Проектная работа.
Knapsack Problem

Студент 1: Статный Дмитрий
Студент 2: Смирнов Алексей

Весенний семестр, 2024

Вступление

Задача нашего проекта заключалась в рассмотрении возможной имплементации задачи Grid Scheduler в квантовой формулировке.

Многомерный Мультирюкзак

Формулировка в терминах задач и серверов

Дано N задач, M серверов, j задача имеет следующие параметры: потребление памяти диска α_j, потребление оперативной памяти β_j и требует ядер γ_j. Необходимо найти такое подмножество задач, чтобы распределение ресурсов по M серверам было максимально, но количество ядер потребляемых совокупностью задач на i сервере не превосходило заданной величины A_i, по потребляемой памяти диска – B_i, а также по оперативной памяти – C_i.

Многомерный мультирюкзак в QUBO

$$
\begin{aligned}
f_Q(x) = & -\lambda_1 \cdot \sum_{i=1}^{N}\sum_{j=1}^{M}(\alpha_{ij}+\beta_{ij}+\gamma_{ij})\cdot x_{ij} + \\
& + \lambda_2 \cdot \sum_{i=1}^{M}\left[\left(\sum_{j=1}^{N}\alpha_{ij}\cdot x_{ij}\right) + \left(\sum_{b=0}^{\lfloor \log_2 A_i \rfloor} 2^b \cdot \widetilde{a}_{ib}\right) - A_i\right]^2 + \\
& + \lambda_3 \cdot \sum_{i=1}^{M}\left[\left(\sum_{j=1}^{N}\beta_{ij}\cdot x_{ij}\right) + \left(\sum_{b=0}^{\lfloor \log_2 B_i \rfloor} 2^b \cdot \widetilde{b}_{ib}\right) - B_i\right]^2 + \\
& + \lambda_4 \cdot \sum_{i=1}^{M}\left[\left(\sum_{j=1}^{N}\gamma_{ij}\cdot x_{ij}\right) + \left(\sum_{b=0}^{\lfloor \log_2 C_i \rfloor} 2^b \cdot \widetilde{c}_{ib}\right) - C_i\right]^2 + \\
& + \lambda_5 \cdot \sum_{j=1}^{N}\left(\sum_{i=1}^{M} x_{ij}\right)\left(\sum_{i=1}^{M}(x_{ij}-1)\right)
\end{aligned}
$$

Заполнение матрицы

Пример одного из 15 блоков, как заполняется матрица в коде:

```
# пенальти на вместительность на xy
for i in range(num_knapsacks):
    for j in range(num_items):

        # вместительность по диску
        for b in range(data["num_slack_bits_array_space"][i]):
            index1 = i * num_items + num_slack_bits[:i].sum() + j
            index2 = (i + 1) * num_items + num_slack_bits[:i].sum() + b
            Q[index1][index2] = 2 * betta_1 * data_space_array[i][j] * 2 ** b
```

Решение задачи

Запуск решения происходит следующим образом:

```
Q, offset = builder.build_qubo(data)

bqm = BinaryQuadraticModel.from_qubo(Q, offset=offset)

response = SimulatedAnnealingSampler().sample(bqm, num_reads=data["num_reads"])
```

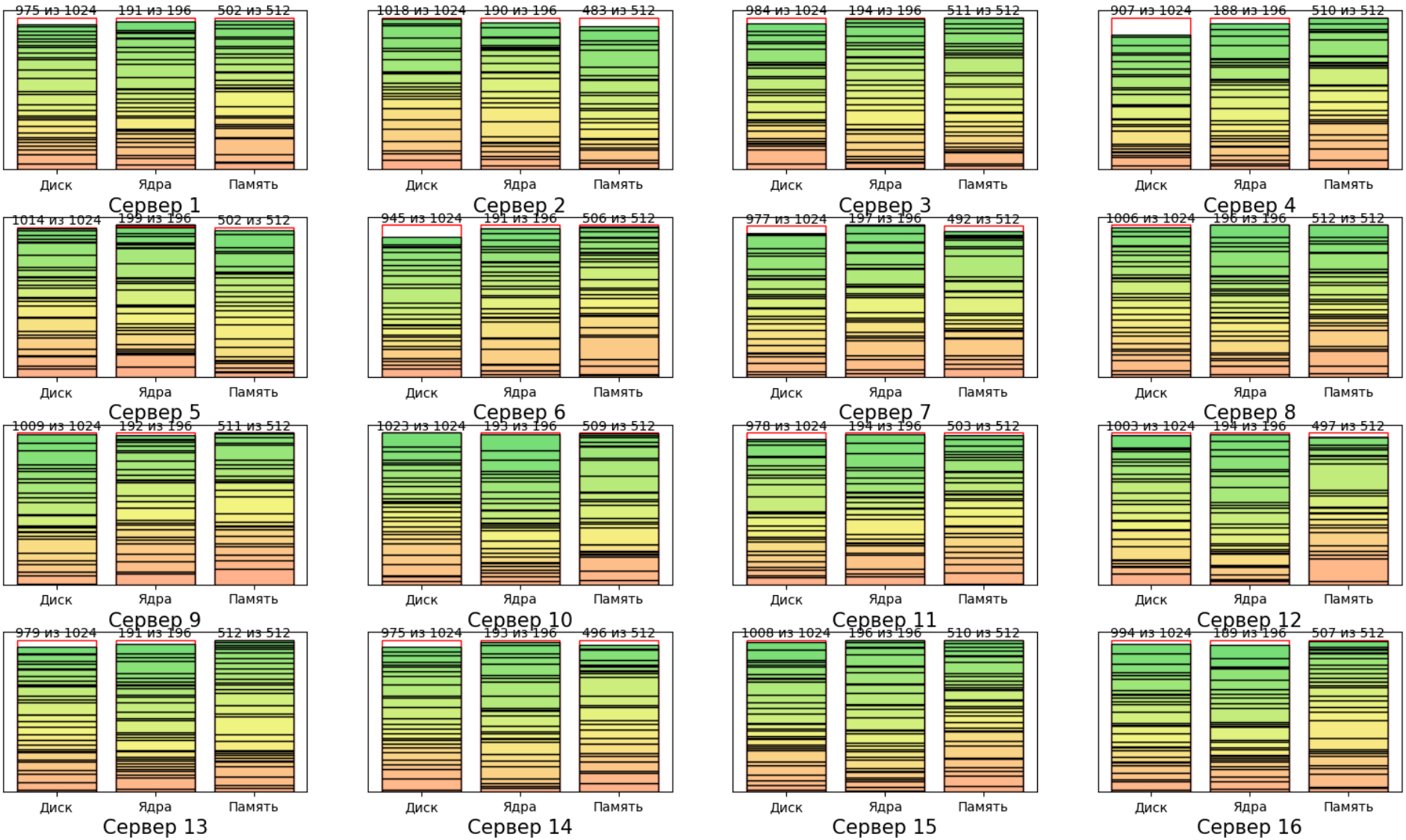
Анализ работы sampler

Посмотрим, что лежит в `response`:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	...	1031	energy	num_oc.
9	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	...	0	-9354.0	1
8	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	...	0	-9002.0	1
0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	1	...	0	-8861.0	1
6	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	...	0	-8397.0	1
3	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	...	0	-8360.0	1
2	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	...	0	-8184.0	1
1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	...	0	-8120.0	1
7	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	...	0	-7906.0	1
5	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	...	0	-7477.0	1
4	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	...	0	-7435.0	1

['BINARY', 10 rows, 10 samples, 1032 variables]

Пример работы



Общее заключение

Решение работает на высоком уровне по рассматриваемой метрике оценки решения.

Общее заключение

Решение работает на высоком уровне по рассматриваемой метрике оценки решения.

Но всё равно задача остаётся достаточно сложной для получения точного ответа на квантовом компьютере.

Репозиторий проекта