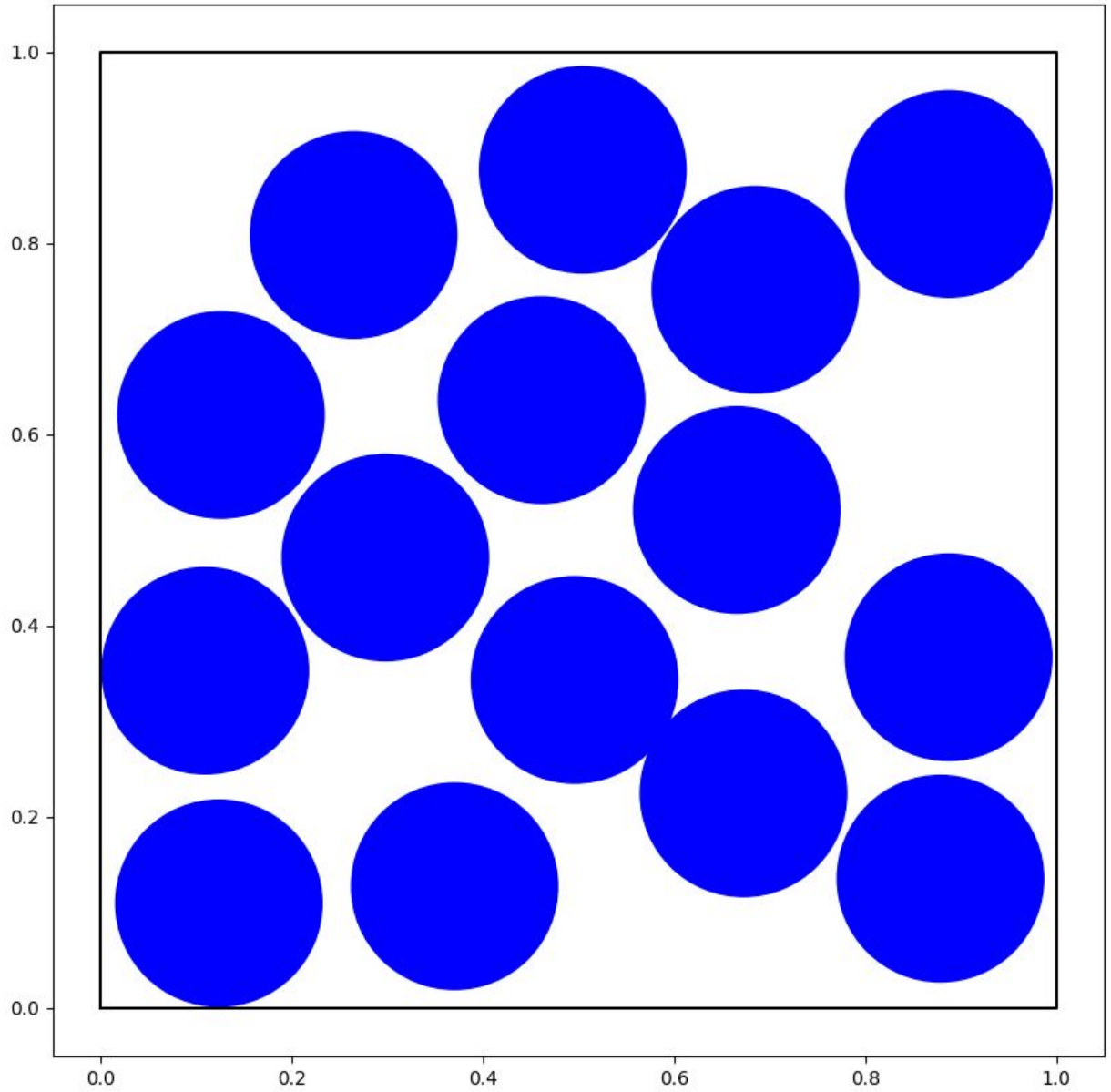
Решение задачи

Построим область (контейнер) по заданным координатам вершин с помощью уравнений прямых. Далее находим попарно расстояние между центрами кругов, делим на 2 и выбираем наименьшее из них. Находим расстояние от центров кругов до сторон квадрата. Выбираем наименьшее значение из двух полученных, оно и будет является новым радиусом кругов. Далее сдвигаем центры кругов и повторяем тот же алгоритм, таким образом на n-ном шаге находим лучшее значение радиуса.

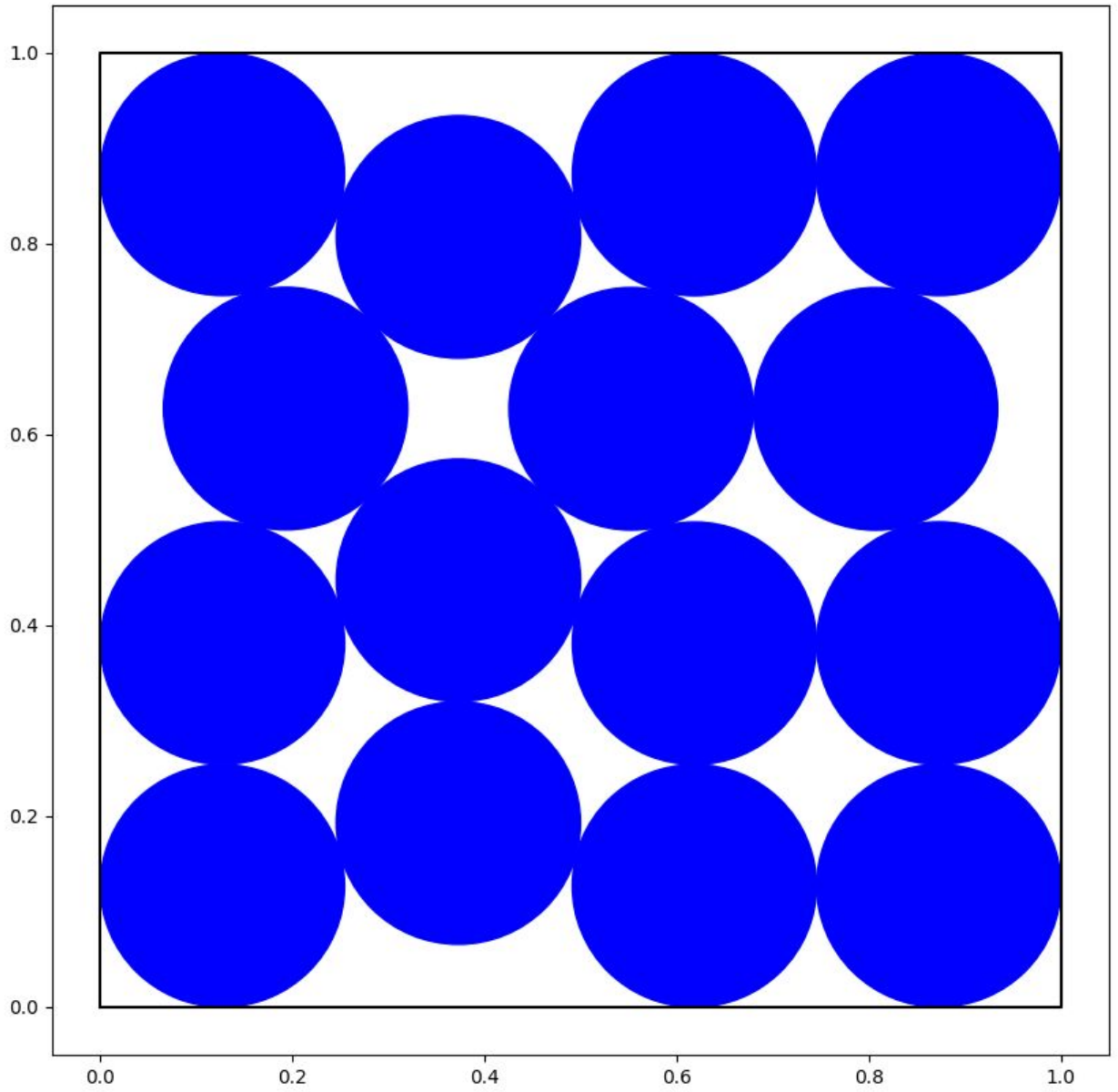
В программе реализуем этот же алгоритм, с использованием тензоров.



ПРИМЕР ВЫПОЛНЕНИЯ



ПРИМЕР ВЫПОЛНЕНИЯ



ПРИМЕР ВЫПОЛНЕНИЯ

Результаты

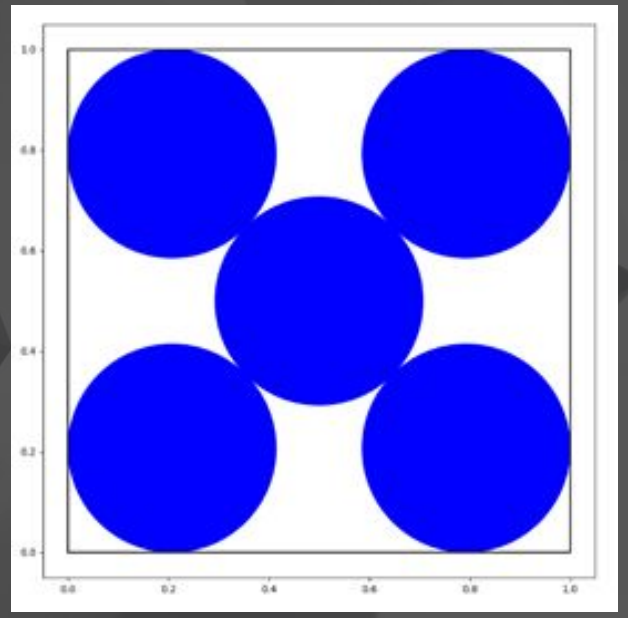
Количество кругов	Количество запусков	Количество рекордных значений	Процент рекордных значений	Лучшее значение	Мировой рекорд
5	5000	0	0%	0.2070484	0. 2071067
10	5000	2	0.04%	0.1481970	0.1482043
15	5000	304	6.08%	0.1271605	0.1271665
20	5000	292	5.84%	0. 1113750	0.1113823
25	5000	178	3.56%	0.0999949	0.1
30	5000	2	0.04%	0. 0916613	0.0916710

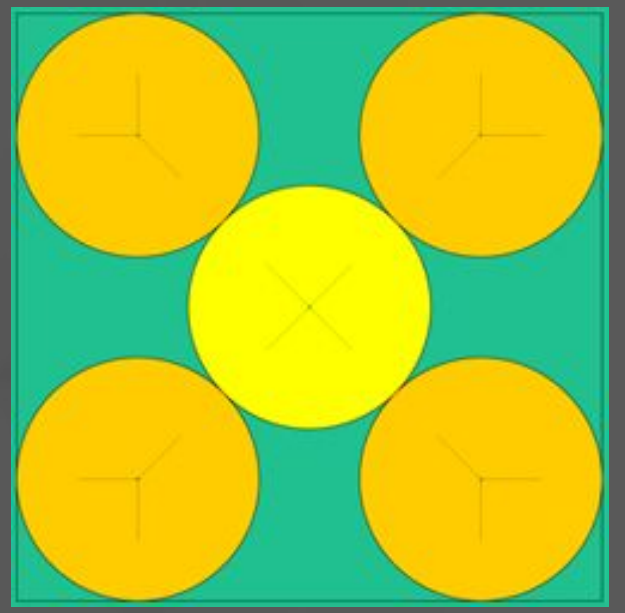
Полученное значение считалось рекордным, если разница мирового рекорда и этого значения была меньше либо равна 10^{-5}.

Все значения в таблице округлены до 10^{-7}.

Структура

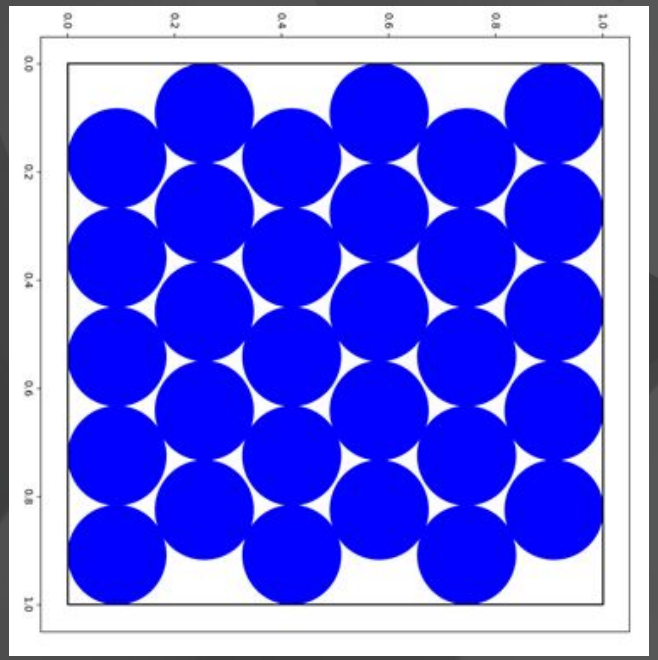
Как можно заметить, при малом количестве кругов рекордных значений практически нет. Вероятно, это связано с тем, что при такой размерности критерии остановки алгоритма оказываются слишком грубыми. По получаемому изображению, например, для 5 кругов, видно, что структура составляется верно, но алгоритм не может найти более точно локальный минимум.

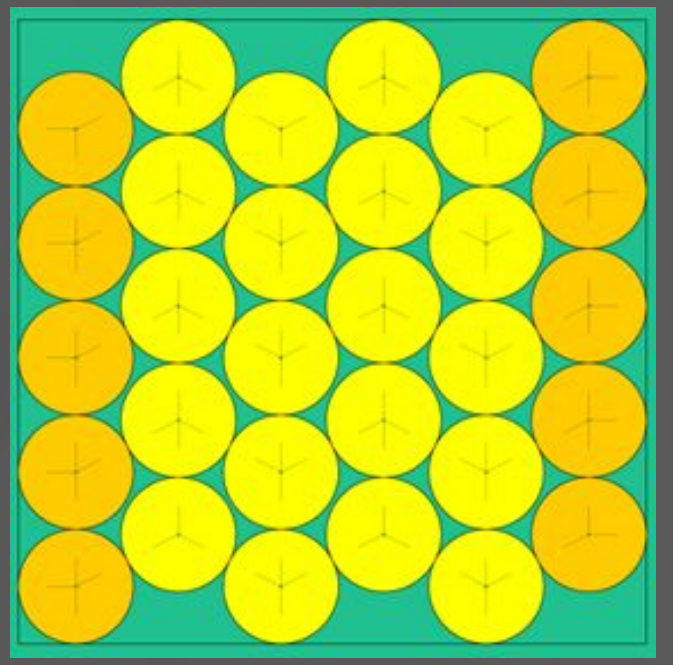




Структура

С ростом количества кругов сильно растет влияние фактора случайности генерации центров, поэтому процент рекордных значений уменьшается. При этом программа не редко находит нужную структуру размещения, но ей не удается получить лучшее значение локального минимума.





Заключение

Как показали исследования, при выбранных параметрах алгоритма и критериях сравнения с рекордом количество рекордных значений быстро уменьшается с ростом количества кругов. Чтобы улучшить результаты нужно более детально подобрать значение patience для каждого набора кругов, а также изменять множитель learning rate для более тонкой настройки.

Из приведенных рисунков видно, что во многих случаях программа правильно находит структуру оптимальной упаковки, но не может вычислить значение с высокой точностью.

Спасибо за внимание!