Министерство науки и высшего образования
Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«Ярославский государственный университет им. П.Г. Демидова»

Кафедра дифференциальных уравнений

Сдано на кафедру
«_____» июня 2023 г.
Заведующий кафедрой
д.ф.-м.н., профессор
_____ Бережной Е. И.

Курсовая работа

Оптимизация работы общественного городского транспорта

направление подготовки
01.03.02 Прикладная математика и информатика

Научный руководитель
Ассистент кафедры дифференциальных уравнений,
старший преподаватель
_____ Преображенский И. Е.
«_____» июня 2023 г.

Студент группы ПМИ-33БО
_____ Сидоров М.В.
«_____» июня 2023 г.

Ярославль 2023 г.

Содержание

1 Введение

Рассматривается задача оптимизации работы городского общественного транспорта. Предприятия городского общественного транспорта осуществляют перевозки людей с использованием транспортных средств различных видов (автобус, троллейбус, трамвай) и вместимости. Транспорт перемещается по заданным маршрутам согласно расписанию, совершая остановки в оборудованных для этого местах. При осуществлении движения по маршруту транспортным средством управляет водитель, контроль за оплатой проезда может выполнять кондуктор-контролёр. Расписание движения должно учитывать обеденный и технические перерывы, во время которых транспортное средство простаивает в одной из конечных точек маршрута в течение заданного времени.

Люди пользуются общественным транспортом для перемещения на работу, учёбу, в учереждения здравоохронения, на спортивные мероприятия и просто по своим повседневным делам. Распределение числа пассажиров, которым необходимо доехать из пункта A в пункт B, неравномерно и зависит от дня недели, времени суток, времени года и от нерегулярных мероприятий. В будние дни утром люди отправляются из спальных районов на работу и учёбу, а вечером возвращаются обратно. В летний сезон отпусков число пассажиров общественного транспорта меньше, чем зимой, при этом в пятницу вечером люди едут из города на дачи, пользуясь автомобильным и железнодорожным междугородним транспортом. При проведении массовых спортивных мероприятий, люди едут к месту проведения этих мероприятий к моменту их начала и возвращаются домой после их окончания.

С точки зрения жителя города работа общественного транспорта оценивается по следующим критериям.

- Время перемещения из точки A в точку B.
- Число пересадок, которое необходимо сделать при движении.
- Комфорт передвижения, который зависит как от состояния материально-технической базы, так и от загруженности транспортного средства.
- Время ожидания транспорта на остановках.
- Время начала и окончания работы маршрута.

С точки зрения пассажира, идеальное перемещение из точки A в точку B выглядит следующим образом. Пассажир мгновенно садится в транспорт на остановке, которая находится прямо в точке A, в пустой автобус и едет без остановок по кратчайшему маршруту до остановки, которая находится в точке B.

Движение общественного транспорта зависит от следующих факторов.

- Маршруты движения. Автобусные маршруты могут двигаться только по автомобильным дорогам надлежащего качества с учётом правил дорожного движения. Троллейбусные маршруты, кроме упомянутого выше, должны быть электрифицированы. Трамвайные маршруты могут двигаться только по трамвайным путям. Посадка и высадка пассажиров могут осуществляться только на специально оборудованных остановках. Если маршурт не круговой, то на каждом из его концов обязательно должно быть место для разворота транспорта. В хотя бы одной точке маршрута (обычно в одном из его концов) должно быть оборудованное место для обеда и отдыха водителей и кондукторов-контроллёров. От качества дорожного полотна или трамвайных рельсов зависит скорость движения транспорта.

- Количество и модели транспортных средств на каждом из маршрутов. Чем больше доступных транспортных средств, тем меньше может быть интервал движения между ними. От модели транспортного средства зависит его вместимость (при желании можно отдельно выделить вместимость по количеству сидячих мест), скорость движения, время движения между техническими перерывами (например, заправкой автобусов топливом).

- Расписание движения транспортных средств. Определяется тем, в какие моменты времени транспортное средство прибывает на остановки и какая у него модель. Время перемещения между остановками может зависеть от ограничений скорости, светофоров, загруженности дорог (в часы пик) или трамвайных путей, модели транспортного средства, количества перевозимых пассажиров. Стоит отметить, что при движении с одной или несколькими пересадками время, потраченное на пересадку зависит от согласованности маршрутов движения. Важным фактором комфорта для жителей является время начала и окончания движения транспорта по маршруту.

Нашу задачу мы хотим сформулировать в терминах задачи линейного программирования.

2 Задачи линейного программирования

2.1 Постановка задач линейного программирования

Задача линейного программирования (ЛП) в стандартной форме выглядит следующим образом.

Мы хотим найти минимум целевой функции $f(x)$

$$f(x) = \sum_{i=1}^{n} c_i x_i$$

При следующих ограниченинях:

$$\sum_{j=1}^{n} a_{ij} x_j \geqslant b_j$$

$$x_j \geqslant 0$$

Стандартным подходом к решению такой задачи является применение симплекс-метода. Опишем основную идею этого метода.

Каждое из линейных неравенств ограничивает некоторое подпростанство в линейном пространстве размерности n. В совокупности все неравенства ограничивают полиэдр в этом линейном пространстве. Уравнение $f(x) = c$, где $f(x)$ — целевая функция, порождает гиперплоскость $L(c)$.

Переформулируем задачу следующим образом: требуется найти наименьшее c, для которого гиперплоскость $L(c)$ пересекает полиэдр хотя бы в одной точке. Заметим, что пересечение оптимальной гиперплоскости и полиэдра будет содержать хотя бы одну вершину. В случае, если вершин будет более одной, в пересечении будет находиться ребро или грань полиэдра. Поэтому минимум целевой функции можно искать в вершинах многогранника.

Принцип симплекс-метода состоит в том, что выбирается одна из вершин многогранника, после чего начинается движение по его рёбрам от вершины к вершине в сторону уменьшения значения целевой функции. Когда переход по ребру из текущей вершины в вершину с меньшим значением целевой функции невозможен, считается, что оптимальное значение c найдено.

Подробное описание алгоритма симплекс-метода Данцига можно найти в [1].

2.2 Задачи целочисленного линейного программирования и солверы

В нашей задаче мы хотим дополнительно потребовать целочисленность переменных $x_j \in \mathbb{Z}$. Поэтому наша задача будет сформулирована в терминах задачи целочисленного линейного программирования (ЦЛП). Для решения таких задач существуют специальные программные комплексы - солверы. Разберем принцип их работы.

Работу солвера можно разбить на следующие этапы:

1. **Решение линейной релаксации**

   В начале солвер ищет решение для задачи, пренебрегая целочисленностью переменных. Такая задача называется линейной релаксацией исходной и решается симплекс-методом.

2. **Эвристические алгоритмы**

Цель этих алгоритмов - найти какие-то допустимые решения задачи ЦЛП за короткий срок. Они не гарантируют оптимальность найденных решений и вообще не гарантируют, что смогут найти хотя какое-то решение. Найденные решения могут быть полезны для работы oracle в алгоритме ветвей и границ.

Ознакомиться с основными эвристиками можно в обзорной статье [2].

3. **Метод секущих плоскостей**

   Цель этого метода в уменьшении размера полиэдра генерацией новых ограничений и соответствующих им секущих гиперплоскостей.

   Ознакомиться с реализациями методов секущих плоскостей можно в [4].

4. **Метод ветвей и границ**

   Метод ветвей и границ (Branch & Bound, B&B) - финальная стадия работы солвера, когда он начинает делать "разумный"полный перебор. Найденное решение уже будет оптимальным.

   Для работы этого алгоритма необходим вспомогательный метод *oracle*.

   Пусть $\overline{S}$ - пространство, в котором мы ищем решение.

   $oracle(x) = FALSE$, если решение $x \in \overline{S}$ недопустимо.

   $oracle(U) = TRUE$, если подпространство $U \in \overline{S}$ может содержать допустимое/оптимальное решение.

   Основная идея B&B - рекурсивное деление пространства поиска решений на подпространства и поиск решений в этих подпространствах.

   Приведем пример стандартной реализации этого алгоритма для задачи ЦЛП, поставленной следующим образом:

$$(P) : z = min\{cx | Ax \leqslant b, c, x \in \mathbb{Z}_+^n\}$$

   z^* - наименьшее уже найденное значение целевой функции.

   $\overline{S} = \{x | Ax \leqslant b\}$

   Будем выбирать подпространства в порядке LIFO обходом в глубину.

   На каждой итерации обрабатываем подпространство U следующим образом:

$$\overline{z}^U = min\{cx | x \in U\}, \text{ тогда } orale(U) = \overline{z}^U \leqslant z^* - 1$$

   - Если $oracle(U) = FALSE$ отбросим подпростанство U.
   - Если $oracle(U) = TRUE$ и $\overline{x}^U \in S$ обновим $z^* = \overline{z}^U$.
   - Иначе сделаем ветвление, поделим пространство U на два подпростанства, исключая $\overline{x}^U$. Выберем индекс $i \in \{1, ..., n\}$ и получим две ветки:
     - Ветвь 1: $U \cap \{x_i \leqslant |\overline{x_i^U}|\}$
     - Ветвь 2: $U \cap \{x_i \geqslant |\overline{x_i^U}|\}$

3 Задачи оптимизации общественного транспорта

3.1 Постановка задач оптимизации общественного транспорта

В рамках оптимизации общественного транспорта можно рассмотреть следующие задачи.

- Составление расписания движения транспорта по одному маршруту.
- Частичное изменение маршрутов движения транспорта.
- Построение городской маршрутной сети.
- Оптимизация расположения остановочных комплексов.
- Задача оптимальной закупки дополнительных транспортных средств при ограниченном бюджете.

Эти и многие другие задачи, связанные с оптимизацией общественного транспорта, в разных постановках рассматриваются в [3], [5].

Далее рассмотрим упрощенную задачу оптимизации расписания для одного маршрута.

3.2 Составление расписания движения транспорта по одностороннему маршруту: цепочка станций

Рассматривается горизонт планирования T и множество остановок $1, \ldots, k$. В каждый момент времени $t \in 1 \ldots T$ на остановку i приходит $a_{i,j,t}$ пассажиров, которые хотят попасть на остановку j, где $j > i$. По маршруту в горизонте планирования транспортные средства вместимости c совершают рейсы. Время движения между остановками i и $i+1$ равняется p_i. Необходимо составить расписание рейсов так, чтобы суммарное время ожидания рейсов для пассажиров было минимально или доказать, что такого не существует. Отметим, что пассажиры, вышедшие на остановке, освобождают место в автобусе, которое могут занять другие пассажиры, заходящие на той же остановке или на последующих.

3.3 Модель ЦЛП

Рассмотрим следующую модель целочисленного линейного программирования, которая позволяет решать упрощённую постановку рассматриваемой задачи.

3.3.1 Исходные данные

- T – горизонт планирования.
- n – количество остановок.
- $p_i, \forall i = 1, \ldots, n-1$ – время движения между остановками i И $i+1$. Мы также будем использовать вспомогательные числа $P_i = \sum_{j=1}^{i} p_i$ $(i \in \overline{2, ..., n})$ – время движения от первой станции до станции i.
  А также $P_i^* = \sum_{j=1}^{i} p_i \quad (i \in \overline{1, ..., n-1})$, где $P_1^* = 0$.
- $g_{ij}(t)$ – количество пассажиров, приходящее в момент времени t на станцию i с целью отправиться на станцию $j > i$.
- m - доступное число транспортных средств.
- c - вместимость транспортного средства.

3.3.2 Переменные

Введем следующие переменные.

- $x_{ij}(t)$ – количество пассажиров, которые отправились со станции i на станцию j в момент времени t.
- y_{it} – количество транспортных средств, отправившихся от остановки i в момет времени t.
- z_{it} – количество пассажиров, оставшихся на остановке i по окончании момента времени t.

3.3.3 Ограничения

1. $x_{ij}(t)$, z_{ij} положительны для любых i, j, t.
2. Число человек на остановке i в момент времени t – те, кто пришел за вычетом тех, кто уже уехал.
$$\forall i, t \quad z_{it} = \sum_{\tau=0}^{t} \sum_{j=j+1}^{n} g_{ij}(t) - \sum_{\tau=0}^{t} \sum_{j=j+1}^{n} x_{ij}(t)$$
3. Число человек, уехавших с остановки i на остановку j за весь горизонт планирования, равняется количеству прибывших на остановку i и желающих поехать на остановку j.
$$\forall i, j \quad \sum_{t=1}^{T} x_{ij}(t) = \sum_{t=1}^{T} g_{ij}(t)$$
4. На остановке в любой момент времени не должно находиться слишком много ожидающих транспорт пассажиров.
$$\forall i, t \quad z_{i,t} \leqslant M$$
5. Пассажиры могут начать перемещение со станции i в момент времени t только в случае отправления транспорта, и в него помещается число пассажиров, не превосходящее его вместимости.
$$\forall i, t \quad cy_{it} \geq \sum_{j=j+1}^{n} x_{ij}(t)$$
6. Вместимость транспортного средства не нарушается для любого участка движения.
$$\forall t \leqslant T - P_n, k \quad \sum_{i=1}^{k} \sum_{j=k+1}^{n} x_{ij}(t + P_i^*) \leqslant cy_{1t}$$
7. Время отправления транспорта с первой станции однозначно определяет время отправления со станции i.
$$\forall t \leqslant T - P_i \quad y_{1,t} = y_{i,t+P_i}$$
8. Все пассажиры должны доехать до со своих станций к концу горизонта планирования.
$$\forall i \leqslant n-1, j \geqslant i+1, t \geqslant T \quad \sum_{k=i}^{j-1} p_i x_{ij}(t) = 0$$

9. Также выполнено условие, что в конце горизонта планирования на любой остановке i никого не осталось.

$$\forall i \quad z_{i(T-1)} = 0.$$

10. Если в момент времени t от станции 1 выехали y_{1t} транспортных средств, то пока они не закончат рейс, они не могут отправляться от других остановок.

$$\forall i, t \leqslant T - P_i \quad y_{1t} > 0 \implies \sum_{\tau=0}^{P_i} y_{i,t+\tau} \leqslant m - y_{1t}$$

11. Если в момент времени t от станции 1 выехали y_{1t} транспортных средств, то они не могут отправляться от станции 1, пока не закончат рейс.

$$\forall t \leqslant T - P_n \quad y_{1t} > 0 \implies \sum_{\tau=1}^{P_n} y_{1,t+\tau} \leqslant m - y_{1t}$$

12. Транспортные средства не могут появиться на остановке $i > 1$ до отправления от остановки 1 в начале горизонта планирования.

$$\forall i > 2, t < P_i \quad y_{it} = 0$$

13. Нет отправлений, если автобус не доедет до конечной станции в течение горизонта планирования.

$$\forall t > T - P_n \quad y_{it} = 0$$

3.3.4 Целевая функция

Рассматривается функция минимизации суммарного времени ожидания транспорта всеми пассажирами.

$$\min \sum_{i=1}^{n-1} \sum_{t=0}^{T-1} z_{it}$$

3.4 Генерация данных для тестирования модели

Для примера возьмем ярославский маршрут 22с, автобусы по которому следуют от остановки Красная площадь до остановки Студенческий городок.

Предположим следующие исходные данные.

- Пусть автобусы следуют по маршруту с 7:00 до 22:00, возьмем дискретизацию времени равную пяти минутам, тогда горизонт планирования $T = 180$.

- Всего остановок будет 17 ($n = 17$):

  Красная площадь → Октябрьская площадь → Улица Дачная → Школа № 50 → Школа № 46 → 8-й переулок → Медсанчасть ЯЗДА → Университетский городок → Студенческий городок → Университетский городок → Медсанчасть ЯЗДА → 8-й переулок → Школа № 46 → Школа № 50 → Улица Дачная → Октябрьская площадь → Красная площадь.

- Предположим, что время следования между остановками будет составлять следующее количество времени, выраженное в пяти минутах, соответственно.

  p = [2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1]

- По маршруту будут следовать всего два автобуса вместимостью 40 пассажиров ($c = 40$).

Сгенерируем пассажиропоток следующим образом. Сгенерериуем массив необходимой размерности из чисел от 0 до 8, вероятность появления каждого числа в котором равна [0.975, 0.015, 0.005, 0.0025, 0.0015, 0.0005, 0.0001, 0.0001, 0.0003], соответственно. Теперь мы хотим добавить "всплески"в те моменты времени, когда студенты отправляются с утра в университет и уезжают из него после. Для этого добавим к нашему массиву функцию Гаусса, каждое значение которой округлим до целого числа, в необходимые моменты времени для нужных остановок.

$$G(x) = ae^{-\frac{(x-b)^2}{2c^2}}$$

Где a - максимальная высота графика, b - среднее значение (сдвиг от 0), c - стандартное отклонение. Добавим эту функцию со средним значением в 8 часов, стандартным отклонением 4 и высотой 8 к пассажиропотоку людей с остановок Красная площадь и Школа № 50, следующих на остановку Студенческий городок. Для пассажиропотока с остановки Студенческий городок до Школы №50 и Красной площади добавим три "всплеска"в 14, 15, 16 часов со стандартным отклонением 2 и высотой 7.

Код для генерации пассажиропотока указан в приложении 1. Графики пассажиропотока для конкретных остановок изображены в приложении 3.

3.5 Имплементация модели

Будем использовать платформу MiniZinc (https://www.minizinc.org/). Язык программирования, используемый в MiniZinc, позволяет формулировать задачи програмирования в ограничениях на высоком уровне абстракции. Его основной функционал заключается в том, что он преобразует записанные нами ограничения в линейные и передает их солверу в необходимом виде, что позволяет нам описывать модель независимо от выбранного солвера. В качестве солвера для задач линейного программирования выберем CPLEX 12.10 (https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-optimizer).

Код, реализующий модель в MiniZinc, указан в приложении 2.

В результате получим следующее расписание, в котором указано время отправления транспортных средств от начальной остановки.

['07:40', '08:05', '09:15', '10:05', '10:50', '11:40', '12:25', '13:35', '14:10', '15:25', '16:15', '17:00', '17:50', '18:40', '19:25', '20:15']

Заметим, что оптимальное значение целевой функции равняется 3662, то есть за день было потрачено 305 человеко-часов на ожидание транспорта. Учитывая, что за день маршрут обслужил 828 пассажиров, среднее время ожидания составило 22 минуты.

Если мы увеличим количество транспортных средств до 3, то получим следующее расписание:

['07:00', '07:35', '08:00', '08:35', '09:10', '09:40', '10:10', '10:45', '11:35', '12:00', '12:25', '13:20', '13:45', '14:10', '14:55', '15:25', '16:00', '16:30', '17:05', '17:35', '18:20', '19:00', '19:25', '20:10']

Оптимальное значение целевой функции теперь будет равняется 2136, то есть за день будет потрачено 178 человеко-часов на ожидание транспорта. Среднее время ожидания составит 13 минут. Таким образом, от добавления на маршрут еще одного транспортного средства время ожидания существенно уменьшится.

4 Заключение

В результате проделанной работы мы научились ставить прикладные задачи в терминах задач ЦЛП, применять современные программные комплексы для их решения. В ходе решения поставленной задачи мы ознакомились с основными принципами работы солверов. Мы сгенерировали конкретные данные для тестирования созданной модели и получили оптимальное решение для нашей задачи, сделали небольшой анализ полученных результатов.

Работу можно продолжить, оптимизировав модель ЦЛП изменением и добавлением ограничений с целью повышения скорости поиска решения. Также можно расширить модель, попытаться избавиться от недостатков, имеющихся у текущей модели.

- Модель не учитывает влияние трафика на скорость движения транспорного средства в разное время суток.
- Модель не предусматривает выпуск на один маршрут транспортных средств различной вместимости.
- Модель оптимизирует время ожидания, но не оптимизирует степень наполненности транспорта пассажирами.

5 Литература

Список литературы

[1] Пантелеев А.В., Летова Т.А. 2015 Методы оптимизации в примерах и задачах **ISBN** 978-5-8114-1887-9.

[2] Fischetti, Matteo and Lodi, Andrea 2010 Heuristics in mixed integer programming *Wiley Encyclopedia of Operations Research and Management Science*

[3] Konstantinos Gkiotsalitis 2022 Public Transport Optimization *Springer* **ISBN** 978-3-0311-2443-3.

[4] Marchand, Hugues and Martin, Alexander and Weismantel, Robert and Wolsey, Laurence 2002 Cutting planes in integer and mixed integer programming *Discrete Applied Mathematics* **123** no. 1-3 397–446

[5] Toth, Paolo and Vigo, Daniele 2002 The vehicle routing problem *SIAM*

6 Приложение

6.1 Приложение 1

```
import numpy as np
import matplotlib.pyplot as plt

T = 180  # Planning horizon
sleep = 20
stations = 16  # Stations num minus 1

# Probability for each num
probs = [0.975, 0.015, 0.005, 0.0025, 0.0015, 0.0005, 0.0001, 0.0001, 0.0003]

nulls = np.zeros(T)
seq = np.array([])

for i in range(stations):
    for j in range(stations):
        if j >= i and (i < 8 and j <= 8 or i >= 8 and j > 8):
            t = np.random.choice(range(len(probs)), size=T - sleep, p=probs)
            t = np.concatenate([t, np.zeros(sleep)])

            # Students go to university
            if j == 8 and (i == 0 or i == 3):
                mean = 12
                std = 4
                height = 8

                # Generate array with Gauss func values
                x = np.arange(0, T - sleep, 1)
                gaus = height * np.exp(-(x - mean) ** 2 / (2 * std ** 2))
                gaus = np.concatenate([gaus, np.zeros(sleep)])
                t += np.round(gaus)

            # Students go from university
            if i == 8 and (j == 13 or j == 16):
                for mean in 90, 110, 120:
                    std = 2
                    height = 7

                    # Generate array with Gauss func values
                    x = np.arange(0, T - sleep, 1)
                    gaus = height * np.exp(-(x - mean) ** 2 / (2 * std ** 2))
                    gaus = np.concatenate([gaus, np.zeros(sleep)])
                    t += np.round(gaus)

            seq = np.concatenate([seq, t])
        else:
            seq = np.concatenate([seq, nulls])

f = open('text.txt', 'w')
for i in range(len(seq)):
    f.write(str(int(seq[i])) + ", ")
```

6.2 Приложение 2

```
% Parameters
int: T;  % Planning horizon
int: n;  % Stations num
```

```
int: m;  % Vehicles num
int: cp; % Vehicles capacity
set of int: nws = 1..n-1;
set of int: nw = 2..n;
set of int: Tw = 1..T;

% passing time between i and i+1 stations
array[nws] of int: p;
% passing time between 1 and i stations  (i from 2 to n)
array[nw] of int: P;
% passing time between 1 and i stations  (i from 1 to n-1)
array[nws] of int: P2;
% num of passengers, coming at moment t on station i to go to station j > i

array[nws, nw, Tw] of int: g;

% Variables
% num of passengers who are going from station i to j at time t
array[nws, nw, Tw] of var int: x;
% num of vehicles goes from station i at moment t
array[nws, Tw] of var 0..1: y;
% num of passengers standing at station at moment t
array[nws, Tw] of var int: z;

% Constraints
% x, z have to be positive
constraint forall(i in nws, j in nw, t in Tw)(x[i, j, t] >= 0);
constraint forall(i in nws, t in Tw)(z[i, t] >= 0);
constraint forall(j in 2..n, i in j..n-1, t in Tw)(x[i, j, t] == 0);
% Expressing z in terms of g and x
constraint forall(i in nws, t in Tw)(
             z[i, t] = sum(dj in (i+1)..n, dt in 1..t)(g[i, dj, dt])
              - sum(dt in 1..t, dj in (i+1)..n)(x[i, dj, dt])
             );

% The number of people leaving from stop i to j over the entire
% planning horizon is equal to the number of arrivals
constraint forall(i in nws, j in nw) (sum(t in Tw) (x[i, j, t]) ==
                                      sum(t in Tw)(g[i,j,t]));

% Passengers start moving only when the transport arrives,
% and no more than its capacity gets into the transport
constraint forall(i in nws, t in Tw)(cp * y[i, t] >=
                       sum(j in (i+1)..n)(x[i, j, t]));

% Vehicle capacity is not violated for any section of traffic
constraint forall(t in 1..T-P[n]+1, k in nws)
    (sum(i in 1..k, j in (k+1)..n)(x[i, j, t + P2[i]]) <= y[1, t] * cp);

% The departure time of transport from subsequent stations
% depends on the departure time from the first station
constraint forall(i in 2..(n-1), t in 1..T-P[i])(y[1,t] == y[i, t + P[i]]);

% All passengers reach their station by the end of the planning horizon
constraint forall(i in nws, j in (i+1)..n, t in T-sum(k in i..j-1)
                                      (p[k])+1..T)(x[i,j,t] == 0);

% At the end of the planning horizon, all stops are empty
constraint forall(i in nws)(z[i, T-1] == 0);
```

```
% If at time t y[1,t] vehicles have left station 1,
% then until they finish the journey, they cannot depart from other stops
constraint forall(i in 2..(n-1), t in 1..T-P[i])(if y[1, t] != 0
then sum(dt in 0..P[i]-1)(y[i, t + dt]) <= m - y[1, t] else true endif);
% Same for the first station
constraint forall(t in 1..(T - P[n]))(if y[1, t] != 0
then sum(dt in 1..P[n])(y[1, t+dt]) <= m - y[1, t] else true endif);

% Vehicles cannot arrive at stop i > 1 before departing
% from stop 1 at the beginning of the planning horizon
constraint forall(i in 2..(n-1))(forall(t in 1..P[i])(y[i,t] == 0));

% No departures if vehicle does not reach the final station
% within the planning horizon
constraint forall(t in (T-P[n]+1)..T)(y[1, t] == 0);

% Objective
solve minimize sum(i in 1..n-1, j in 1..T)(z[i, j]);

% Output
output[show(y[1,t]) ++ " " | t in Tw]
```

6.3 Приложение 3

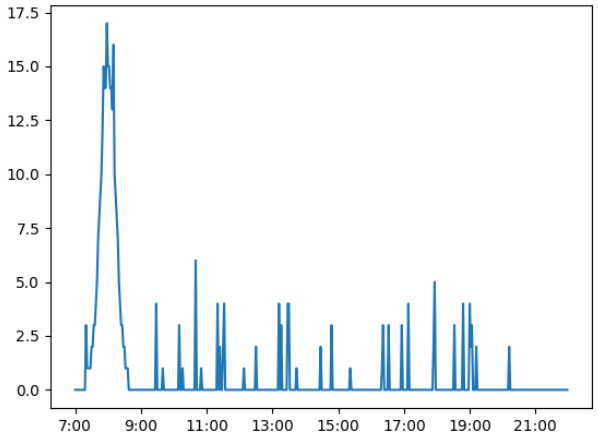

Рис. 1: Количество пассажиров, приходящих на остановку Красная площадь

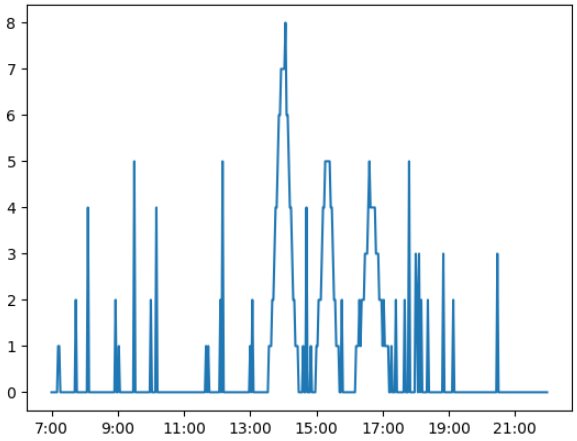

Рис. 2: Количество пассажиров, приходящих на остановку Студенческий городок