Модель QUBO для оптимизации проблемы маршрутизации транспортных средств

Морев Георгий
МФТИ

I. Введение

Задача маршрутизации транспортных средств (VRP) [1] является базовой NP-полной математической задачей, связанной с оптимизацией планирования, логистики и транспортировки. Цель состоит в том, чтобы спланировать поездки на транспортных средствах для максимально эффективного обслуживания заданного количества клиентов. Благодаря недавнему интересу к квантовым отжиговым машинам, которые стали коммерчески доступными компанией D-Wave Systems Inc. [2], исследование VRP как квадратичная безусловная бинарная оптимизация (QUBO) стало очень важным, особенно в попытке достичь квантово-механической оптимизации реальных проблем. Также стоит отметить, что в данной статье будет рассматриваться более обобщенная версия задачи, а именно MDCVRP (Multi-depot capacitated vehicle routing problem), то есть у каждой машины будет грузоподъёмность, будет несколько складов и количество ресурсов на каждом складе будет ограничено.

II. Постановка задачи

Пусть $G = (V, E)$ - полный граф, где $V = \{1, ..., n\}$ — набор вершин, представляющих n местоположений клиентов, также будут k вершин, являющиеся складами, E — набор неориентированных рёбер. У каждого ребра $(i, j) \in E, i \neq j$ есть неотрицательная стоимость D_{ij}. Эта стоимость может, например, представлять (географическое) расстояние между двумя клиентами i и j. Кроме того, предположим, что на складе размещено m_k транспортных средств с вместимостью Q_k. Кроме того, каждый клиент имеет определенный спрос q_i. MDCVRP состоит из поиска набора маршрутов транспортных средств, таких что:

- все маршруты начинаются и заканчиваются на депо
- каждый клиент в V посещается ровно один раз ровно одним транспортным средством
- сумма спроса клиентов на маршруте не превышает вместимости транспортных средств
- Для каждого маршрута сумма спросов клиентов не превышает вместимости транспортного средства
- сумма стоимостей всех маршрутов минимальна с учетом ограничений, указанных выше

III. Модель QUBO

Модель QUBO (Quadratic Unconstrained Binary Optimization) описывает задачу оптимизации в виде квадратичной формы:

$$H(x) = \sum_{1 \leq i \leq j \leq N} Q_{ij} x_i x_j = x^T Q x \tag{1}$$

, где:

- x — вектор бинарных переменных (0 или 1),
- Q — симметричная матрица коэффициентов размера $N \times N$.

QUBO минимизирует гамильтониан $H(x)$, который соответствует энергии системы.

IV. Модель

A. Параметры

Определим параметры, которые мы будем использовать:

- Задаваемые параметры
  - T - Множество всех клиентов
  - D - Множество всех складов
  - K - множество всех машин
  - $D_{ij}, (i, j \in T)$ - расстояние между клиентами
  - $D_{di}, (d \in D, i \in T)$ - расстояние между клиентами и складами
  - $V_d, (d \in D)$ - Количество товаров на складе
  - $Q_k, (k \in K)$ - Вместимость машины
  - $q_i, (i \in T)$ - Количество товаров необходимое клиенту
  - $y_{kd}, (k \in K, d \in D)$ - 1, если машина k стоит на складе d
- Оптимизируемые параметры
  - $x_{ijk}, (i, j \in T, k \in K)$ - 1, если машина k поедет к клиенту j после клиента i
  - u_{ik} - 1, если машина k посетит клиента i первым
  - n_{ik} - 1, если машина k посетит клиента i последним

B. Целевые функции и ограничения

Целевая функция: Функия описывающая издержки для передвижения всех машин. Данную функцию мы хотим минимизировать.

$$\text{Min: } \sum_{k\in K}\sum_{i\in T}\sum_{j\in T} D_{ij}x_{ijk} + \sum_{k\in K}\sum_{i\in T}\sum_{d\in D} D_{id}u_{ik}y_{kd} + \sum_{k\in K}\sum_{i\in T}\sum_{d\in D} D_{id}n_{ik}y_{kd} \tag{2}$$

Ограничение 1: Нельзя поехать к тому же покупателю.

$$\forall i \in T, \forall k \in K: \ x_{iik} = 0 \tag{3}$$

Ограничение 2: К каждому покупателю должна приехать ровно одна машина.

$$\forall i \in T: \ \sum_{k\in K}\sum_{j\in T} x_{jik} + \sum_{k\in K} u_{ik} = 1 \tag{4}$$

Ограничение 3: От каждого покупателя должна уехать ровно одна машина.

$$\forall i \in T: \ \sum_{k\in K}\sum_{j\in T} x_{ijk} + \sum_{k\in K} n_{ik} = 1 \tag{5}$$

Ограничение 4: Каждая машина должна отъехать ровно с одного склада.

$$\forall k \in K: \ \sum_{i\in T} u_{ik} = 1 \tag{6}$$

Ограничение 5: Каждая машина должна приехать ровно на один склад.

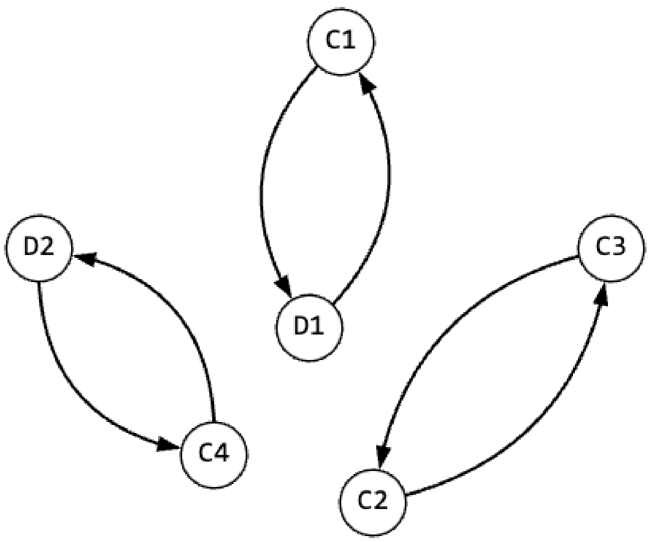
$$\forall k \in K: \ \sum_{i\in T} n_{ik} = 1 \tag{7}$$

Ограничение 6: "Целостность маршрута". Если машина приехала к покупателю, то она должна уехать от него.

$$\forall k \in K, \forall i \in T: \quad u_{ik} + \sum_{j\in T} x_{jik} - n_{ik} - \sum_{p\in T} x_{ipk} = 0 \tag{8}$$

Ограничение 7: У машин не должно быть возможности иметь на своем маршруте цикл из городов, который они не посещают.

Пример, удовлетворяющий ограничениям 1-6, который мы хотим запретить.



Например, представим сценарий, когда дано 2 склада D_1, D_2, в каждом из которых по 1 машине, V_1 на D_1 и V_2 на D_2. Пусть также есть 4 покупателя: $\mathrm{C}_i, i = 1...4$. Потенциально маршруты могли распределиться так:

- Для $V_1: \ D_1 \to C_1 \to D_1, C_2 \to C_3, C_3 \to C_2$
- Для $V_2: \ D_2 \to C_4 \to D_2$

Данные маршруты удовлетворяют всем ограничениям, наложенным выше, но не являются правильными потому что в цикл из покупателей C_2, C_3 никто никогда не заедет.

Обозначим за $\mathbb{P}(T)$ - множество всех подмножеств T, тогда необходимо наложить ограничение:

$$\forall S \in \mathbb{P}(t), 2 \le |S| \le |T|: \ \sum_{k\in K}\sum_{i,j\in S} x_{ijk} \le |S| - 1 \tag{9}$$

Ограничение 8: Суммарное количество продуктов, доставленное покупателям не должно превышать вместимости машины.

$$\forall k \in K: \ \sum_{i\in T}\sum_{j\in T} q_i x_{ijk} + \sum_{i\in T} q_i n_{ik} \le Q_k \tag{10}$$

Ограничение 9: Суммарное количество продуктов, доставленное покупателям с каждого склада не должно превышать его вместимости.

$$\forall d \in D: \quad \sum_{k\in K} y_{kd}\left(\sum_{i\in T}\sum_{j\in T} q_i x_{ijk} + \sum_{i\in T} q_i n_{ik}\right) \le V_d \tag{11}$$

Замечание: при реализации стоит создать фиктивные оптимизируемые параметры u_{0k} и n_{0k}, которые будут равны 1 в случае, если машине не надо никуда ехать.

C. Гамильтониан задачи

Общий подход:

Для ограничений вида $\sum_{i=1}^{n_x} A_i x_i = b$, где x_i - i-ый оптимизируемый параметр, Гамильтониан будет записан как $\left(\sum_{i=1}^{n_x} A_i x_i - b\right)^2$

Для ограничений вида $\sum_{i=1}^{n_x} A_i x_i \le b$, чтобы представить модель QUBO придется сначала представить ограничение в виде равенства [3]. Это можно сделать путем введения резервных переменных, и записать Гамильтониан в виде $\left(\sum_{i=1}^{n_x} A_i x_i + \sum_{j=0}^{n_\lambda} 2^\lambda \lambda_j - b\right)^2$, где $n_\lambda = \lceil \log_2(b+1) \rceil$.

Гамильтониан целевой функции:

$$H_O = \sum_{k\in K}\sum_{i\in T}\sum_{j\in T} D_{ij}x_{ijk} + \sum_{k\in K}\sum_{i\in T}\sum_{d\in D} D_{id}u_{ik}y_{kd} + \sum_{k\in K}\sum_{i\in T}\sum_{d\in D} D_{id}n_{ik}y_{kd} \tag{12}$$

Ниже приведены члены Гамильтониана H_{C_i}, которые представляют i-ое ограничение, приведенное выше.

$$H_{C_1} = B\sum_{i\in T}\sum_{k\in K}(x_{iik}) \tag{13}$$

$$H_{C_2} = B\sum_{i\in T}\left(1 - \left(\sum_{k\in K}\sum_{j\in T} x_{jik} + \sum_{k\in K} u_{ik}\right)\right)^2 \tag{14}$$

$$H_{C_3} = B\sum_{i\in T}\left(1 - \left(\sum_{k\in K}\sum_{j\in T} x_{ijk} + \sum_{k\in K} n_{ik}\right)\right)^2 \tag{15}$$

$$H_{C_4} = B \sum_{k \in K} \left(1 - \left(\sum_{i \in T} u_{ik}\right)\right)^2 \quad (16)$$

$$H_{C_5} = B \sum_{k \in K} \left(1 - \left(\sum_{i \in T} n_{ik}\right)\right)^2 \quad (17)$$

$$H_{C_6} = B \sum_{i \in T} \sum_{k \in K} \Bigg(u_{ik} + \sum_{j \in T} x_{jik} - n_{ik} - \sum_{p \in T} x_{ipk}\Bigg)^2 \quad (18)$$

$$H_{C_7} = B \sum_{S \in \mathbb{P}(T) 2 \leq \ |S| \ \leq \ |T|} \Bigg(\sum_{k \in K} \sum_{i,j \in S} x_{ijk} + \sum_{l=0}^{\lceil \log_2(|S|) \rceil} 2^l \lambda_{lS} - |S| + 1\Bigg)^2 \quad (19)$$

$$H_{C_8} = B \sum_{k \in K} \Bigg(\sum_{i \in T} \sum_{j \in T} q_i x_{ijk} + \sum_{i \in T} q_i n_{ik} + \sum_{l=0}^{\lceil \log_2(Q_k+1) \rceil} 2^l \lambda_{lk} - Q_k\Bigg)^2 \quad (20)$$

$$H_{C_9} = B \sum_{d \in D} \Bigg(\sum_{k \in K} y_{kd} \left(\sum_{i \in T} \sum_{j \in T} q_i x_{ijk} + \sum_{i \in T} q_i n_{ik}\right) + \sum_{l=0}^{\lceil \log_2(V_d+1) \rceil} 2^l \lambda_{ld} - V_d\Bigg)^2 \quad (21)$$

Где B - большая константа, обозначающая штраф, который возникнет при нарушении ограничений.

Тогда Гамильтониан данной задачи получился равен:

$$H_F = H_O + \sum_{i=1}^{9} H_{C_i} \quad (22)$$

V. Реалмзация

Для нас удобно, что это уже задача минимизации. В данном случае QUBO-матрица получается при помощи явного раскрытия скобок в выражении для стоимости. Можно заметить, что в этом случае получаем также элементы 0-й степени, но формат QUBO-матрицы такого не предусматривает. Но во-первых, в данном случае легко можем определить разницу между $X^T Q X$ и минимумом H_F, а во-вторых, для нас это не столь важно – нам нужно решение, а значение энергии/функции стоимости получается без каких-либо проблем за полиномиальное время. [4]

Для того, чтобы не усложнять реализацию, было написано решение задачи MDVRP, то есть мы не учитываем ограничения 8-9 (на вместимость складов и грузоподъёмность курьеров).

[Реализация на Python]

VI. Оптимизации

Можно заметить, что количество оптимизируемых параметров можно оценить как $O(2^{|T|})$, где T - множество всех клиентов, т.к. нам нужно выполнить ограничение 7, которое запрещает иметь цикл, не пересекающийся с путем, то есть есть ограничение на каждое подмножество размера больше 2, количество которых оценивается как $O(2^{|T|})$.

Для более оптимального решения предлагается воспользоваться Формулой Миллера-Такера-Землина [5].

Основная её идея заключается в том, чтобы поставить в соответсвие каждому клиенту оптимизируемую переменную $t_i \in \mathbb{N}_0$ и поддерживать инвариант:

$$\forall k \in K, \forall i, j \in T : x_{ijk} = 1 \rightarrow t_i < t_j \quad (23)$$

Таким образом количество оптимизируемых параметров для данного ограничения снизится с $2^{|T|}$ бинарных до $|T|$ целочисленных, которые будут задавать t_i и $|T| * |T| * |K|$ целочисленных для сведения неравенства к матрице QUBO.

Цикл в таком случае невозможен, так как на цикле можно найти такой путь, который пройдет дважды по одной вершине, но тогда инвариант о том, что при переходе к следующей вершине её параметр t увеличивается.

Заметим, что для любого расположения вершин на маршруте можно найти такие параметры t_i, что $\forall i \in T : t_i < |T|$.

Необходимо обойти все маршруты начиная с каждого из складов и поставить в соответствие каждому городу число от 0 до $|T| - 1$ по возрастанию в порядке обхода.

■

Ограничение 10: У машин не должно быть возможности иметь на своем маршруте цикл из городов, который они не посещают (оптимальная версия).

Требуемое ограничение можно записать следующим образом:

$$\forall i, j \in T : t_i - t_j \leq \begin{cases} -1, \text{ если } \sum_{k \in K} x_{ijk} = 1 \\ |T| - 1, \text{ иначе} \end{cases} \quad (24)$$

Проведем преобразования, чтобы описать данное ограничение Гамильтонианом.

$$\forall i, j \in T : t_i - t_j \leq -1 + |T| \left(1 - \sum_{k \in K} x_{ijk}\right) \quad (25)$$

Тогда запишем Памильтониан, выразив оптимизируемую переменную через бинарные следующим образом: $t_i = \sum_{l=0}^{\lceil \log_2(t_i+1) \rceil} t_{li}$ используя уже известное сведение [3]:

$$H_{C_{10}} = B \sum_{\substack{i,j \in T \\ i \neq j}} \left(\sum_{l=0}^{\lceil \log_2(|T|+1) \rceil} 2^l t_{li} - \sum_{l=0}^{\lceil \log_2(|T|+1) \rceil} 2^l t_{lj} + \sum_{l=0}^{\lceil \log_2(2(|T|+1)) \rceil} 2^l \lambda_{l_{ij}} + |T| \sum_{k \in K} x_{ijk} - |T| + 1 \right)^2 \tag{26}$$

Таким образом, теперь можно использовать ограничение 10 вместо ограничения 7.

VII. Методы решения на классическом компьютере

A. *Полный перебор*

Для каждой из машин рекурсивно переберем количество клинетов, которые будут к ней относиться и также рекурсивно выбираем количество тех клиентов, которые относятся к текущей машине, передаваая оставшихся клиентов следующим машинам, затем после того, как зафиксировано, какая машина обслуживает какой набор клиентов возьмем декартово произведение между всеми возможными перестановками для каждой машины.

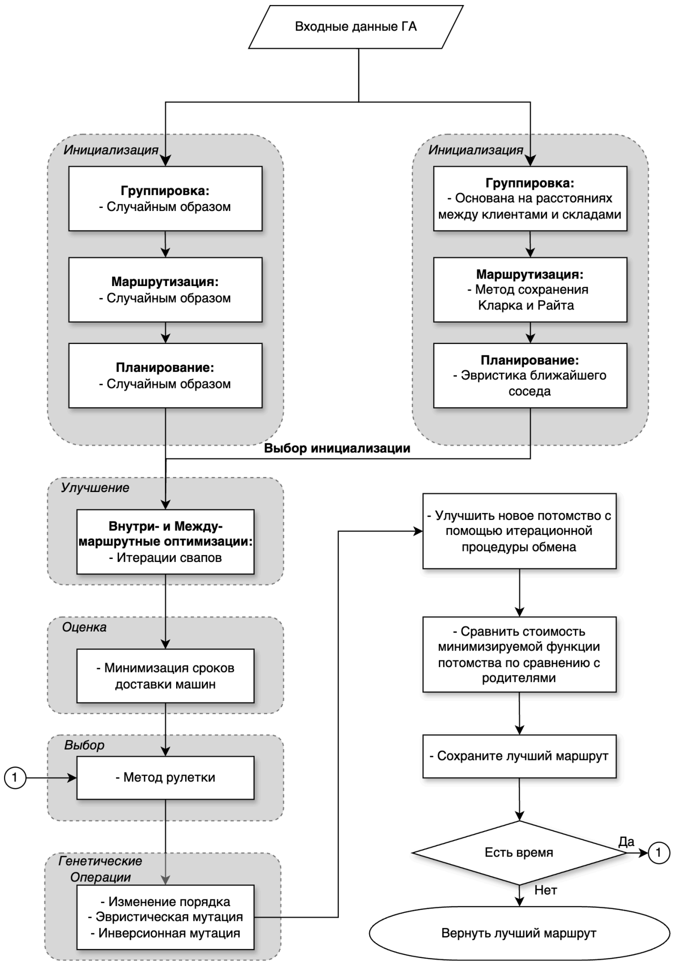
Пример: Хотим разделить 3 клиентов между 2 машинами

- Первая машина обслуживает 0 клиентов → вторая - 3 клиентов
  - Добавить к ответам все перестановки ({}, {1, 2, 3})
    - Ответ: **((), (1, 2, 3))**
    - Ответ: **((), (1, 3, 2))**
    - Ответ: **((), (2, 1, 3))**
    - Ответ: **((), (2, 3, 1))**
    - Ответ: **((), (3, 1, 2))**
    - Ответ: **((), (3, 2, 1))**
- Первая машина обслуживает 1 клиента → вторая - 2 клиентов
  - Первая обслуживает {1} → добавить к ответам все перестановки ({1}, {2, 3})
    - Ответ: **((1), (2, 3))**
    - Ответ: **((1), (3, 2))**
  - Первая обслуживает {2} → добавить к ответам все перестановки ({2}, {1, 3})
    - Ответ: **((2), (1, 3))**
    - Ответ: **((2), (3, 1))**
  - Первая обслуживает {3} → добавить к ответам все перестановки ({3}, {1, 2})
    - Ответ: **((3), (1, 2))**
    - Ответ: **((3), (2, 1))**
- Первая машина обслуживает 2 клиентов → вторая - 1 клиента
  - Первая обслуживает {1, 2} → добавить к ответам все перестановки ({1, 2}, {3})
    - Ответ: **((1, 2), (3))**
    - Ответ: **((2, 1), (3))**
  - Первая обслуживает {1, 3} → добавить к ответам все перестановки ({1, 3}, {2})
    - Ответ: **((1, 3), (2))**
    - Ответ: **((3, 1), (2))**
  - Первая обслуживает {2, 3} → добавить к ответам все перестановки ({2, 3}, {1})
    - Ответ: **((2, 3), (1))**
    - Ответ: **((3, 2), (1))**
- Первая машина обслуживает 3 клиентов → вторая - 0 клиентов
  - Добавить к ответам все перестановки ({1, 2, 3}, {})
    - Ответ: **((1, 2, 3), ())**
    - Ответ: **((1, 3, 2), ())**
    - Ответ: **((2, 1, 3), ())**
    - Ответ: **((2, 3, 1), ())**
    - Ответ: **((3, 1, 2), ())**
    - Ответ: **((3, 2, 1), ())**

B. *Генетический алгоритм [6]*

Данный алгоритм представляет собой гибридный генетический алгоритм (HGA) для решения задачи MDVRP. Основная идея заключается в комбинации генетического алгоритма с эвристическими методами, такими как метод сбережений Кларка-Райта и метод ближайшего соседа. Алгоритм включает три ключевых этапа: группировку клиентов по депо, маршрутизацию внутри каждого депо и оптимизацию последовательности доставки. Данные этапы можно сделать как сгенеровав начальное решения путем случайного распределения клиентов по депо и маршрутам, так и жадными алгоритмами. После этого для улучшения решений используется процедура итераций, где случайные мутации (например, swap, insert, 2-opt) модифицируют маршруты, а новые решения принимаются либо жадно, либо с вероятностным критерием (например, алгоритм Метрополиса-Гастингса). Такой подход прост в рализации и позволяет избегать локальных оптимумов, но не гарантирует высокого качества решений и может требовать тонкой настройки параметров (например, температуры в имитации отжига). Для улучшения эффективности случайные методы часто комбинируют с Критерием остановки может служить максимальное число итераций, отсутствие улучшений или ограничение по времени. Хотя метод гибок и легко адаптируется, его основными недостатками остаются медленная сходимость для больших задач и зависимость от начальных условий.

Работу генетического алгоритма можно описать следующей схемой:



VIII. Сравнение результатов

References

[1] J. H. Dantzig G. B. Ramser, "The Truck Dispatching Problem," *Management Science*, pp. 80–91, 1959.

[2] G. S. L. T. H. F. D. N. R. B. A. J. J. J. B. P. C. E. M. E. C. J. P. K. K. L. E. L. N. O. T. P. I. R. C. T. M. C. T. E. T. C. J. S. U. S. W. J. W. B. G. Johnson M. W. Amin M. H. S., "Quantum annealing with manufactured spins," *Nature 473*, 2011.

[3] T. Vyskoˇcil S. Pakin and H. N. Djidjev, "Embedding Inequality Constraints for Quantum Annealing Optimization," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019.

[4] K. Y. Sinchenko Semyon, "Quantum Machine Learning Course," *https://quantum-ods.github.io/qmlcourse/book/problems2qml/ru/np2ising.html#id10*, 2021.

[5] "Miller-Tucker-Zemlin Formulation," *https://how-to.aimms.com/Articles/332/332-Miller-Tucker-Zemlin-formulation.html*, 2024.

[6] P. J. H. C. L. William Hoa George T.S. Ho, "A hybrid genetic algorithm for the multi-depot vehicle routing problem," *https://shorturl.at/i7Se5*, 2007.