1 Введение

1.1 Постановка задачи

Задача коммиворяжера. Дан граф $G(V, E)$ с множеством вершин V и рёбер между ними E. Каждое ребро обладает своей длинной. В дальнейшем, расстояние между ребрами i и j будет обозначать как $d(i, j)$. Также скажем, что $|V| = N$ (то есть в графе N вершин).

Путём размера n будем называть вектор вида $\vec{W} = (v^1, v^2, \cdots, v^{n-1}, v^n)$ (причём для каждой пары вершин вида (v^i, v^{i+1}) найдётся соответствующее ребро). В дальнейшем, длину пути будем обозначать как $L\left(\vec{W}\right)$. Очевидно, что она равна сумме длин рёбер между каждой парой вершин, то есть:

$$L\left(\vec{W}\right) = \sum_{i=1}^{l-1} d(v^i, v^{i+1}) \tag{1}$$

Замкнутым путём размера n будем называть путь размера n такой, что его первая и последняя вершины совпадают, а также каждая вершина (кроме первой) встречается в нём ровно 1 раз. Очевидно, что в таком пути будет $n - 1$ разных вершин.

Решением задачи коммивояжера будем называть такой замкнутый путь $\vec{W}_{res}$ длины $N + 1$, что его длина наименьшая среди всех путей длины $N + 1$, то есть $\forall \vec{W} \in \mathbb{W}^{N+1} : L\left(\vec{W}_{res}\right) \leq L\left(\vec{W}\right)$ (здесь $\mathbb{W}^{N+1}$ - множество всех замкнутых путей размера $N + 1$).

1.2 Идея решения

Предположим, что по данному графу $G(V, E)$ была построена матрица расстояний $M_{N\times N}$. Для данной матрицы верно, что:

$$M_{i,j} = \begin{cases} d(i,j) & , \quad (i,j) \in E \\ 0 & , \quad \text{иначе} \end{cases} \tag{2}$$

Комбинируя формулы (1) и (2), получаем, что длина замкнутого пути будет вычисляться по формуле:

$$L\left(\vec{W}\right) = \sum_{i=1}^{N} M_{v^i, v^{i+1}} \tag{3}$$

Легко заметить, что при $i = N$ мы пытаемся обратиться к несуществующей вершине v^{N+1}, а значит и к несуществующему элементу матрицы $M_{v^N,v^{N+1}}$. Чтобы этого избежать, договоримся, что с точки зрения индексов, индекс $N+1$ тождественен индексу 1.

Так как решение задачи это путь вида $\vec{W} = (v^1, v^2, \cdots, v^{N-1}, v^N, v^1)$, разобьём его на конкретные рёбра по итерациям. То есть в данном случае, на первой итерации мы пойдём по ребру (v^1, v^2), на второй итерации - по ребру (v^2, v^3) и т.д.. Заметим, что всего таких итераций будет ровно N. Построим матрицу $X_{N\times N}$ такую, что $X_{i,t} \in \{0, 1\}$, причем значение 1 будет только тогда, когда на итерации t мы выходим из вершины i. Легко заметить, что каждый замкнутый путь $\vec{W}$ можно записать в таком виде, причём единственным образом, и тоже самое верно в обратную сторону. То есть мы нашли преобразование $\mathcal{F}$, для которого верно, что $\mathcal{F} : \mathbb{W}^{N+1} \to X_{N\times N}$ и $\mathcal{F}^{-1} : X_{N\times N} \to \mathbb{W}^{N+1}$. О том, как оно работает, будет сказано далее.

2 Формирование целевой функции

2.1 Основа целевой функции

Пусть нам уже дана некая матрица $X_{N\times N}$, соответствующая какому-то замкнутому пути $\vec{W}$. Посчитаем по ней длину пути. Исходя из формулы (3), имеем:

$$L\left(\vec{W}\right) = \sum_{i=1}^{N} M_{v^i,v^{i+1}}$$

Теперь, так как мы знаем устройство матрица X, можем сказать, что вершина (i, j) будет в нашем пути, если $\exists t : X_{i,t} = X_{j,t+1} = 1$. Отсюда можно сказать, что:

$$M_{i,j} = \sum_{t=1}^{N} M_{i,j} X_{i,t} X_{j,t+1} \tag{4}$$

Теперь легко заметить, что если мы просуммируем выражение (4) по всем i и j, то это значит, что мы проверяем каждое ребро на то, есть ли оно в нашем пути и, если оно есть, включаем его в общую сумму. Отсюда имеем:

$$f(X) = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{t=1}^{N} M_{i,j} X_{i,t} X_{j,t+1}$$

Здесь $f(X)$ - *целевая функция* для матрицы X. Теперь задачу можно сформулировать так: найти такую матрицу X, чтобы выражение $f(X)$ достигало своего минимума.

Также, для простоты, введем следующее обозначение:

$$\sum_{i_1,i_2,\cdots,i_n} A_{i_1,i_2,\cdots,i_n} = \sum_{i_1=1}^{N} \sum_{i_2=1}^{N} \cdots \sum_{i_n=1}^{N} A_{i_1,i_2,\cdots,i_n} \tag{5}$$

Отсюда целеыая функция примет вид:

$$f(X) = \sum_{i,j,t} M_{i,j} X_{i,t} X_{j,t+1} \tag{6}$$

2.2 Штрафы

Очевидно, что так как мы не знаем итоговый путь $\vec{W}$, то и не знаем матрицу X, а значит нет гарантий, что она будет удовлетворять условиям того, что каждое ребро будет включено ровно один раз. Более того, легко заметить, что если матрица X будет полностью заполнена нулями, то $f(X) = 0$. Это будет, очевидно, самым минимальным путём, но такой вариант рассматривать нельзя. Поэтому введём систему штрафов.

Штрафом будем называть величину, на которую увеличивается значение целевой функции при нарушении условий задачи. В случае задачи коммивояжера штрафы будут, если в нашем пути:

1. Одна из вершина графа не встречается ни разу, или встречается более 1 раза (кроме последней)

2. Какое-то ребро встречается более одного раза

3. Определённость старта итерации

Разберёмся с каждым штрафом по порядку. Чтобы проверить, все ли вершины мы прошли, сделаем следующее: просуммируем все столбцы матрицы X как вектора-столбцы. В итоге мы должны получить вектор-столбец Y такой, что:

$$Y_i = \sum_{j=1}^{N} X_{i,j}$$

Очевидно, что если каждая вершина встречается ровно один раз, то в каждой строке найдётся ровно одна единица, поэтому вектор-столбец Y обязан быть заполнен единицами. Чтобы это проверить, вычтем из каждого элемента данного столбца единицу, то есть:

$$Y_i - 1 = \sum_{j=1}^{N} X_{i,j} - 1$$

А теперь просуммируем по всем строкам данную величину, то есть получим:

$$\sum_{i=1}^{N} (Y_j - 1) = \sum_{i=1}^{N} \left(\sum_{j=1}^{N} X_{i,j} - 1 \right)$$

Таким образом, если какая-то вершина встречается более одного раза, данное выражение будет больше нуля. Но стоит отметить, что если все вершины отсутствуют, то данное выражение будет отрицательным. Чтобы этого избежать, можно либо добавить модуль, либо возвести в квадрат. Разберём конкретный пример - пусть все вершины встречаются по одному разу, но какая-то одна встречается k раз ($k \in \overline{0, N}$). В таком случае данное выражение примет значение $k - 1$. Если она выбивается от условий только на единицу (то есть её либо нет, либо она встречается два раза), то значение данного выражения что с модулем, что без даст 1. Если же она встречается больше раз, то с модулем будет просто $k-1$, а с квадратом $(k+1)^2$ соответственно. Так как мы «максимально не хотим» учитывать вершину несколько раз, положим квадрат. Отсюда итоговый штраф будет вычисляться как:

$$\sum_{i=1}^{N} \left(\sum_{j=1}^{N} X_{i,j} - 1 \right)^2 = \sum_{i} \left(\sum_{j} X_{i,j} - 1 \right)^2 \tag{7}$$

Добавив его к целевой функции, получаем:

$$f(X) = \sum_{i,j,t} M_{i,j} X_{i,t} X_{j,t+1} + \sum_{i} \left(\sum_{j} X_{i,j} - 1 \right)^2 \tag{8}$$

Теперь проверим, все ли ребра встречаются не более одного раза. Легко заметить, что так как ребро - это пара вершин, то оно будет встречаться ещё раз тогда и только тогда, когда хотя бы одна из двух вершин повторяется. Поэтому данный случай формально мы уже рассмотрели.

Остался последний - определённость старта итераций. Это значит, что на каждой итерации первая вершина ребра определена, причём однозначно. С точки зрения матрицы это значит, что в каждом столбце должна быть ровно одна единица. Данный случай почти полностью совпадает с предыдущим штрафом, только сначала мы складываем строки, а потом уже столбцы, то есть штраф примет вид:

$$\sum_j \left(\sum_i X_{i,j} - 1 \right)^2 \tag{9}$$

Добавив его к целевой функции, получаем:

$$f(X) = \sum_{i,j,t} M_{i,j} X_{i,t} X_{j,t+1} +$$

$$+ \sum_i \left(\sum_j X_{i,j} - 1 \right)^2 + \sum_j \left(\sum_i X_{i,j} - 1 \right)^2 \tag{10}$$

Таким образом мы создали нашу целевую функцию. Её минимизация справляется и с условиями задачи коммивояжера, и с поиском кратчайшего пути.

2.3 Веса

На данный момент целевая функция минимизирует штрафы, а также значение функции. Но до этого момента мы не говорили о порядках. Рассмотрим конкретную матрицу расстояний:

$$M_1 = \begin{pmatrix} 0 & 0.1 & 0.01 & 0.01 \\ 0.1 & 0 & 0.01 & 0.05 \\ 0.01 & 0.01 & 0 & 0.1 \\ 0.01 & 0.05 & 0.1 & 0 \end{pmatrix}$$

Если провести простой перебор данной задачи по целевой функции (10), получим следующую матрицу X.

$$X_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Теперь сделаем тоже самое, но возьмем все значения нашей матрицы и умножим на 100, то есть:

$$M_2 = 100M_1 = \begin{pmatrix} 0 & 10 & 1 & 1 \\ 10 & 0 & 1 & 5 \\ 1 & 1 & 0 & 10 \\ 1 & 5 & 10 & 0 \end{pmatrix}$$

Если провести простой перебор данной задачи по целевой функции (10), получим следующую матрицу X.

$$X_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Заметим, что матрицы M_1 и M_2 связаны линейно между собой, то есть мы увеличили все длины рёбер в одно и тоже число раз. Логично, что в таком случае результат не должен поменяться, поскольку минимальное значение пути также измениться в тоже число раз, но наш алгоритм дал осечку. Это связано с взаимосвязью порядка основной части функции (непосредственно минимизация длины пути) и порядка штрафов. В случае матрицы M_1 основная часть целевой функции мала. В частности, максимальное значение основной части - если матрица X полностью состоит из единиц и будет равно:

$$f(X) = \sum_{i,j,t} M_{i,j} =$$

$$= 4 \sum_{i,j} M_{i,j} = 2.24$$

В тоже время минимальное значение штрафа (хотя бы одно из них) равно 1. А если сумма штрафов больше, то и подавно. То есть в первом случае доля штрафов в целевой функции сильно больше доли основной части.

Если же провести такой же анализ во втором случае, то там максимальное значение основной части будет уже $224 \sim 15^2$. То есть штрафов должно быть очень много, чтобы нивелировать основную часть.

Отсюда вывод - штрафы должны быть подобраны правильным образом под каждую задачу. Самый простой способ это сделать - добавить множители или *веса* перед ними, то есть:

$$f(X) = \sum_{i,j,t} M_{i,j} X_{i,t} X_{j,t+1} +$$

$$+C_1 \sum_i \left(\sum_j X_{i,j} - 1 \right)^2 + C_2 \sum_j \left(\sum_i X_{i,j} - 1 \right)^2 \tag{11}$$

, где C_1 и C_2 - веса, тем большие, чем больше мы хотим избавиться от того или иного штрафа. В общем случае нас мало интересует, какой штраф будет, поэтому можно сказать, что $C_1 = C_2 = C$, то есть:

$$f(X) = \sum_{i,j,t} M_{i,j} X_{i,t} X_{j,t+1} +$$

$$+C \sum_i \left(\sum_j X_{i,j} - 1 \right)^2 + C \sum_j \left(\sum_i X_{i,j} - 1 \right)^2 \tag{12}$$

Теперь оценим само значение данной константы. Как видно, оно зависит от порядка длин расстояний между вершинами в графе. В целом можно положить эту константу бесконечно большим (или максимально допустимыми числами в том или ином языке программирования, где реализуется данный метод). Но такой путь не всегда выгоден с точки зрения нагрузки на вычислительную технику, поэтому можно положить её, как вариант, $C = N^2 \max_{i,j \in \overline{1,N}} M_{i,j}$, где $m \in \overline{1,3}$. Данная оценка эмпирическая и означает, своего рода, верхнюю оценку суммы всех расстояний матрицы M). Также стоит отметить, что в выражении (12) можно пойти по иному пути - не увеличивать штрафы, а уменьшить основную часть целевой функции. Фактически это равносильно домножению на константу, что, очевидно, никак не повлияет на минимум. Таким образом выражение (12) примет вид:

$$f(X) = \frac{1}{C} \sum_{i,j,t} M_{i,j} X_{i,t} X_{j,t+1} +$$

$$+ \sum_i \left(\sum_j X_{i,j} - 1 \right)^2 + \sum_j \left(\sum_i X_{i,j} - 1 \right)^2 \tag{13}$$

3 QUBO-матрица

3.1 Определение

Квадратичная неограниченная бинарная оптимизация (QUBO) - задача комбинаторной оптимизации. Она задаётся следующим образом:

Пусть дана функция вида:

$$f(\vec{x}) = \vec{x}^{\mathrm{T}} Q \vec{x} = \sum_{i,j} Q_{i,j} x_i x_j \tag{14}$$

Найти вектор $\vec{x}_{res}$ такой, что $\forall \vec{x} : f(\vec{x}_{res}) \leq f(\vec{x})$.

3.2 Формирование матрицы QUBO

Для того, чтобы получить матрицу Q, расширим вводные данные. Пусть вместо вектора $\vec{x}$ на вход подаётся матрица X. Тогда матрица Q станет своего рода подобием тензора, то есть будет обладать 4-мя индексами: $Q \equiv Q_{i,j}^{k,m}$. С точки зрения реализации в компьютере, матрицу Q можно воспринимать как обычный словарь. В обычном случае, ключём является пара индексов (i, j), но в нашем расширенном случае, ключами будет пара пар индексов, то есть $[(i, j), (k, m)]$. Таким образом выражение (14) примет вид:

$$\begin{aligned} f(X) &= \sum_{(i,j)} \sum_{(k,m)} Q_{i,j}^{k,m} x_{i,j} x_{k,m} = \\ &= \sum_{i,j,k,m} Q_{i,j}^{k,m} x_{i,j} x_{k,m} \end{aligned} \tag{15}$$

Попробуем привести выражение (13) к данному виду.

$$f(X) = \frac{1}{C} \sum_{i,j,t} M_{i,j} X_{i,t} X_{j,t+1} +$$

$$+ \sum_{i} \left(\sum_{j} X_{i,j} - 1 \right)^2 + \sum_{j} \left(\sum_{i} X_{i,j} - 1 \right)^2$$

Преобразуем первое слогаемое:

$$\frac{1}{C} \sum_{i,j,t} M_{i,j} X_{i,t} X_{j,t+1} \equiv \frac{1}{C} \sum_{i,j,k} M_{i,k} X_{i,j} X_{k,j+1}$$

Мы хотим представить данное выражение в виде матрицы QUBO, то есть:

$$\frac{1}{C}\sum_{i,j,k} M_{i,k}X_{i,j}X_{k,j+1} = \sum_{i,j,k,m} Q_{i,j}^{k,m}X_{i,j}X_{k,m} =$$

$$= \sum_{i,j,k} Q_{i,j}^{k,j+1}X_{i,j}X_{k,j+1} + \sum_{i,j,k}\sum_{m\neq j+1} Q_{i,j}^{k,m}X_{i,j}X_{k,m} \Longleftrightarrow$$

$$\Longleftrightarrow \sum_{i,j,k}\left(\frac{1}{C}M_{i,k} - Q_{i,j}^{k,j+1}\right)X_{i,j}X_{k,j+1} = \sum_{i,j,k}\sum_{m\neq j+1} Q_{i,j}^{k,m}X_{i,j}X_{k,m}$$

Заметим, что мы выбираем матрицу Q произвольно, а значит положим:

$$\begin{cases} \frac{1}{C}M_{i,k} - Q_{i,j}^{k,j+1} &= 0 \\ Q_{i,j}^{k,m} &= 0 \end{cases}$$

В таком случае вне зависимости от выбора матрицы X, данное преобразование будет работать. Таким образом получаем, что матрица Q для первого слагаемого имеет вид:

$$Q_{i,j}^{k,m} = \begin{cases} \frac{M_{i,k}}{C} & , \quad m = j+1 \\ 0 & , \quad \text{иначе} \end{cases} \tag{16}$$

Преобразуем второе слогаемое:

$$\sum_i \left(\sum_j X_{i,j} - 1\right)^2 = \sum_i \left(\left(\sum_j X_{i,j}\right)^2 - 2\sum_j X_{i,j} + 1\right) =$$

$$= \sum_i \left(\sum_j\sum_k X_{i,j}X_{i,k} - 2\sum_j X_{i,j} + 1\right) =$$

$$= \left(\sum_{i,j,k} X_{i,j}X_{i,k} - 2\sum_{i,j} X_{i,j} + N\right) =$$

$$= \sum_{i,j,k}\left(X_{i,k} - \frac{2}{N}\right)X_{i,j} + N$$

Очевидно, что слогаемое N не повлияет на минимизацию, поэтому её можно игнорировать. Также отметим, что в силу большого набора данных N, выражение $\frac{2}{N}$ очень мало, а значит можно сказать, что:

$$\sum_{i,j,k} \left(X_{i,k} - \frac{2}{N} \right) X_{i,j} \approx \sum_{i,j,k} X_{i,j}^2$$

Мы хотим представить данное выражение в виде матрицы QUBO, то есть:

$$\sum_{i,j,k} X_{i,j}^2 = \sum_{i,j,k,m} Q_{i,j}^{k,m} X_{i,j} X_{k,m} \iff$$

$$\iff \frac{1}{N} \sum_{i,j,k,m} X_{i,j}^2 = \sum_{i,j,k,m} Q_{i,j}^{k,m} X_{i,j} X_{k,m} \iff$$

Теперь, абсолютно теже рассуждения приводят к аналогичному выражению для третьего слогаемого в целевой функции:

Таким образом получаем, что они эквивалентны. Рассмотрим шаблон

$$\frac{1}{N} \sum_{i,j,k,m} X_{i,j}^2 = \sum_{i,j,k,m} Q_{i,j}^{k,m} X_{i,j} X_{k,m}$$