

# Решение задачи MaxCut при помощи Deep Neural Network

---

## Введение

### Задача MaxCut

MaxCut - задача о поиске максимального разреза в произвольном графе. Задача NP-полнная (к ней может быть сведена любая иная NP-задача за полиномиальное время).

Формальная постановка задачи:

Дано: \$Graph\quad G = (E, V)\$ на \$n = |V|\$ вершинах с \$e = |E|\$ ребрами

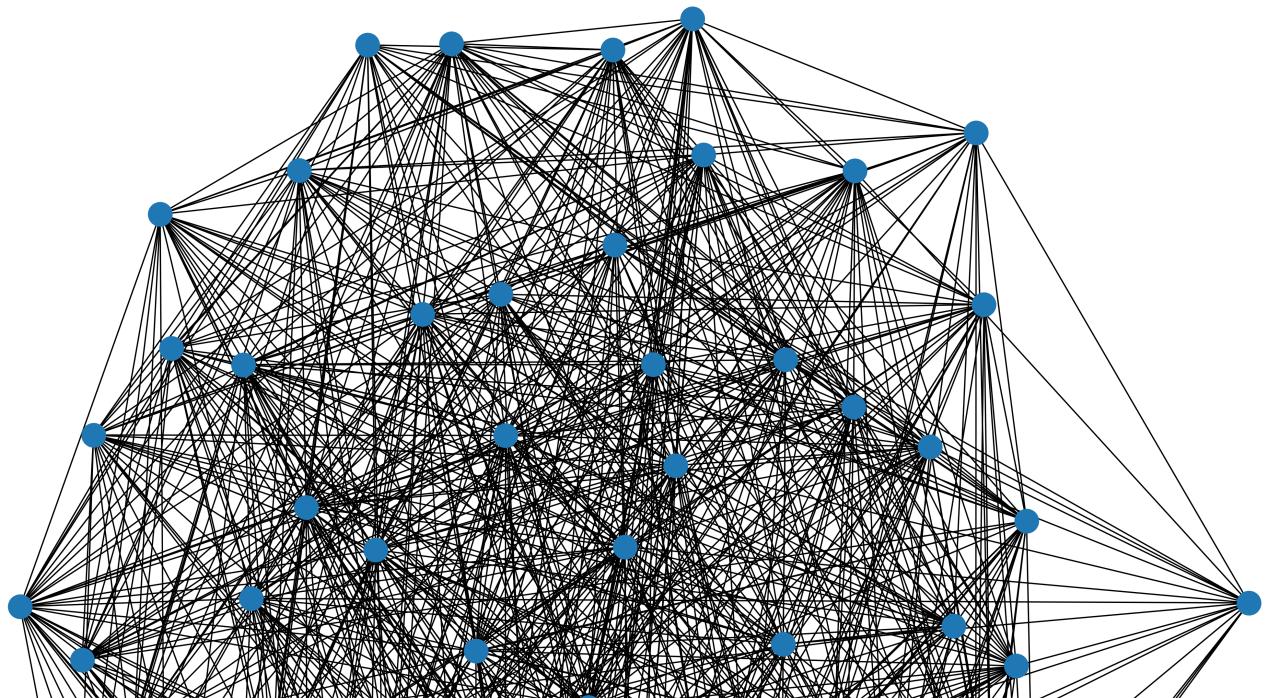
Найти: \$x \in \{+1; -1\}^n\$ : \$\sum\_{\{i,j \in E\}} (1 - x\_i x\_j) \rightarrow \max\$

### Тестовые данные

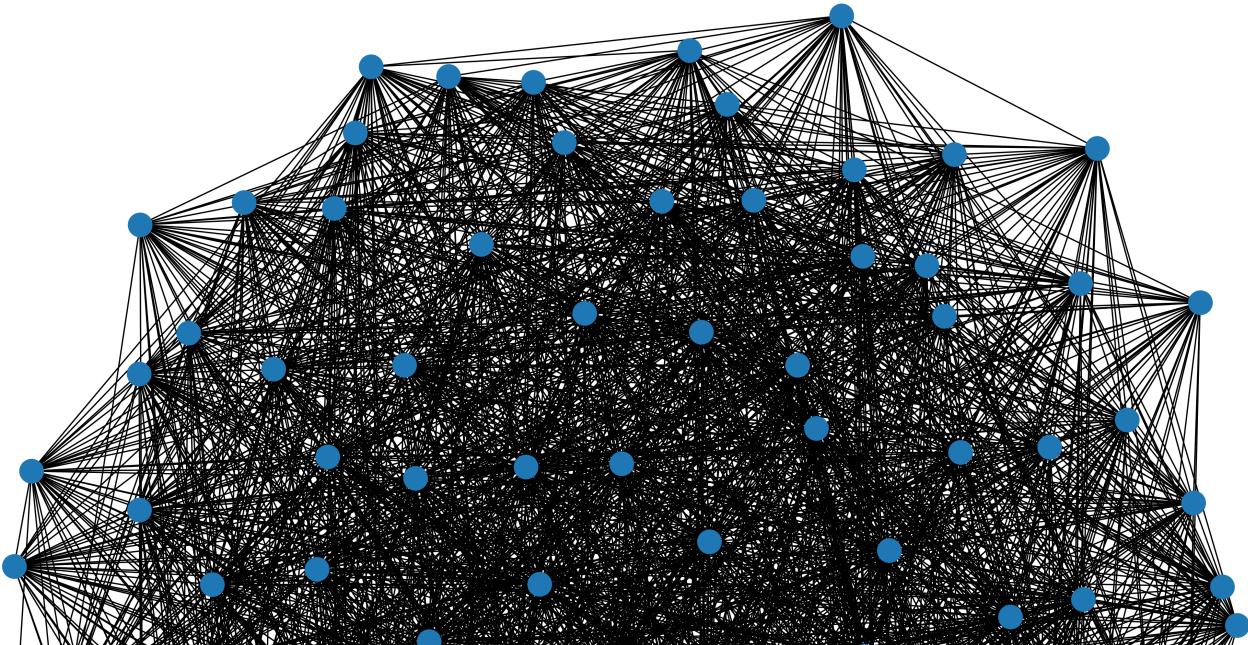
Мы будем рассматривать два графа из [1], которые в терминах оригинальной статьи носят имена:

- g05\_60.0 - граф на 60 вершинах, который имеет 885 ребер. Точное решение: максимальный разрез из 536 ребер.
- g05\_100.0 - граф на 100 вершинах, который имеет 2475 ребер. Точное решение: максимальный разрез из 1430 ребер.

Граф на 60 вершинах:



Граф на 100 вершинах:



## Классические аппроксимационные алгоритмы

Из [2] известны аппроксимационные алгоритмы, гарантирующие разрез не меньше, чем  $\sim 0.88$  от максимально возможного.

## Связь с физикой

Известно, что задача MaxCut эквивалентна минимизации Гамильтониана для т.н. "спинового стекла" и в этом ключе и применяется.

Действительно, если мы представим, что наш граф это условно атомы со спинами +1 и -1, а связи графа это взаимодействия между ними, то мы можем записать Гамильтониан такого вида:  $H = \sum_{\langle i, j \rangle} (\sigma_z)_i (\sigma_z)_j - 1$

Константа выносится и мы приходим к такой записи:  $H = \sum_{\langle i, j \rangle} (\sigma_z)_i (\sigma_z)_j$

Состояние с минимальной энергией даст нам решение задачи MaxCut (и, соответственно, наоборот решение задачи MaxCut позволяет найти минимум такого Гамильтониана). Причем минимальная энергия связана с максимальным разрезом так:  $\text{MaxCutSize} = -(E_{\min} - |E|) / 2$

## Neural Quantum States

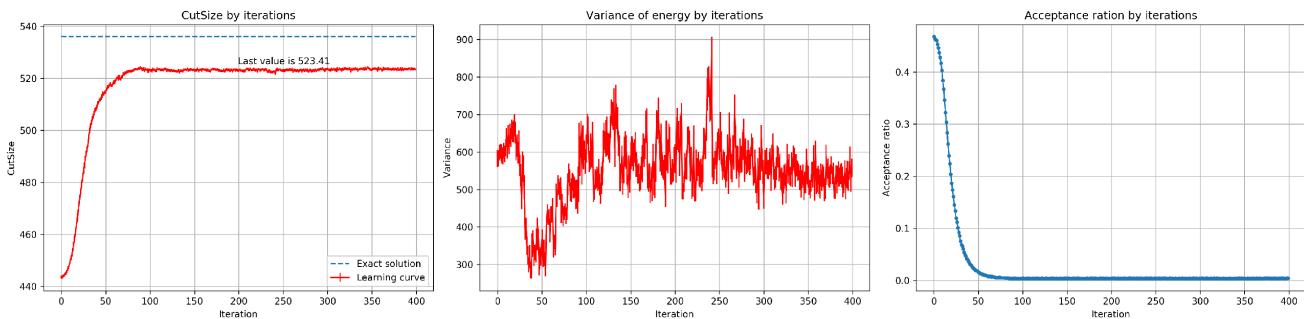
Идея предложена Carleo в 2017-м году [3]. Идея в том, чтобы заменить невычислимую волновую функцию  $\Psi(s)$  ( $s$  - состояние, например, комбинация наших спинов) выходом глубокой неройнной сети. В этом случае функция  $\Psi$  будет определять вероятность конкретного состояния, а значит, с применением известных MCMC-методов мы можем сэмплировать из нее состояния. Из физики известно, что состоянию с минимальной энергией будет соответствовать наибольшая вероятность. Таким образом, мы можем набрать сэмплов, оценить их локальную энергию, а также градиент энергии по самой волновой функции. Дальше применяя градиентные методы мы "обновляем" нашу волновую функцию (точнее ту нейронную сеть, которая ее заменяет) и пытаемся сделать так, чтобы средняя энергия по нашим сэмплам уменьшалась. Таким образом, мы приходим к такой  $\Psi$ -функции, сэмплируя из которой мы будем почти всегда получать состояние с энергией близкой к минимальной.

Большой плюс подхода из [3] в том, что он обобщается на произвольный гамильтониан.

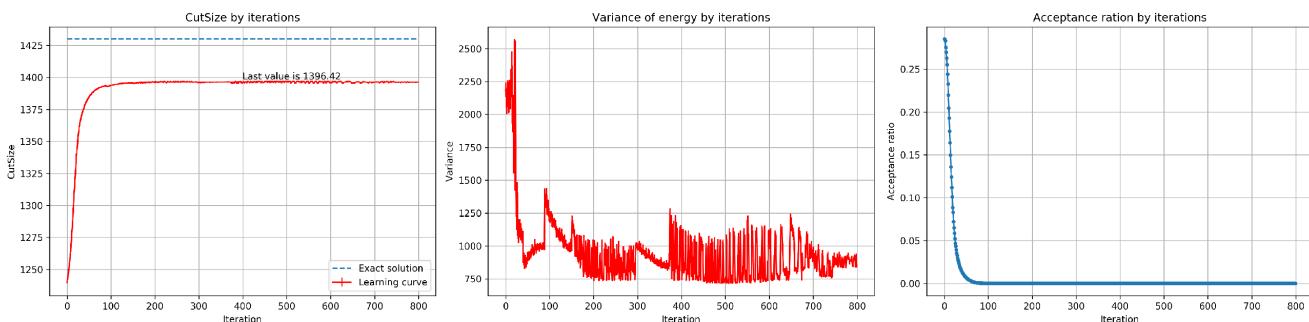
## Эксперимент

Для обоих наших графов, был произведен поиск энергии основного состояния с помощью метода из работы [3].

Результаты для графа из 60 вершин:



Результаты для графа из 100 вершин:

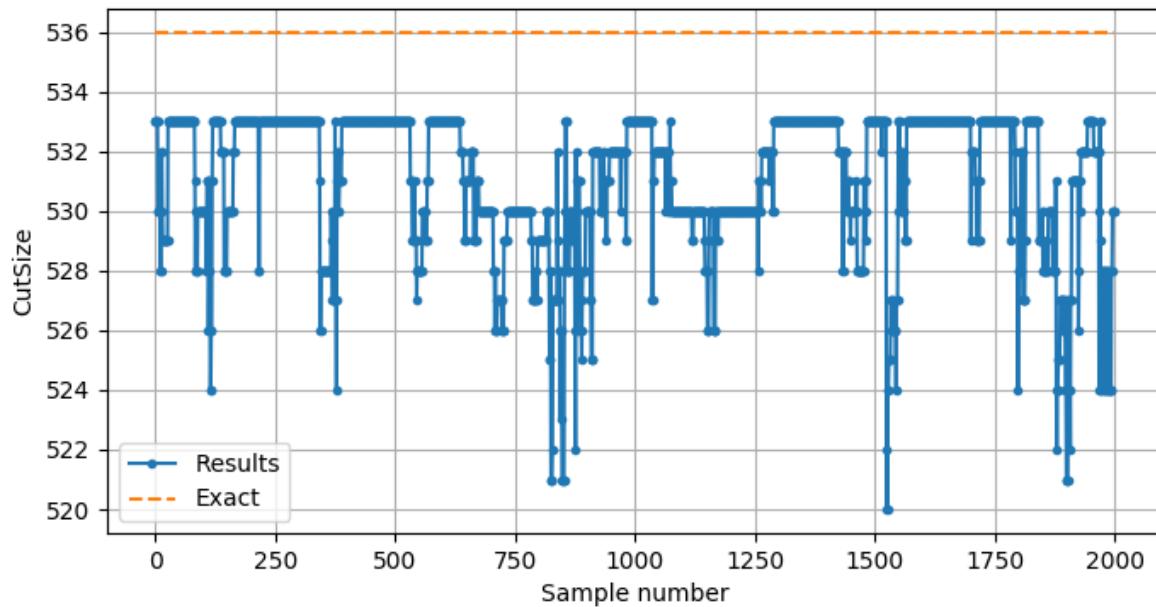


Можно видеть стабильно приближающееся к оптимуму решение, а также стабильно уменьшающееся разнообразность состояний - сэмплируя мы получаем все более похожие состояния.

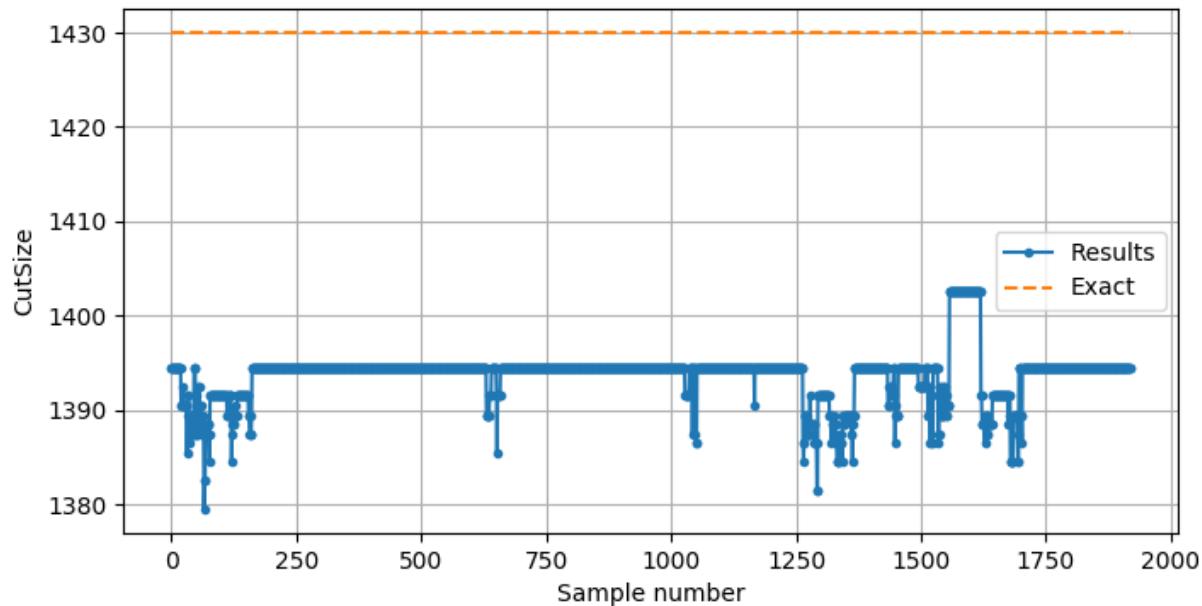
## Как получать итоговое решение?

Итоговое решение получается сэмплирование из нашей нейронной сети ( $\text{\Psi}$ -функции). Пример по 1000 сэмплов:

Граф на 60-и вершинах:



Граф на 100 вершинах:



Видим, что часто мы получаем решения очень близкие к оптимуму - итоговое качество на уровне 97% от оптимального разреза (что лучше гарантий аппроксимационных алгоритмов).

## Литература

- [1] [Biq Mac Library - A collection of Max-Cut and quadratic 0-1 programming instances of medium size](#), Angelika Wiegele, September 2007
- [2] [Information-Theoretic Analysis of MaxCut Algorithms](#), Yatao Bian, Alexey Gronskiy and Joachim M. Buhmann, July 2016

[3] Solving the quantum many-body problem with artificial neural networks, Giuseppe Carleo, Matthias Troyer, Feb 2017