Секция «Сети в экономике и менеджменте: теория и практика»

Моделирование модернизации дорожных сетей городов

Научный руководитель – Рюмкин Валерий Иванович

Якунин Сергей Александрович
Студент (бакалавр)
Национальный исследовательский Томский государственный университет,
Экономический факультет, Томск, Россия
E-mail: ciellol@yandex.ru

Дорожные сети являются неотъемлемой частью городов, агломераций, мегаполисов и больших территориальных образований. Роль дорожных сетей городов в экономической сфере заключается в доставке товаров в магазины, сырья на производство и пр. Актуальность темы исследования, таким образом, обусловлена важностью оптимизации городских дорожных сетей, поскольку они прямо влияют на экономику городов, и, как следствие, на экономику страны.

Качество транспортных сетей можно оценить различными методами. Например, применяя теорию гидравлических сетей [6], теорию экономического равновесия [3], используя муравьиный подход [4] и др. Однако, все эти подходы требуют достоверной и полной информации (подробный автомобильный трафик), получить которую достаточно затруднительно.

Целью настоящей работы стало формирование более универсального и доступного метода моделирования транспортной сети с целью ее дальнейшей модернизации. Практическая реализация данного метода была рассмотрена на примере города Томска.

Постановка модели. Пусть существует сеть автомобильных дорог, которую можно представить в форме графа, где вершинами являются перекрестки дорог, дуги - входящие и выходящие из перекрестков дороги. При помощи интернет-сервисов карт возможно найти координаты перекрестков и выгрузить их единым массивом. Затем происходит самый трудоемкий этап - формирование матрицы инцидентности, которая будет характеризовать граф. Затем, при помощи программных средств можно сформировать случайные наборы потенциальных входящих и выходящих потоков из вершин в другие вершины. А уже в такой форме решение задачи сводится к решению задачи линейного программирования [5].

Безусловно, данная модель не является сильно приближенной к реальной дорожнойсети города Томска. А решение данной задачи не позволит оценить оптимальность дорожной сети объективно. Однако, данная модель, например, может позволить сравнить заданную сеть с другой, потенциально возможной. Так, например, мы можем ввести новые вершины или ребра графа, что позволит нам рассматривать некоторые потенциальные модернизации транспортной сети. А в качестве параметра, по которому мы сможем выбрать наилучшую возможную модернизацию из представленных - мы будем использовать минимально возможную сумму времен, затраченных на проезд из одних вершин графа в другие.

Источники и литература

1) Булавина Л. В. Проектирование и оценка транспортной сети и маршрут-ной системы в городах
2) Введение в математическое моделирование транспортных потоков / А. В. Гасников [и др.]. – МЦНМО. – Москва. – 2013. – 428 с.

3) Коваленко А. Г. Теоретико-игровой подход и теория гидравлических се-тей в проблеме моделирования движения городских транспортных потоков // Вестник СамГУ. –2013. – Том 7. – №1.

4) Кочегурова Е. А. Алгоритм муравьиных колоний для задачи проектиро-вания рациональных маршрутных сетей городского пассажирского транспорта // Вест-ник СибГУТИ. – 2014. – №3. с. 89-100.

5) Васин А.А., Морозов В.В. Теория игр и модели математической эконо-мики: учебное пособие. М.: МАКС Пресс, 2005. 272 с.

6) Основы теории транспортных систем: учебное пособие / А. Э. Горев. – СПбГАСУ. – Санкт-Петербург. – 2010. – 214 с.

Иллюстрации

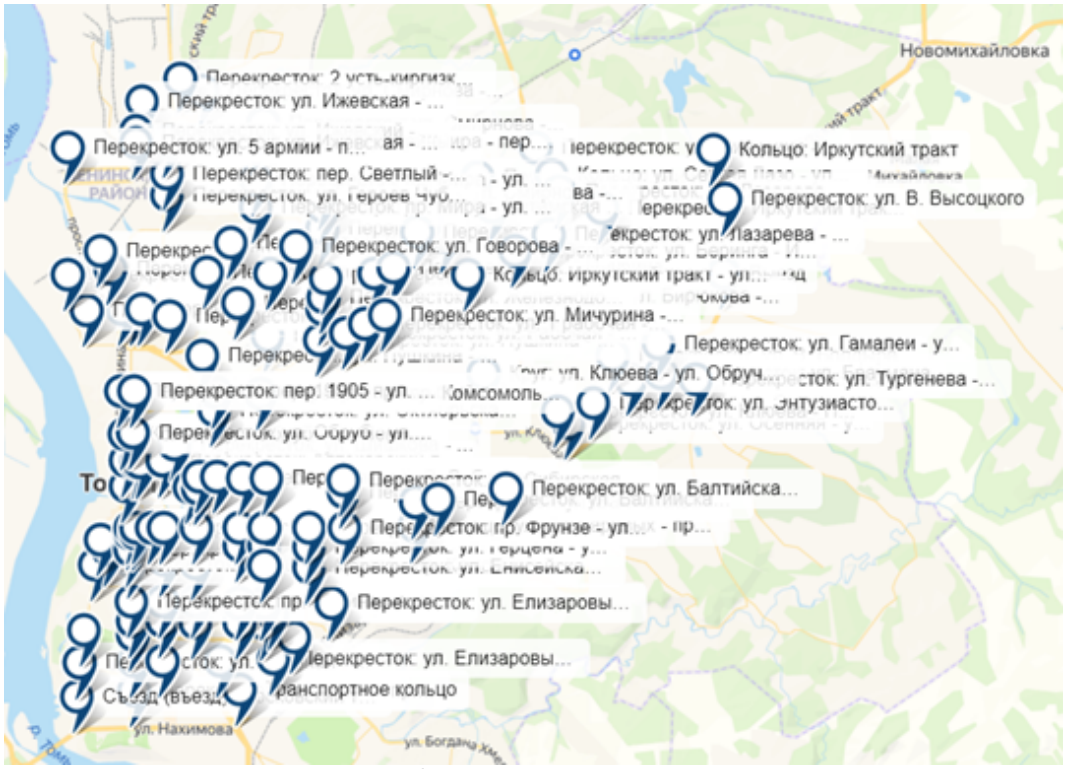

Рис. Построение графа при помощи сервиса Яндекс Карт

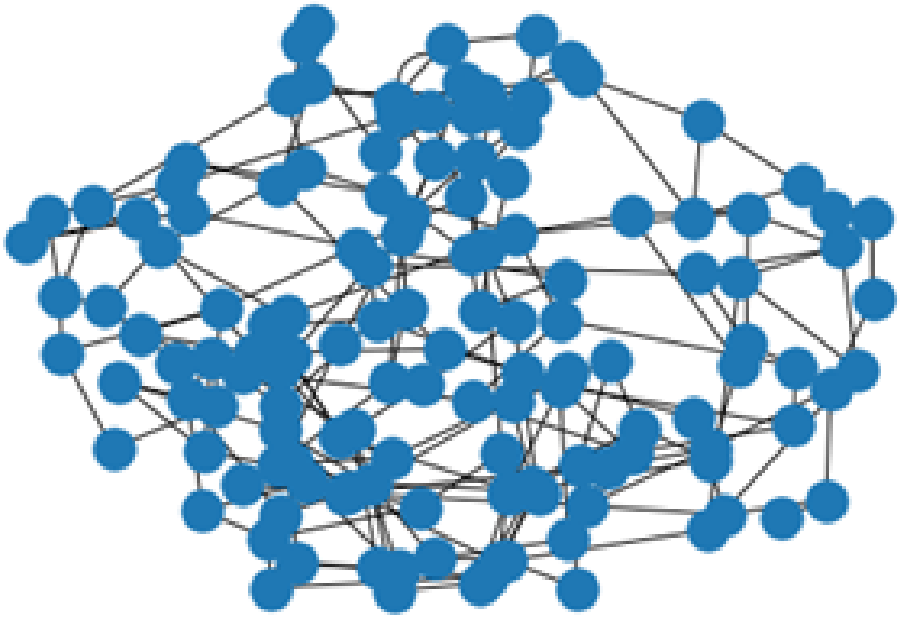

Рис. Визуализация полученного графа

```python
def sev_zap(a_, b_, c_):
    a = np.copy(a_)
    b = np.copy(b_)
    c = np.copy(c_)

    # Проверяем условие замкнутости:
    if a.sum() > b.sum():
        b = np.hstack((b, [a.sum() - b.sum()]))
        c = np.hstack((c, np.zeros(len(a)).reshape(-1, 1)))
    elif a.sum() < b.sum():
        a = np.hstack((a, [b.sum() - a.sum()]))
        c = np.vstack((c, np.zeros(len(b))))

    m = len(a)
    n = len(b)
    i = 0
    j = 0
    funk = 0
    x = np.zeros((m, n), dtype=int)
    while (i<m) and (j<n): # повторяем цикл до сходимости метода
        x_ij = min(a[i], b[j]) # проверяем минимальность a_i и b_j
        funk += c[i, j]*min(a[i], b[j]) # записываем в итоговую функцию элемент трат
        a[i] -= x_ij #
        b[j] -= x_ij # обновляем векторы a и b
        x[i, j] = x_ij # добавляем элемент x_ij в матрицу x

        if a[i]>b[j]: # делаем сдвиги при выполнении условий
            j += 1
        elif a[i]<b[j]:
            i += 1
        else:
            i += 1
            j += 1
    return x, funk # возращаем матрицу x и целевую функцию
```

Рис. Пример решения полученной задачи линейного программирования методом северо-западного угла

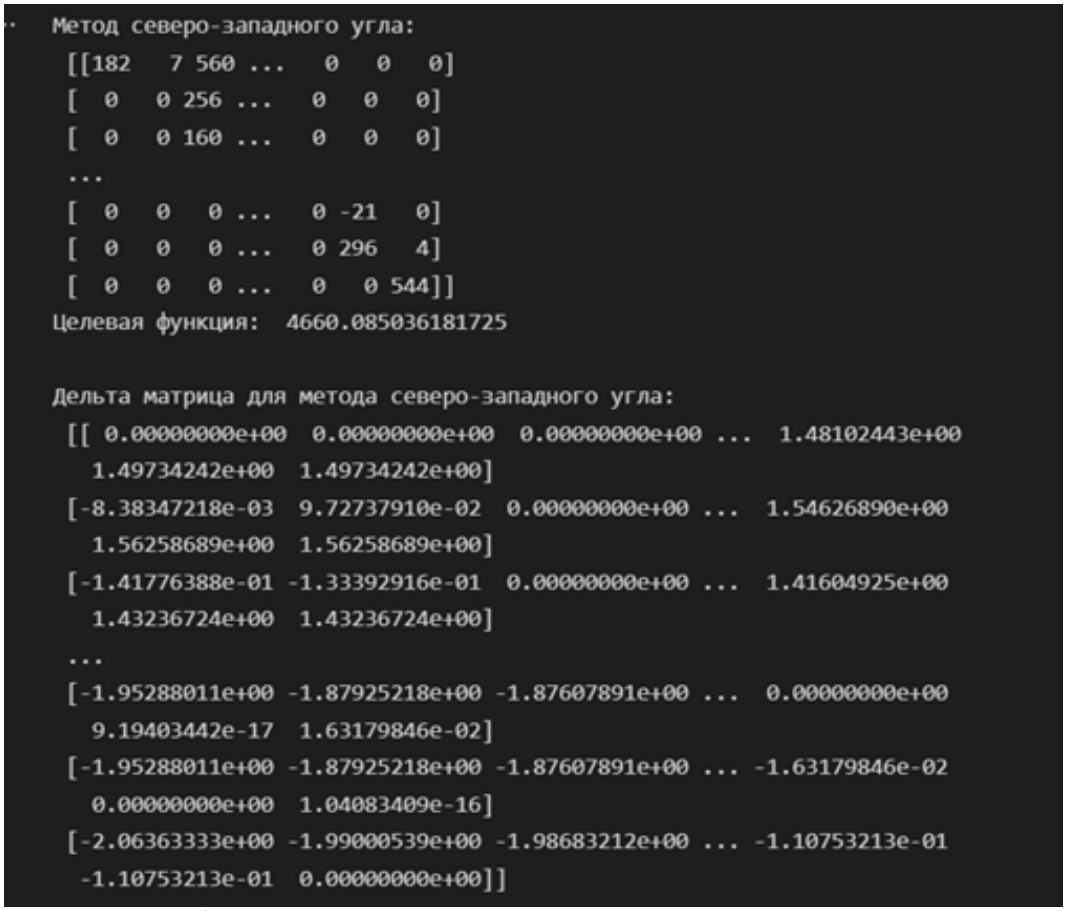

Рис. пример вывода функции, решающей задачу линейного программирования методом северо-западного угла