Проектная работа. Knapsack problem

Студент 01: Статный Дмитрий
Студент 02: Смирнов Алексей
Студент 03: Смирнов Марк

ВШПИ МФТИ

Весенний семестр

Overview

1D-Knapsack problem

Формулировка проблемы

Дано N предметов, n_i предмет имеет массу $w_i > 0$ и стоимость $c_i > 0$. Необходимо выбрать из этих предметов такой набор, чтобы суммарная масса не превосходила заданной величины W (вместимость рюкзака), а суммарная стоимость была максимальна.

1D-Knapsack problem

Формулировка проблемы

Дано N предметов, n_i предмет имеет массу $w_i > 0$ и стоимость $c_i > 0$. Необходимо выбрать из этих предметов такой набор, чтобы суммарная масса не превосходила заданной величины W (вместимость рюкзака), а суммарная стоимость была максимальна.

Линейно-алгебраическая формулировка

Дано два вектора $w = (w_0, w_1, \dots, w_{n-1}) \in \mathbb{R}^n$ и $c = (c_0, c_1, \dots, c_{n-1}) \in \mathbb{R}^n$. Нужно построить битовый вектор $b \in \{0,1\}^n$:

$$\begin{cases} (w, c) \to \mathbf{max} \\ (w, b) \leqslant W \end{cases}$$

Точные алгоритмы

Решение, которое позволяет получить эталонный (точный) ответ, существует.

Точные алгоритмы

Решение, которое позволяет получить эталонный (точный) ответ, существует.

Асимптотическая сложность данных алгоритмов не является привлекательной для больших наборов данных:

Точные алгоритмы

Решение, которое позволяет получить эталонный (точный) ответ, существует.

Асимптотическая сложность данных алгоритмов не является привлекательной для больших наборов данных:

Точные алгоритмы:

- Полный перебор – $\mathcal{O}(2^N)$

Точные алгоритмы

Решение, которое позволяет получить эталонный (точный) ответ, существует.

Асимптотическая сложность данных алгоритмов не является привлекательной для больших наборов данных:

Точные алгоритмы:

- Полный перебор – $\mathcal{O}(2^N)$
- Meet-in-Middle решение – $\mathcal{O}(2^{N/2}N)$

Точные алгоритмы

Решение, которое позволяет получить эталонный (точный) ответ, существует.

Асимптотическая сложность данных алгоритмов не является привлекательной для больших наборов данных:

Точные алгоритмы:

- Полный перебор – $\mathcal{O}(2^N)$
- Meet-in-Middle решение – $\mathcal{O}(2^{N/2}N)$
- Динамическое программирование – $\mathcal{O}(NW)$

Точные алгоритмы

Решение, которое позволяет получить эталонный (точный) ответ, существует.

Асимптотическая сложность данных алгоритмов не является привлекательной для больших наборов данных:

Точные алгоритмы:

- Полный перебор – $\mathcal{O}(2^N)$
- Meet-in-Middle решение – $\mathcal{O}(2^{N/2}N)$
- Динамическое программирование – $\mathcal{O}(NW)$
- Метод ветвей и границ – $\mathcal{O}(2^N)$

Описание точных алгоритмов

- Полный перебор

  Простое и очевидное решение – осуществить полный перебор возможных комбинаций и выбрать самую оптимальную из них. На наборе, состоящем из 50 предметов, придётся подождать 13 лет.

Описание точных алгоритмов

- Полный перебор

  Простое и очевидное решение – осуществить полный перебор возможных комбинаций и выбрать самую оптимальную из них. На наборе, состоящем из 50 предметов, придётся подождать 13 лет.

- Динамическое программирование

  Идея решения заключается в решении подзадач с меньшим максимальным весом и дальнейшем переходе к большему весу. Асимптотика линейно зависит от второго параметра – вместимости рюкзака, что делает это решение непригодным даже на некоторых малых наборах данных.

Описание точных алгоритмов

- Метод ветвей и границ

  Данный метод отличается от полного перебора лишь тем, что мы убираем из рассмотрения ветви, которые не принесут успеха. Но возможны и ситуации, когда таких ветвей не будет. Поэтому сложность данного алгоритма ничем не лучше, чем у полного переборного алгоритма.

Описание точных алгоритмов

- Метод ветвей и границ

  Данный метод отличается от полного перебора лишь тем, что мы убираем из рассмотрения ветви, которые не принесут успеха. Но возможны и ситуации, когда таких ветвей не будет. Поэтому сложность данного алгоритма ничем не лучше, чем у полного переборного алгоритма.

- Middle-in-Meet

  Изначальное множество вещей N разобьём на два равных (или примерно равных) подмножества, для которых за приемлемое время можно перебрать все варианты и подсчитать суммарный вес и стоимость, а затем для каждого из них найти группу предметов из первого подмножества с максимальной стоимостью, укладывающуюся в ограничение по весу рюкзака. На наборе, состоящем из 90 предметов, придётся ждать 36 лет.

Аппроксимирующие алгоритмы

Из-за высокой сложности проблемы начали появляться аппроксимирующие алгоритмы, позволяющие получить ответ на задачу с некоторой точностью.

Аппроксимирующие алгоритмы

Из-за высокой сложности проблемы начали появляться аппроксимирующие алгоритмы, позволяющие получить ответ на задачу с некоторой точностью.

Аппроксимирующие алгоритмы:

- Жадный алгоритм – $\mathcal{O}(N \log N)$

Аппроксимирующие алгоритмы

Из-за высокой сложности проблемы начали появляться аппроксимирующие алгоритмы, позволяющие получить ответ на задачу с некоторой точностью.

Аппроксимирующие алгоритмы:

- Жадный алгоритм – $\mathcal{O}(N \log N)$
- Приближенная схема полностью полиномиального времени – $\mathrm{poly}(N)$

Аппроксимирующие алгоритмы

Из-за высокой сложности проблемы начали появляться аппроксимирующие алгоритмы, позволяющие получить ответ на задачу с некоторой точностью.

Аппроксимирующие алгоритмы:

- Жадный алгоритм – $\mathcal{O}(N \log N)$
- Приближенная схема полностью полиномиального времени – $\mathrm{poly}(N)$
- Генетические алгоритмы – $\mathrm{poly}(N)$

Аппроксимирующие алгоритмы

Из-за высокой сложности проблемы начали появляться аппроксимирующие алгоритмы, позволяющие получить ответ на задачу с некоторой точностью.

Аппроксимирующие алгоритмы:

- Жадный алгоритм – $\mathcal{O}(N \log N)$
- Приближенная схема полностью полиномиального времени – $\mathrm{poly}(N)$
- Генетические алгоритмы – $\mathrm{poly}(N)$

Описание аппроксимирующих алгоритмов

- Жадный алгоритм

  Идея данного алгоритма заключается в сортировке данного набора предметов по удельной стоимости (отношение стоимости предмета к его весу) и жадному набору предметов в рюкзак. Ввиду такого подхода, решение может быть далеко от оптимального.

Описание аппроксимирующих алгоритмов

- Жадный алгоритм

  Идея данного алгоритма заключается в сортировке данного набора предметов по удельной стоимости (отношение стоимости предмета к его весу) и жадному набору предметов в рюкзак. Ввиду такого подхода, решение может быть далеко от оптимального.

- Приближённая схема полностью полиномиального времени

  Реализация такого алгоритма очень сильно зависит от постановки проблемы. Идея, стоящая за классической схемой, заключается в снижении точности, с которой заданы ценности предметов. Объединяя предметы близкой ценности в одну группу, можно снизить количество разных предметов.

Описание аппроксимирующих алгоритмов

- Генетические алгоритмы

  Каждая особь (генотип) представляет собой подмножество предметов, которые мы хотим упаковать в ранец (их общий вес может превысить допустимую грузоподъемность). Для удобства информация хранится в виде бинарных строк, в которых каждый бит определяет, помещается ли этот предмет в ранец.

Описание аппроксимирующих алгоритмов

- Генетические алгоритмы

  Каждая особь (генотип) представляет собой подмножество предметов, которые мы хотим упаковать в ранец (их общий вес может превысить допустимую грузоподъемность). Для удобства информация хранится в виде бинарных строк, в которых каждый бит определяет, помещается ли этот предмет в ранец.

  Функция приспособленности определяет близость решения к оптимальному. Например, таковой может служить суммарная ценность предметов, при условии, что суммарный вес не превосходит грузоподъемность.

Описание аппроксимирующих алгоритмов

- Генетические алгоритмы

  Каждая особь (генотип) представляет собой подмножество предметов, которые мы хотим упаковать в ранец (их общий вес может превысить допустимую грузоподъемность). Для удобства информация хранится в виде бинарных строк, в которых каждый бит определяет, помещается ли этот предмет в ранец.

  Функция приспособленности определяет близость решения к оптимальному. Например, таковой может служить суммарная ценность предметов, при условии, что суммарный вес не превосходит грузоподъемность.

  После серии смен поколений, в которых скрещиваются наиболее приспособленные особи и игнорируются оставшиеся, алгоритм, по предположению, должен улучшить исходные решения.

Решение на квантовом компьютере

Квантовые компьютеры DWave, к примеру, умеют решать лишь одну задачу – нахождение минимума энергии гамильтониана, поэтому для получения ответа на какую-то проблему необходимо интерпретировать условие в матрицу QUBO или модель Изинга.

Решение на квантовом компьютере

Квантовые компьютеры DWave, к примеру, умеют решать лишь одну задачу – нахождение минимума энергии гамильтониана, поэтому для получения ответа на какую-то проблему необходимо интерпретировать условие в матрицу QUBO или модель Изинга.

QUBO (Quadratic unconstrained binary optimization) формулировка

$$f_Q(x) = x^T Q x = \sum_{i=1}^{N} \sum_{j \geqslant i}^{N} Q_{ij} x_i x_j, \text{ где } x_i \in \{0, 1\}$$

Решение на квантовом компьютере

Квантовые компьютеры DWave, к примеру, умеют решать лишь одну задачу – нахождение минимума энергии гамильтониана, поэтому для получения ответа на какую-то проблему необходимо интерпретировать условие в матрицу QUBO или модель Изинга.

QUBO (Quadratic unconstrained binary optimization) формулировка

$$f_Q(x) = x^T Q x = \sum_{i=1}^{N} \sum_{j \geqslant i}^{N} Q_{ij} x_i x_j, \text{ где } x_i \in \{0, 1\}$$

Модель Изинга

$$\hat{H} = -\sum_{i=1}^{N} J_{ij} \sigma_i \sigma_j - \mu \sum_{j=1}^{N} h_j \sigma_j, \text{ где } \sigma_i \in \{-1, 1\}$$

Ising <-> QUBO

QUBO (Quadratic unconstrained binary optimization) формулировка

$$f_Q(x) = x^T Q x = \sum_{i=1}^{N} \sum_{j \geqslant i}^{N} Q_{ij} x_i x_j, \text{ где } x_i \in \{0, 1\}$$

Ising <-> QUBO

QUBO (Quadratic unconstrained binary optimization) формулировка

$$f_Q(x) = x^T Q x = \sum_{i=1}^{N} \sum_{j \geqslant i}^{N} Q_{ij} x_i x_j, \text{ где } x_i \in \{0, 1\}$$

Модель Изинга

$$\hat{H} = -\sum_{i=1}^{N} J_{ij} \sigma_i \sigma_j - \mu \sum_{j=1}^{N} h_j \sigma_j, \text{ где } \sigma_i \in \{-1, 1\}$$

Ising <-> QUBO

QUBO (Quadratic unconstrained binary optimization) формулировка

$$f_Q(x) = x^T Q x = \sum_{i=1}^{N} \sum_{j \geqslant i}^{N} Q_{ij} x_i x_j, \text{ где } x_i \in \{0, 1\}$$

Модель Изинга

$$\hat{H} = -\sum_{i=1}^{N} J_{ij} \sigma_i \sigma_j - \mu \sum_{j=1}^{N} h_j \sigma_j, \text{ где } \sigma_i \in \{-1, 1\}$$

Несложно видеть, что простой заменой $x_i = \frac{\sigma_i + 1}{2}$ можно перейти от QUBO к Ising. И при $\sigma_i = 2x_i - 1$ к Ising модели.

Приходим к QUBO...

Необходимо понять, как свести данную постановку проблему к матрице QUBO, чтобы была возможность получить аппроксимирующее решение через достижение минимума энергии гамильтониана.

Приходим к QUBO...

Необходимо понять, как свести данную постановку проблему к матрице QUBO, чтобы была возможность получить аппроксимирующее решение через достижение минимума энергии гамильтониана.

Постановка проблемы имеет в себе знак неравенства, который мы не можем просто перенести в нашу матрицу, ведь у нас есть возможность использовать лишь равенства.

Приходим к QUBO...

Во-первых, разберёмся с самой записью и сделаем некоторые преобразования – выделим линейную часть:

$$\min_x x^T Q x = \min_x \left(\sum_{i=1}^{N} \sum_{j \geqslant i}^{N} Q_{ij} x_i x_j\right) = \dots$$

Приходим к QUBO...

Во-первых, разберёмся с самой записью и сделаем некоторые преобразования – выделим линейную часть:

$$\min_x x^T Q x \;=\; \min_x \Big(\sum_{i=1}^{N} \sum_{j \geqslant i}^{N} Q_{ij} x_i x_j\Big) = \ldots$$

Заметим, что в силу определений $x_i x_i \equiv x_i$:

$$\cdots = \min_x \Big(\sum_{i=1}^{N} Q_{ii} x_i \;+\; \sum_{i=1}^{N} \sum_{j > i}^{N} Q_{ij} x_i x_j\Big)$$

Приходим к QUBO...

Во-первых, разберёмся с самой записью и сделаем некоторые преобразования – выделим линейную часть:

$$\min_x x^T Q x = \min_x \left(\sum_{i=1}^{N} \sum_{j \geqslant i}^{N} Q_{ij} x_i x_j\right) = \ldots$$

Заметим, что в силу определений $x_i x_i \equiv x_i$:

$$\cdots = \min_x \left(\sum_{i=1}^{N} Q_{ii} x_i + \sum_{i=1}^{N} \sum_{j > i}^{N} Q_{ij} x_i x_j\right)$$

Задача поиска минимума гамильтониана эквивалентна нахождению минимума $f_Q(x) = x^T Q x$, где Q – матрица QUBO.

Приходим к QUBO...

Что в данном случае будем принимать за x_i? В данном случае это переменные регистра. Если в какой-то момент времени $x_i = 1$, означает, что данный предмет находится в рюкзаке. В противном случае, данного предмета нет в рюкзаке. Следовательно, все слагаемые с предметами, которых нет в рюкзаке, будут равны 0, что крайне логично.

Приходим к QUBO...

Заметим, что мы умеем находить минимальную энергию гамильтониана, а по условиям проблемы необходима максимальная стоимость.

Приходим к QUBO...

Заметим, что мы умеем находить минимальную энергию гамильтониана, а по условиям проблемы необходима максимальная стоимость.

Но если умеем находить минимум, то легко найдём и максимум, рассмотрев $f_{-Q}(x) = -f_Q(x)$. Но как же составить матрицу QUBO под условия задачи?

Приходим к QUBO...

Заметим, что мы умеем находить минимальную энергию гамильтониана, а по условиям проблемы необходима максимальная стоимость.

Но если умеем находить минимум, то легко найдём и максимум, рассмотрев $f_{-Q}(x) = -f_Q(x)$. Но как же составить матрицу QUBO под условия задачи?

Представим, что нет никакого ограничения на вес: необходимо просто максимизировать стоимость. Тогда набираем все предметы. Матрица QUBO примет вид:

$$\begin{cases} Q_{ii} & = & c_i \\ Q_{ij} & = & 0, \; i \neq j \end{cases}$$

Приходим к QUBO...

Теперь хотим уметь ограничивать вес каким-то значением. Неравенств у нас нет. Что же делать? Введём понятия пенальти! То есть если мы набрали больше, чем нужно, то получаем какой-то большой штраф, показывая таким образом, что такой выбор был не оптимален – дальше наша функция будет иметь сильный рост.

Приходим к QUBO...

Но как понять, что мы набрали больше? Введём вспомогательную функцию, которая будет принимать на вход бинарный вектор x.

Приходим к QUBO...

Но как понять, что мы набрали больше? Введём вспомогательную функцию, которая будет принимать на вход бинарный вектор x.

Заметим, что если у нас есть ограничение по весу – W, то можно рассматривать функцию $\rho(x) = \sum\limits_{i=1}^{N} w_i x_i + \sum\limits_{k=1}^{W} k y_k - W$, где y_k – переменные регистра, отличные от x_i.

Приходим к QUBO...

Иными словами, слагаемое $\sum\limits_{k=1}^{W} ky_k$ будет равняться значению, которое необходимо набрать, чтобы получить вес W. Тогда мы сможем сказать следующее: если вес, который был набран некоторым подмножеством предметов $\sum\limits_{i=1}^{N} w_i x_i$, меньше, чем W, то $\sum\limits_{i=1}^{N} w_i x_i + \sum\limits_{k=1}^{W} ky_k = W$, а значит $\rho(x) = 0$, тогда и только тогда, когда $\sum\limits_{i=1}^{N} w_i x_i \leqslant W$, а в остальных случаях $\rho(x) \neq 0$.

Приходим к QUBO...

Иными словами, слагаемое $\sum_{k=1}^{W} k y_k$ будет равняться значению, которое необходимо набрать, чтобы получить вес W. Тогда мы сможем сказать следующее: если вес, который был набран некоторым подмножеством предметов $\sum_{i=1}^{N} w_i x_i$, меньше, чем W, то $\sum_{i=1}^{N} w_i x_i + \sum_{k=1}^{W} k y_k = W$, а значит $\rho(x) = 0$, тогда и только тогда, когда $\sum_{i=1}^{N} w_i x_i \leqslant W$, а в остальных случаях $\rho(x) \neq 0$.

Таким образом, при переполнении рюкзака получаем слишком большое отклонение – пенальти (штраф).

Приходим к QUBO...

Итого, построили такую функцию для пенальти:

$$\rho(x) = \sum_{i=1}^{N} w_i x_i + \sum_{k=1}^{W} k y_k - W.$$

Приходим к QUBO...

Итого, построили такую функцию для пенальти:

$$\rho(x) = \sum_{i=1}^{N} w_i x_i + \sum_{k=1}^{W} k y_k - W.$$

Казалось бы, уже хорошо, но можно лучше: сократим количество слагаемых в вспомогательной функции $\rho(x)$ до $N + \lfloor \log_2 W \rfloor$.

Приходим к QUBO...

Итого, построили такую функцию для пенальти:

$$\rho(x) = \sum_{i=1}^{N} w_i x_i + \sum_{k=1}^{W} k y_k - W.$$

Казалось бы, уже хорошо, но можно лучше: сократим количество слагаемых в вспомогательной функции $\rho(x)$ до $N + \lfloor \log_2 W \rfloor$.

Понимаем, что каждую сумму в диапазоне $1 - W$ можно представить через сумму степеней двоек.

Приходим к QUBO...

Итого, построили такую функцию для пенальти:

$$\rho(x) = \sum_{i=1}^{N} w_i x_i + \sum_{k=1}^{W} k y_k - W.$$

Казалось бы, уже хорошо, но можно лучше: сократим количество слагаемых в вспомогательной функции $\rho(x)$ до $N + \lfloor \log_2 W \rfloor$.

Понимаем, что каждую сумму в диапазоне $1 - W$ можно представить через сумму степеней двоек.

Тогда имеем следующую функцию, где уже представлены переменные регистра y'_k, отличные от y_k и x_i:

$$\rho(x) = \sum_{i=1}^{N} w_i x_i + \sum_{k=0}^{\lfloor \log_2 W \rfloor} 2^k y'_k - W$$

Приходим к QUBO...

Тогда

$$\max\left[f_Q(x) = \sum_{i=1}^{N} c_i x_i - \lambda\rho^2(x)\right] \Leftrightarrow \min\left[f_{-Q(x)} = -\sum_{i=1}^{N} c_i x_i + \lambda\rho^2(x)\right].$$

Приходим к QUBO...

Тогда

$$\max\Big[f_Q(x) = \sum_{i=1}^{N} c_i x_i - \lambda\rho^2(x)\Big] \Leftrightarrow \min\Big[f_{-Q(x)} = -\sum_{i=1}^{N} c_i x_i + \lambda\rho^2(x)\Big].$$

Первое слагаемое берётся со знаком минуса, чтобы при минимизации функции данное слагаемое стремилось к максимуму. А квадрат для вспомогательной функции-пенальти $\rho(x)$ необходим, чтобы отрицательное выражение в скобках делать положительным.

Приходим к QUBO...

Тогда

$$\max\left[f_Q(x) = \sum_{i=1}^{N} c_i x_i - \lambda\rho^2(x)\right] \Leftrightarrow \min\left[f_{-Q(x)} = -\sum_{i=1}^{N} c_i x_i + \lambda\rho^2(x)\right].$$

Первое слагаемое берётся со знаком минуса, чтобы при минимизации функции данное слагаемое стремилось к максимуму. А квадрат для вспомогательной функции-пенальти $\rho(x)$ необходим, чтобы отрицательное выражение в скобках делать положительным.

$\lambda \in \mathbb{R} > 0$ – необходимый коэффициент для увеличения пенальти. На практике можно использовать как $\max_{i=1}^{N} c_i + 1$.

Приходим к QUBO...

Подставим $\rho(x)$ в $-f_Q(x)$:

$$-f_Q(x) = -\sum_{i=1}^{N} c_i x_i + \lambda(\sum_{i=1}^{N} w_i x_i + \sum_{k=0}^{\lfloor \log_2 W \rfloor} 2^k y'_k - W)^2$$

Вообще говоря, по построению y'_k и x_i – имеют разные регистры. Но в сформированной матрице QUBO они, грубо говоря, неразличимы.

Для удобства записи будем считать, что $k = i - N - 1 :\ y'_k = x_i$ и $2^k = w_i\ \forall i > N$.

Эта договорённость сделает форму записи более компактной:

$$-f_Q(x) = -\sum_{i=1}^{N} c_i x_i + \lambda(\sum_{i=1}^{M} w_i x_i - W)^2, \text{ где } M = N + \lfloor \log_2 W \rfloor$$

Приходим к QUBO...

В итоге имеем следующую функцию, минимум которой будем искать:

$$-f_Q(x) = -\sum_{i=1}^{N} c_i x_i + \lambda(\sum_{i=1}^{M} w_i x_i - W)^2, \text{ где } M = N + \lfloor \log_2 W \rfloor$$

Приходим к QUBO...

В итоге имеем следующую функцию, минимум которой будем искать:

$$-f_Q(x) = -\sum_{i=1}^{N} c_i x_i + \lambda(\sum_{i=1}^{M} w_i x_i - W)^2, \text{ где } M = N + \lfloor \log_2 W \rfloor$$

Продолжаем упрощать:

$$(\sum_{i=1}^{M} w_i x_i - W)^2 = \sum_{i=1}^{M} \sum_{j=1}^{M} w_i w_j x_i x_j - 2W \sum_{i=1}^{M} w_i x_i + W^2 =$$

Приходим к QUBO...

В итоге имеем следующую функцию, минимум которой будем искать:

$$-f_Q(x) = -\sum_{i=1}^{N} c_i x_i + \lambda(\sum_{i=1}^{M} w_i x_i - W)^2, \text{ где } M = N + \lfloor \log_2 W \rfloor$$

Продолжаем упрощать:

$$(\sum_{i=1}^{M} w_i x_i - W)^2 = \sum_{i=1}^{M}\sum_{j=1}^{M} w_i w_j x_i x_j - 2W \sum_{i=1}^{M} w_i x_i + W^2 =$$

$$2\sum_{i=1}^{M}\sum_{j>i}^{M} w_i w_j x_i x_j - \sum_{i=1}^{M}(2W - w_i) w_i x_i + W^2$$

Приходим к QUBO...

В итоге имеем следующую функцию, минимум которой будем искать:

$$-f_Q(x) = -\sum_{i=1}^{N} c_i x_i + \lambda(\sum_{i=1}^{M} w_i x_i - W)^2, \text{ где } M = N + \lfloor \log_2 W \rfloor$$

Продолжаем упрощать:

$$(\sum_{i=1}^{M} w_i x_i - W)^2 = \sum_{i=1}^{M} \sum_{j=1}^{M} w_i w_j x_i x_j - 2W \sum_{i=1}^{M} w_i x_i + W^2 =$$

$$2 \sum_{i=1}^{M} \sum_{j>i}^{M} w_i w_j x_i x_j - \sum_{i=1}^{M} (2W - w_i) w_i x_i + W^2$$

Внимательно смотрим... И радуемся победе. Мы смогли получить элементы матрицы:

Приходим к QUBO...

В итоге имеем следующую функцию, минимум которой будем искать:

$$-f_Q(x) = -\sum_{i=1}^{N} c_i x_i + \lambda(\sum_{i=1}^{M} w_i x_i - W)^2, \text{ где } M = N + \lfloor \log_2 W \rfloor$$

Продолжаем упрощать:

$$(\sum_{i=1}^{M} w_i x_i - W)^2 = \sum_{i=1}^{M} \sum_{j=1}^{M} w_i w_j x_i x_j - 2W \sum_{i=1}^{M} w_i x_i + W^2 =$$
$$2 \sum_{i=1}^{M} \sum_{j>i}^{M} w_i w_j x_i x_j - \sum_{i=1}^{M} (2W - w_i) w_i x_i + W^2$$

Внимательно смотрим... И радуемся победе. Мы смогли получить элементы матрицы:

$$\begin{cases} Q_{ii} & = & -c_i - \lambda w_i (2W - w_i) \\ Q_{ij} & = & 2\lambda w_i w_j \end{cases}$$

1D-Multi-Knapsack problem

Формулировка проблемы

Дано N предметов, M рюкзаков, n_i предмет имеет массу $w_i > 0$ и стоимость $v_i > 0$. Необходимо распределить из этих предметов такое подмножество предметов по M рюкзакам, чтобы масса предметов в i-м рюкзаке не превосходила заданной величины c_i (вместимость i-го рюкзака), а суммарная стоимость была максимальна.

Приходим к QUBO...

Перед началом формирования матрицы QUBO введём некоторые переменные:

- $j \in \{0, 1, \ldots, N-1\}$ – нумерация предметов

Приходим к QUBO...

Перед началом формирования матрицы QUBO введём некоторые переменные:

- $j \in \{0, 1, \ldots, N-1\}$ – нумерация предметов
- $w_j \in \mathbb{N}_0$ – веса предметов

Приходим к QUBO...

Перед началом формирования матрицы QUBO введём некоторые переменные:

- $j \in \{0, 1, \ldots, N-1\}$ – нумерация предметов
- $w_j \in \mathbb{N}_0$ – веса предметов
- $v_{i,j} \in \mathbb{N}_0$ – вес предмета j в рюкзаке $i \in \{0, 1, \ldots, M-1\}$

Приходим к QUBO...

Перед началом формирования матрицы QUBO введём некоторые переменные:

- $j \in \{0, 1, \dots, N-1\}$ – нумерация предметов
- $w_j \in \mathbb{N}_0$ – веса предметов
- $v_{i,j} \in \mathbb{N}_0$ – вес предмета j в рюкзаке $i \in \{0, 1, \dots, M-1\}$
- c_i – вместимость i-го рюкзака

Приходим к QUBO...

Перед началом формирования матрицы QUBO введём некоторые переменные:

- $j \in \{0, 1, \ldots, N-1\}$ – нумерация предметов
- $w_j \in \mathbb{N}_0$ – веса предметов
- $v_{i,j} \in \mathbb{N}_0$ – вес предмета j в рюкзаке $i \in \{0, 1, \ldots, M-1\}$
- c_i – вместимость i-го рюкзака
- $x_{i,j}$ – переменная, которая $= 1$ тогда и только тогда, когда j предмет в i рюкзаке

Приходим к QUBO...

Необходимо отслеживать, что предмет находится только в одном рюкзаке, либо ни в одном. Как будем это делать? Введём для этого слагаемое H_{single}

Приходим к QUBO...

Необходимо отслеживать, что предмет находится только в одном рюкзаке, либо ни в одном. Как будем это делать? Введём для этого слагаемое H_{single}

Значит, ввиду введённых ранее переменных, нужно «пройтись» по всем предметам и посмотреть на принадлежность их всем рюкзакам.

Приходим к QUBO...

Необходимо отслеживать, что предмет находится только в одном рюкзаке, либо ни в одном. Как будем это делать? Введём для этого слагаемое H_{single}

Значит, ввиду введённых ранее переменных, нужно «пройтись» по всем предметам и посмотреть на принадлежность их всем рюкзакам.

Так мы можем сделать это: $\sum_{j=0}^{N-1}\left(\sum_{i=0}^{M-1} x_{i,j}\right)$.

Приходим к QUBO...

Необходимо отслеживать, что предмет находится только в одном рюкзаке, либо ни в одном. Как будем это делать? Введём для этого слагаемое H_{single}

Значит, ввиду введённых ранее переменных, нужно «пройтись» по всем предметам и посмотреть на принадлежность их всем рюкзакам.

Так мы можем сделать это: $\sum\limits_{j=0}^{N-1}\left(\sum\limits_{i=0}^{M-1} x_{i,j}\right)$.

Заметим, что довольно легко установить ограничение, что предмет лишь только в одном рюкзаке:

Приходим к QUBO...

Необходимо отслеживать, что предмет находится только в одном рюкзаке, либо ни в одном. Как будем это делать? Введём для этого слагаемое H_{single}

Значит, ввиду введённых ранее переменных, нужно «пройтись» по всем предметам и посмотреть на принадлежность их всем рюкзакам.

Так мы можем сделать это: $\sum_{j=0}^{N-1}\left(\sum_{i=0}^{M-1} x_{i,j}\right)$.

Заметим, что довольно легко установить ограничение, что предмет лишь только в одном рюкзаке:

Ставим ограничение, что количество $\leqslant 1$:

$$H_{single} = \sum_{j=0}^{N-1}\left(\sum_{i=0}^{M-1} x_{i,j}\right)\left(\sum_{i=0}^{M-1} x_{i,j} - 1\right).$$

Приходим к QUBO...

Теперь необходимо задуматься над переполнением рюкзаков – вводим переменную $H_{capacity}$. Но с этим ограничением мы уже разобрались ранее и можно его перенести на каждый рюкзак по отдельности:

Приходим к QUBO...

Теперь необходимо задуматься над переполнением рюкзаков – вводим переменную $H_{capacity}$. Но с этим ограничением мы уже разобрались ранее и можно его перенести на каждый рюкзак по отдельности:

Для начала введём дополнительные переменные регистра $y_{i,b}$, которые соответствуют i-му рюкзаку соответственно.

Приходим к QUBO...

Теперь необходимо задуматься над переполнением рюкзаков – вводим переменную $H_{capacity}$. Но с этим ограничением мы уже разобрались ранее и можно его перенести на каждый рюкзак по отдельности:

Для начала введём дополнительные переменные регистра $y_{i,b}$, которые соответствуют i-му рюкзаку соответственно.

$$H_{capacity} = \sum_{i=0}^{M-1} \left[\left(\sum_{j=0}^{N-1} w_j \cdot x_{i,j} \right) + \left(\sum_{b=0}^{\lfloor \log_2 c_i \rfloor} 2^b \cdot y_{i,b} \right) - c_i \right]^2$$

Приходим к QUBO...

Осталось добавить последнее слагаемое – H_{obj}, в котором будем подсчитывать суммарную стоимость взятых предметов:

Приходим к QUBO...

Осталось добавить последнее слагаемое – H_{obj}, в котором будем подсчитывать суммарную стоимость взятых предметов:

$$H_{obj} = -\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} v_{i,j} \cdot x_{i,j}$$

Приходим к QUBO...

В итоге, каждое слагаемое, из рассмотренных нами, является значимым – получаем общую функцию вида, дополнительно введя коэффициенты пенальти – $A = 1,\ B = C = \max\ v_{i,j} + 1$:

Приходим к QUBO...

В итоге, каждое слагаемое, из рассмотренных нами, является значимым – получаем общую функцию вида, дополнительно введя коэффициенты пенальти – $A = 1,\ B = C = \max\ v_{i,j} + 1$:

$$H = A \cdot H_{obj} + B \cdot H_{capacity} + C \cdot H_{single}$$

Приходим к QUBO...

В итоге, каждое слагаемое, из рассмотренных нами, является значимым – получаем общую функцию вида, дополнительно введя коэффициенты пенальти – $A = 1,\ B = C = \max\ v_{i,j} + 1$:

$$H = A \cdot H_{obj} + B \cdot H_{capacity} + C \cdot H_{single}$$

Или, раскрыв все слагаемые:

Приходим к QUBO...

В итоге, каждое слагаемое, из рассмотренных нами, является значимым – получаем общую функцию вида, дополнительно введя коэффициенты пенальти – $A = 1,\ B = C = \max\ v_{i,j} + 1$:

$$H = A \cdot H_{obj} + B \cdot H_{capacity} + C \cdot H_{single}$$

Или, раскрыв все слагаемые:

$$\begin{aligned} H = & -A \cdot \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} v_{i,j} \cdot x_{i,j} + \\ & + B \cdot \sum_{i=0}^{M-1} \left[\left(\sum_{j=0}^{N-1} w_j \cdot x_{i,j} \right) + \left(\sum_{b=0}^{\lfloor \log_2 c_i \rfloor} 2^b \cdot y_{i,b} \right) - c_i \right]^2 + \\ & + C \cdot \sum_{j=0}^{N-1} \left(\sum_{i=0}^{M-1} x_{i,j} \right) \left(\sum_{i=0}^{M-1} x_{i,j} - 1 \right) \end{aligned}$$

Приходим к QUBO...

Начнём преобразовывать выражение с 3-го слагаемого:

Приходим к QUBO...

Начнём преобразовывать выражение с 3-го слагаемого:

$$H_{single} = \sum_{j=0}^{N-1} \left[\sum_{i=0}^{M-1} x_{i,j} \sum_{i=0}^{M-1} x_{i,j} - \sum_{i=0}^{M-1} x_{i,j} \right] =$$

Приходим к QUBO...

Начнём преобразовывать выражение с 3-го слагаемого:

$$H_{single} = \sum_{j=0}^{N-1}\left[\sum_{i=0}^{M-1} x_{i,j} \sum_{i=0}^{M-1} x_{i,j} - \sum_{i=0}^{M-1} x_{i,j}\right] =$$

$$= \sum_{j=0}^{N-1}\left[\sum_{p=0}^{M-1} \sum_{i=0}^{M-1} x_{i,j} x_{p,j} - \sum_{i=0}^{M-1} x_{i,j}\right] =$$

Приходим к QUBO...

Начнём преобразовывать выражение с 3-го слагаемого:

$$H_{single} = \sum_{j=0}^{N-1} \left[\sum_{i=0}^{M-1} x_{i,j} \sum_{i=0}^{M-1} x_{i,j} - \sum_{i=0}^{M-1} x_{i,j} \right] =$$

$$= \sum_{j=0}^{N-1} \left[\sum_{p=0}^{M-1} \sum_{i=0}^{M-1} x_{i,j} x_{p,j} - \sum_{i=0}^{M-1} x_{i,j} \right] =$$

$$= \sum_{j=0}^{N-1} \sum_{p=0}^{M-1} \sum_{i=0}^{M-1} x_{i,j} x_{p,j} - \sum_{j=0}^{N-1} \sum_{i=0}^{M-1} x_{i,j}$$

Приходим к QUBO...

Теперь приступим ко второму слагаемому:

Приходим к QUBO...

Теперь приступим ко второму слагаемому:
Аналогично замене в рассуждениях для 1D проблемы, перепишем в более компактном виде $H_{capacity}$:

Приходим к QUBO...

Теперь приступим ко второму слагаемому:
Аналогично замене в рассуждениях для 1D проблемы, перепишем в более компактном виде $H_{capacity}$:

$$H_{capacity} = \sum_{i=0}^{M-1} \left[\left(\sum_{j=0}^{N-1} w_j \cdot x_{i,j} \right) + \left(\sum_{b=0}^{\lfloor \log_2 c_i \rfloor} 2^b \cdot y_{i,b} \right) - c_i \right]^2 =$$

Приходим к QUBO...

Теперь приступим ко второму слагаемому:
Аналогично замене в рассуждениях для 1D проблемы, перепишем в более компактном виде $H_{capacity}$:

$$H_{capacity} = \sum_{i=0}^{M-1} \left[\left(\sum_{j=0}^{N-1} w_j \cdot x_{i,j} \right) + \left(\sum_{b=0}^{\lfloor \log_2 c_i \rfloor} 2^b \cdot y_{i,b} \right) - c_i \right]^2 =$$

$$= \sum_{i=0}^{M-1} \left[\left(\sum_{j=0}^{L} w_j \cdot x_{i,j} \right) - c_i \right]^2 L = N + \lfloor \log_2 c_i \rfloor$$

Приходим к QUBO...

$$H_{capacity} = \sum_{i=0}^{M-1}\left[\sum_{j=0}^{L}\sum_{p=0}^{L} w_j w_p x_{i,j} x_{i,p} - 2c_i \sum_{j=0}^{L} w_j x_{i,j} - c_i^2\right] =$$

Приходим к QUBO...

$$H_{capacity} = \sum_{i=0}^{M-1}\left[\sum_{j=0}^{L}\sum_{p=0}^{L} w_j w_p x_{i,j} x_{i,p} - 2c_i \sum_{j=0}^{L} w_j x_{i,j} - c_i^2\right] =$$

$$= \sum_{i=0}^{M-1}\sum_{j=0}^{L}\sum_{p=0}^{L} w_j w_p x_{i,j} x_{i,p} - 2c_i \sum_{i=0}^{M-1}\sum_{j=0}^{L} w_j x_{i,j} - \sum_{i=0}^{M-1} c_i^2$$

Приходим к QUBO...

Теперь осталось определить элементы матрицы. Это делается достаточно просто. Главное – не забывать, что $y_{i,b}$ и $x_{i,j}$ – разные переменные регистра и их нельзя совмещать. Каждому рюкзаку соответствует свой набор переменных регистра, и они – не пересекающиеся, что не имело бы смысла в противном случае

Приходим к QUBO...

В данном случае будет приведён алгоритм заполнения матрицы, а не конкретные формулы для каждого элемента.

Приходим к QUBO...

В данном случае будет приведён алгоритм заполнения матрицы, а не конкретные формулы для каждого элемента.

Начнём с заполнения диагональных элементов:

Приходим к QUBO...

В данном случае будет приведён алгоритм заполнения матрицы, а не конкретные формулы для каждого элемента.

Начнём с заполнения диагональных элементов:

Необходимо будет перебрать все рюкзаки – i – и лежащие в них предметы – j.

Приходим к QUBO...

В данном случае будет приведён алгоритм заполнения матрицы, а не конкретные формулы для каждого элемента.

Начнём с заполнения диагональных элементов:

Необходимо будет перебрать все рюкзаки – i – и лежащие в них предметы – j.

Тогда диагональные элементы, отвечающие переменным $x_{i,j}$ можно определить по формуле:

Приходим к QUBO...

В данном случае будет приведён алгоритм заполнения матрицы, а не конкретные формулы для каждого элемента.

Начнём с заполнения диагональных элементов:

Необходимо будет перебрать все рюкзаки – i – и лежащие в них предметы – j.

Тогда диагональные элементы, отвечающие переменным $x_{i,j}$ можно определить по формуле:

$$Q_{rr} = Bw_j^2 - 2Bc_i w_j - Av_{i,j}, \text{ где } r = iN + \sum_{k=0}^{i} \lfloor \log_2 c_k + 1 \rfloor + j$$

Приходим к QUBO...

В данном случае будет приведён алгоритм заполнения матрицы, а не конкретные формулы для каждого элемента.

Начнём с заполнения диагональных элементов:

Необходимо будет перебрать все рюкзаки – i – и лежащие в них предметы – j.

Тогда диагональные элементы, отвечающие переменным $x_{i,j}$ можно определить по формуле:

$$Q_{rr} = Bw_j^2 - 2Bc_iw_j - Av_{i,j}, \text{ где } r = iN + \sum_{k=0}^{i} \lfloor \log_2 c_k + 1 \rfloor + j$$

А переменным $y_{i,b}$:

Приходим к QUBO...

В данном случае будет приведён алгоритм заполнения матрицы, а не конкретные формулы для каждого элемента.

Начнём с заполнения диагональных элементов:

Необходимо будет перебрать все рюкзаки – i – и лежащие в них предметы – j.

Тогда диагональные элементы, отвечающие переменным $x_{i,j}$ можно определить по формуле:

$$Q_{rr} = Bw_j^2 - 2Bc_i w_j - Av_{i,j}, \text{ где } r = iN + \sum_{k=0}^{i} \lfloor \log_2 c_k + 1 \rfloor + j$$

А переменным $y_{i,b}$:

$$Q_{rr} = 4^b B - 2^{b+1} Bc_i, \text{ где } r = (i+1)N + \sum_{k=0}^{i} \lfloor \log_2 c_k + 1 \rfloor + b$$

Приходим к QUBO...

Самая интересная часть – определение оставшихся элементов матрицы:

Приходим к QUBO...

Самая интересная часть – определение оставшихся элементов матрицы:

Определяем пенальти для $x_{i,j}$ и $y_{i,b}$ Перебираем все рюкзаки – i, все предметы – j, все дополнительные биты – b:

Приходим к QUBO...

Самая интересная часть – определение оставшихся элементов матрицы:

Определяем пенальти для $x_{i,j}$ и $y_{i,b}$ Перебираем все рюкзаки – i, все предметы – j, все дополнительные биты – b:

$$Q_{r_1 r_2} = 2^{b+1} B w_j \quad r_1 = iN + \sum_{k=0}^{i} \lfloor \log_2 c_k + 1 \rfloor + j$$

$$r_2 = r_1 + b - j + N$$

Приходим в QUBO...

Теперь ставим пенальти на элемент $x_{i,j}$ для нахождения только в одном рюкзаке:

Приходим в QUBO...

Теперь ставим пенальти на элемент $x_{i,j}$ для нахождения только в одном рюкзаке:

Перебираем все пары рюкзаков (m, n) и предметы j:

Приходим в QUBO...

Теперь ставим пенальти на элемент $x_{i,j}$ для нахождения только в одном рюкзаке:

Перебираем все пары рюкзаков (m, n) и предметы j:

$$Q_{r_1 r_2} = 2C \quad r_1 = mN + \sum_{k=0}^{m} \lfloor \log_2 c_k + 1 \rfloor + j$$

$$r_2 = nN + \sum_{k=0}^{n} \lfloor \log_2 c_k + 1 \rfloor + j$$

Приходим в QUBO...

Установим ограничения для $x_{i,j}$ на переполнение рюкзака-i

Приходим в QUBO...

Установим ограничения для $x_{i,j}$ на переполнение рюкзака-i

Перебираем все пары предметов (m, n):

Приходим в QUBO...

Установим ограничения для $x_{i,j}$ на переполнение рюкзака-i

Перебираем все пары предметов (m, n):

$$Q_{r_1 r_2} = 2Bw_n w_m \quad r_1 = iN + \sum_{k=0}^{i} \lfloor \log_2 c_k + 1 \rfloor + m$$

$$r_2 = iN + \sum_{k=0}^{i} \lfloor \log_2 c_k + 1 \rfloor + n$$

Приходим в QUBO...

Установим ограничения $y_{i,b}$ на переполнения рюкзака-i

Приходим в QUBO...

Установим ограничения $y_{i,b}$ на переполнения рюкзака-i

Перебираем все пары дополнительных битов (m, n):

Приходим в QUBO...

Установим ограничения $y_{i,b}$ на переполнения рюкзака-i

Перебираем все пары дополнительных битов (m, n):

$$Q_{r_1 r_2} = 2^{m+n+1} B \quad r_1 = (i+1)N + \sum_{k=0}^{i} \lfloor \log_2 c_k + 1 \rfloor + m$$

$$r_2 = (i+1)N + \sum_{k=0}^{i} \lfloor \log_2 c_k + 1 \rfloor + n$$

Приходим в QUBO...

Нетрудно видеть, что все интервалы заполнения непересекающиеся, поэтому получаем корректное заполнение матрицы QUBO с учётом всех ограничений и независимости переменных регистра x и y.

1D-Multi Constraints

Формулировка проблемы

Дано N задач, n_i задача имеет потребление памяти $w_i > 0$, стоимость выполнения $c_i > 0$ и требует ядер $p_i > 0$. Необходимо выбрать такое подмножество задач, чтобы их суммарное потребление памяти не превосходило W, а количество ядер – P, при этом суммарная стоимость была максимальна.

Конец