Министерство образования и науки Российской Федерации
Новосибирский государственный университет

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА ДИСКРЕТНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Курсовая работа

Программная реализация решения многомерной транспортной задачи с помощью библиотеки ORTools

Пинкаля Кирилл Владимирович

Научный руководитель канд. физ.-мат. наук., Новиков А.А.

Новосибирск 2023

Содержание

1 Аннотация

В данной работе мы поставили и решили задачу линейного целочисленного программирования, интерпретируемую как оптимизация складски запасов: построили математическую модель, определили необходимые переменные и ограничения, а также имплементировали программную реализацию для этого класса задач.

Существенной частью работы является использование в целях программной реализации решятеля задачи библиотеку методов оптимзации ORTools.[1]

Полный листинг программной реализации доступен по ссылке https://github.com/SIPLIbY/Monge-Kantorovich-s-problem/.[7]

2 Актуальность работы

В ходе реализации экономической деятельности по доставке продуктов от производства до потребителей возникает ряд логистических задач, связанных с минимизацией инздержек на доставку. [6] Классическими результатами в этой области считается решение транспортной задачи поставленной Монжем в 1781 году[8] и решенной Канторовичем в 1942 [9]. (Нобелевская премия по экономике 1975 года [10]). Многомерный аналог данной задачи обычно назыается задачей Данцига [11].

Проблема управления запасами в складских терминалах включает в себя несколько проблем: требования потребителей, распределение продукции между складскими помещениями в целях снижения стоимости доставки, изготовления и хранения товара и т.д. Задачи этого типа представляют собой семейство дискретных комбинаторных задач и в реальных ситуациях имеют большую размерность с экспоненциально растущей мощностью комбинаций, и поэтому не могут быть оптимально решены обычными способами. Поэтому для решения необходима реализация специальных алгоритмов оптимизации. В этой работе мы рассмотрим реализацию метода MIP (смешанного целочисленного программирования).[12] Задача, которую мы решаем, интерпретируется как задача оптимизации запасов в нескольких складских терминалах с ограниченной вместимостью и перевозки этих запасов в торговые пункты.

3 Введение

Задача Монжа-Канторовича

[5]Однородный груз сосредоточен у m поставщиков в объемах $a_1, a_2, ...a_m$. Данный груз необходимо доставить n потребителям в объемах $b_1, b_2, ...b_n$. Пусть c_{ij} - стоимость перевозки груза от поставщика i до потребителя j. Требуется составить план перевозок, позволяющий полностью вывезти продукты всех производителей, полностью обеспечивающий потребности всех потребителей и дающий минимум суммарных затрат на перевозку. Обозначим как x_{ij} объёмы перевозок от поставщика i до потребителя j

$$\sum_{j=1}^{n} x_{ij} = a_i,\ i = 1, 2, ..., m \tag{1}$$

$$\sum_{i=1}^{m} x_{ij} = b_j,\ j = 1, 2, ..., n \tag{2}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \to \min \tag{3}$$

3.1 Дополнительные ограничения

Данная постановка задачи является классической, но не совсем пригодной для реального использования, ведь на практике, к примеру, есть такие условия как вместимость склада, цены на производство, зависящие от местоположения завода, и многие другие условия, которые надо учитывать. Тем не менее классическая модель даёт понимание задачи, а также наталкивает на некоторые идеи и методы решения транспортных задач.

Теперь, имея понимание, можно сформулировать конкретно нашу задачу, а так же найти оптимальный план перевозок с помощью компьютерных вычислений, который будет удовлетворять гораздо большему количеству условий, нежели в задаче Монжа-Канторовича.

Например в нашей постановке фигурируют следующие ограничения:

$$x_{i,j,k} \in \mathbb{N} \cup \{0\} \qquad \forall i, j, k \tag{4}$$

$$y_{i,r} \in \mathbb{N} \cup \{0\} \qquad \forall i, r \tag{5}$$

$$v_{i,r} \in \mathbb{N} \cup \{0\} \qquad \forall i, r \tag{6}$$

$$\sum_{i} z_{i,j,k} = 1, \tag{7}$$

$$z_{i,j,k} \in [0, 1] \qquad \forall i, j, k$$

$$z_{i,j,k} \cdot d_{j,k} \leq x_{i,j,k} \qquad \forall i, j, k \tag{8}$$

$$\sum_{j} \sum_{k'} x_{i,j,k'} - q_{i,r} \leq y_{i,r} \qquad \forall i, r \tag{9}$$

$$q_{i,r} - \sum_{j} \sum_{k'} x_{i,j,k'} \leq v_{i,r} \qquad \forall i, r \tag{10}$$

$$\sum_{j} \sum_{k} x_{i,j,k} \cdot n_k + \sum_{r} v_{i,r} \leq capacity_i \qquad \forall i \tag{11}$$

$$v_{i,r} = y_{i,r} + q_{i,r} - \sum_{j,k'} x_{i,j,k'} * n_{k',r} \qquad \forall r, i \tag{12}$$

Также мы вводим функцию цели, которая отличается от той, что мы рассмотрели в классической постановке:

$$\sum_{i} \sum_{j} \sum_{k} x_{i,j,k} \cdot n_k \cdot shipping_{i,j} + \sum_{i} \sum_{r} y_{i,r} \cdot procurement_{i,r} \to min \tag{13}$$

4 Используемое программное обеспечение

Одной из самых важных задач в программированиии является грамотный подбор необходимого программного обеспечения для решения задачи. Рассмотрим необходимые библиотеки, которые помогут в программировании задачи.

4.1 Numpy

Numpy[2] - это фундаментальный пакет для научного программирования на языке Python. Эта библиотека обеспечивает пользователя необходимыми математическими объектами, такими как многомерные массивы, матрицы, вектора и т.п. Так же библиотека даёт доступ к математическим операциям над этими объектами, включая логические операции, операции над матрицами, векторами, сортировка, ввод, вывод, дискретные преобразования Фурье, базовая линейная алгебра, статистика, распределение случайных величин и т.д.

4.2 Pandas

Pandas[3] — подходящий инструмент для работы с табличными данными, такими как данные в электронных таблицах или базах данных. Pandas поможет в исследовании, очищении и обрабатке данных.

4.3 Itertools

Itertools[4] - модуль, содержащий множество итераторов, построенных по принципу таких языков как APL, Haskell, и SML. Этот модуль стандартизирует базовый набор инструментов с эффективным использованием памяти. Вместе они образуют «алгебру итераторов», позволяющую эффективно создавать специализированные инструменты на языке Python.

4.4 Ortools

Ortools[1] - библиотека с открытым исходным кодом, позволяющая быстро и эффективно решать задачи оптимизации. Библиотека содержит два решателя задач линейной оптимизации, алгоритмы упаковки контейнеров и рюкзаков, алгоритмы решения зада-

чи коммивояжёра и задачи выбора маршрута транспорта, а также многое другое.

5 Обозначения и предварительные сведения

В отличие от классической задачи Монжа-Канторовича, наша задача дополняется такими условиями как, наличие у складов вместимости, наличие определенного «стартового» набора товаров на складах. Также в нашу модель помимо цен на транспортировку товара мы включаем цену производства товара, которая зависит от склада, где этот товар производится. В дополнении к этому, мы будем работать в терминах наборов товаров, а не отдельно взятых товаров, данное решение поможет избежать нам «сплита» заказов.

Далее мы придерживаемся следующих обозначений:

1. $x_{i,j,k}$ – переменные, обозначающие количество наборов k, которое необходимо отправить со склада i в пункт j;
2. $y_{i,r}$ – переменные, обозначающие сколько предметов r необходимо закупить на складе i;
3. $z_{i,j,k}$ – переменные, обозначающие распределение поставки набора k в пункт j со склада i;
4. $v_{i,r}$ – переменные, обозначающие количество неиспользуемых предметов r, которые собираются хранить на складе i;
5. $capacity_i$ – переменные, обозначающие вместимость склада i;
6. $shipping_{i,j}$ – переменные, обозначающие стоимость перевозки предмета со склада i в пункт j;
7. $procurement_{i,r}$ – переменные, обозначающие стоимость закупки предмета r на складе i;
8. $d_{j,k}$ – переменные, обозначающие сколько надо наборов k в пункте j;
9. $q_{i,r}$ – переменные, обозначающие текущее количество предмета r на складе i;
10. $n_{k,r}$ – переменные, обозначающие количество предметов r в наборе k;
11. n_k – переменные, обозначающие общее количество предметов в наборе k;

12. k' – набор, который содержит предмет r.

6 Решение задачи и программная реализация

6.1 Построение математической модели

Поставим задачу в ранее введённых терминах. Для начала необходимо определить и найти все ограничения.

Так как задача целочисленная, необходимо ввести область допустимых значений для всех переменных

$$x_{i,j,k} \in \mathbb{N} \cup \{0\} \qquad \forall i, j, k \tag{14}$$

$$y_{i,r} \in \mathbb{N} \cup \{0\} \qquad \forall i, r \tag{15}$$

$$v_{i,r} \in \mathbb{N} \cup \{0\} \qquad \forall i, r \tag{16}$$

Далее, введём все необходимые ограничения на переменные

Первое ограничение тривиально:

$$\sum_i z_{i,j,k} = 1, \tag{17}$$

$$z_{i,j,k} \in [0, 1] \qquad \forall i, j, k$$

Второе ограничение следует из того, что мы не можем отправлять в пункт меньше наборов товаров, чем нужно:

$$z_{i,j,k} \cdot d_{j,k} \leq x_{i,j,k} \qquad \forall i, j, k \tag{18}$$

Третье ограничение вводим, чтобы не закупалось больше товаров, чем требуется для отправки:

$$\sum_j \sum_{k'} x_{i,j,k'} - q_{i,r} \leq y_{i,r} \qquad \forall i, r \tag{19}$$

Четвёртое ограничение тривиально:

$$q_{i,r} - \sum_j \sum_{k'} x_{i,j,k'} \leq v_{i,r} \qquad \forall i, r \tag{20}$$

Пятое неравенство даёт нам ограничение на вместимость склада:

$$\sum_j \sum_k x_{i,j,k} \cdot n_k + \sum_r v_{i,r} \leq capacity_i \qquad \forall i \tag{21}$$

С помощью шестого ограничения мы считаем, количество неиспользуемого товара:

$$v_{i,r} = y_{i,r} + q_{i,r} - \sum_{j,k'} x_{i,j,k'} * n_{k',r} \qquad \forall r, i \tag{22}$$

Далее, вводим функцию цели:

$$\sum_i \sum_j \sum_k x_{i,j,k} \cdot n_k \cdot shipping_{i,j} + \sum_i \sum_r y_{i,r} \cdot procurement_{i,r} \tag{23}$$

Именно эта функция и моделирует все затраты в поставленной задаче, поэтому ключевой целью является нахождение минимума этой функции с учётом вышеописанных ограничений, т.е:

$$\sum_i \sum_j \sum_k x_{i,j,k} \cdot n_k \cdot shipping_{i,j} + \sum_i \sum_r y_{i,r} \cdot procurement_{i,r} \to min \tag{24}$$

6.2 Условия существования решения

Условие 1. *Целевая функция (23) должна быть ограничена снизу.*

Условие 2. *Пусть* $\bigcap_{i=1}^{n} A_i = A$, *где* A_i *- множество порождаемое ограничением* i, $E(f)$ *– множество значений целевой функции. Тогда конъюнкция* $E(f) \bigcap A \neq \varnothing$ *должна выполняться.*

Теорема 1. *Если условия (1)–(2) не выполнены, то оптимального решения не существует.*

Доказательство. Рассмотрим условие(1):

Если функция не ограничена снизу, то это значит, что всегда существует более «лучшее» оптимальное решение, по сравнению с тем, что мы нашли, следовательно оптимального решения не существует.

Рассмотрим условие(2):

Действительно, если $E(f) \bigcap A = \varnothing$, то не существует x, такого что $x \in E(f)$ & $x \in A$, следовательно не существует и решения. □

6.3 Структура входных данных

В первую очередь на вход необходимо получить несколько скалярных величин:

1. n –количество складов
2. m – количество центров дистрибуции
3. k – количество различных производимых товаров
4. l – количество наборов товаров

Далее необходимо получить на ввод несколько матриц:

$$\text{Shipping} = \begin{pmatrix} shipping_{11} & shipping_{12} & \dots & shipping_{1m} \\ shipping_{21} & shipping_{22} & \dots & shipping_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ shipping_{n1} & shipping_{n2} & \dots & shipping_{nm} \end{pmatrix}.$$

Элемент матрицы $shipping_{ij}$, обозначает стоимость перевозки товара из склада i в центр дистрибуции j

$$\text{Procurement} = \begin{pmatrix} procurement_{11} & procurement_{12} & \dots & procurement_{1n} \\ procurement_{21} & procurement_{22} & \dots & procurement_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ procurement_{k1} & procurement_{k2} & \dots & procurement_{kn} \end{pmatrix}.$$

Элемент матрицы $procurement_{ij}$, обозначает стоимость закупки товара i на складе j

$$\text{Q} = \begin{pmatrix} q_{11} & q_{12} & \dots & q_{1n} \\ q_{21} & q_{22} & \dots & q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{k1} & q_{k2} & \dots & q_{kn} \end{pmatrix}.$$

Элемент матрицы q_{ij}, обозначает начальное количество товара i на складе j

$$\text{D} = \begin{pmatrix} d_{11} & d_{12} & \dots & d_{1m} \\ d_{21} & d_{22} & \dots & d_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ d_{l1} & d_{l2} & \dots & d_{ln} \end{pmatrix}.$$

Элемент матрицы d_{ij}, обозначает сколько надо наборов i в пункте j.

Следом на вход программы необходимо получить массив данных:

$$\text{Capacity} = \begin{pmatrix} \text{capacity}_1 \\ \text{capacity}_2 \\ \vdots \\ \text{capacity}_m \end{pmatrix},$$

где $capacity_i$ – вместимость i-го склада.

Таким образом, этих данных необходимо и достаточно для инициализации модели.

6.4 Программная реализация

Для программной реализации поставленной задачи мы используем бибилотеку ORTools. Для начала необходимо инициализировать «решатель».

```
def get_solver():
    solver = pywraplp.Solver.CreateSolver('SCIP')
    if not solver:
        exit("No Solver")
    return solver

solver = get_solver()
```

Далее необходимо задать область допустимых значений для всех переменных, для того чтобы задать область для вещественной переменной используем метод solver.NumVar(), для целочисленных же переменных используем метод solver.IntVar():

```
for i, j, k in itertools.product(SUPPLY_NODE_LIST, DEMAND_NODE_LIST, ITEMSET_LIST):
    x[i, j, k] = solver.IntVar(0, infty, f'x[{i}][{j}][{k}]')
    z[i, j, k] = solver.NumVar(0, 1, f'z[{i}][{j}][{k}]')

for i, r in itertools.product(SUPPLY_NODE_LIST, SKU_LIST):
    y[i, r] = solver.IntVar(0, infty, f'y[{i}][{r}]')
    v[i, r] = solver.IntVar(0, infty, f'v[{i}][{r}]')
```

Затем вводим все необходимые ограничения с помощью двух методов. Методом solver.RowConstraint задаём правые части неравенств, с помощью метода solver.SetCoefficient задаем необходимые коэффициенты перед переменными. Покажем на примере, ввёдем ограничение (17). Полный ввод всех ограничений можно посмотреть по ссылке [7].

```
for j, k in itertools.product(DEMAND_NODE_LIST, ITEMSET_LIST):
    constraint = solver.RowConstraint(1, 1, f'equation 1: (j, k) = ({j}, {k})')
```

```
    for i in SUPPLY_NODE_LIST:
      constraint.SetCoefficient(z[i, j, k], 1)
```

После ввода всех ограничений инициализируем функцию цели (23) с помощью метода solver.Objective() и начинаем расставлять коэффициенты перед всеми переменными с помощью метода objective.SetCoeff Следом минимизируем функцию цели.

```
for i, j, k in itertools.product(SUPPLY_NODE_LIST, DEMAND_NODE_LIST, ITEMSET_LIST):
    objective.SetCoefficient(x[i, j, k], item_set_sku_id_count[k]*shipping_cost_dict[i, j
        ]['cost'])

for i, r in itertools.product(SUPPLY_NODE_LIST, SKU_LIST):
    objective.SetCoefficient(y[i, r], supply_dict[i, r]['cost'])

objective.SetMinimization()
```

7 Обсуждение результатов

В качестве результата работы мы получили программную реализацию расширенной задачи Монжа-Канторовича для произвольных входных данных. Полный листинг программы доступен по ссылке [7].

Список литературы

[1] https://github.com/google/or-tools

[2] https://numpy.org/

[3] https://pandas.pydata.org/docs/

[4] https://docs.python.org/3/library/itertools.html

[5] Богачев В.И., Колесников А.В. Задача Монжа-Канторовича: достижения, связи и перспективы // Успехи математических наук. – 2012. Т.67. – №5(407). – С. 3-110.

[6] https://www.researchgate.net/publication/354671163_SUPPLY_CHAIN_MANAGEMENT_A_REVIEW

[7] https://github.com/SIPLIbY/Monge-Kantorovich-s-problem/blob/main/main.py

[8] Monge G. Mémoire sur la théorie des déblais et de remblais. Histoire de l'Académie Royale des Sciences de Paris, avec les Mémoires de Mathématique et de Physique pour la même année, pages 666—704, 1781.

[9]] Л. В. Канторович, “О перемещении масс”, Докл. АН СССР, 37 (1942), 227–229

[10] https://www.nobelprize.org/prizes/economic-sciences/1975/kantorovich/facts/

[11] George B. Dantzig; Philip Wolfe. Decomposition Principle for Linear Programs // Operations Research — 1960. — Т. 8. — С. 101—111.

[12] Papadimitriou, C. H.; Steiglitz, K. (1998). Combinatorial optimization: algorithms and complexity. Mineola, NY: Dover. ISBN 0486402584.