IEEE

II Международная научная конференция по проблемам управления в технических системах

Разработка метода выбора предпочтительного маршрута на основе субъективных критериев

Авторы: Я.А. Селиверстов, С.А. Селиверстов, О.Ю. Лукомская (*ИПТ РАН*); **К.В. Никитин** (*СПБПУ*); **В.А. Григориев** (*ИТМО*); **Е.О. Выдрина** (СПб МВД); **Г. Ю. Гергель** (*СПб ГУГА*)

АКТУАЛЬНОСТЬ

Актуальность темы исследования обусловлена :

- развитием интеллектуальных транспортных систем;
- совершенствованием систем городского транспортного мониторинга и городских информационно-аналитических систем;
- развитием персональных навигационных автомобильных и пешеходных систем с функцией прокладки маршрута по нескольким критериям.
- интеллектуализация городских логистических процессов, реализуемых на принципах Европейской концепции Смарт Сити.

Таким образом, построение систем управления процессами городской мобильности населения, анализ и учет персонального транспортного поведения становятся авангардом сегодняшнего времени.

ПОСТАНОВКА ЗАДАЧИ

Объект исследования.
Транспортное поведение городского населения

Предмет исследования.
Методы и алгоритмы прокладки маршрутов, методы поиска пути на графах, технология разработки программных приложений

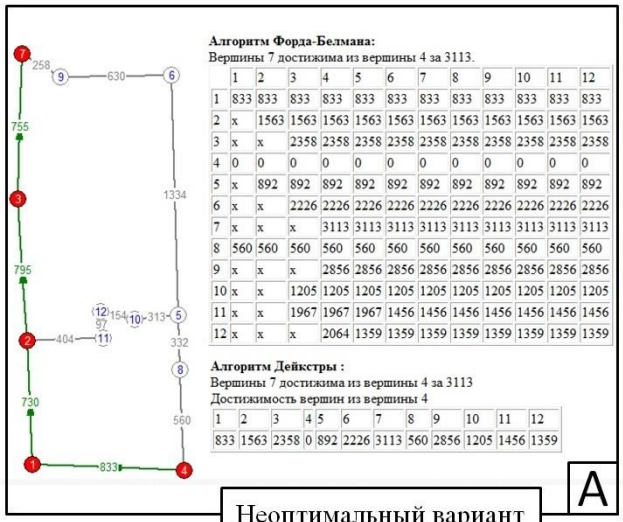
Цель исследования.
1. Разработка метода выбора предпочтительного маршрута на основе субъективных критериев.
2. Разработка метода селекционной фильтрации свойств маршрутов.
3. Разработка мобильного персонального планировщика (в разработке).

АЛГОРИТМ ДЕЙКСТРЫ

```
def dijkstra(graph, source):

    vertices, edges = graph
    dist = dict()
    previous = dict()

    for vertex in vertices:
        dist[vertex] = float("inf")
        previous[vertex] = None

    dist[source] = 0
    Q = set(vertices)

    while len(Q) > 0:
        u = minimum_distance(dist, Q)
        print('Currently considering', u, 'with a distance of', dist[u])
        Q.remove(u)

        if dist[u] == float('inf'):
            break

        n = get_neighbours(graph, u)
        for vertex in n:
            alt = dist[u] + dist_between(graph, u, vertex)
            if alt < dist[vertex]:
                dist[vertex] = alt
                previous[vertex] = u

    return previous
```

АЛГОРИТМ БЕЛМАНА ФОРДА

```
def bellmanFord(source, weights):
    '''
    This implementation takes in a graph and fills two arrays
    (distance and predecessor) with shortest-path (less cost/distance/metric)

    https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm
    '''
    n = weights.shape[0]

    # Step 1: initialize graph
    distance = np.empty(n)
    distance.fill(float('Inf'))      # At the beginning, all vertices have a weight
    predecessor = np.empty(n)
    predecessor.fill(float('NaN'))   # And a null predecessor

    distance[source] = 0             # Except for the Source, where the Weight is zero

    # Step 2: relax edges repeatedly
    for _ in xrange(1, n):
        for (u, v), w in np.ndenumerate(weights):
            if distance[u] + w < distance[v]:
                distance[v] = distance[u] + w
    predecessor[v] = u

    # Step 3: check
    for negative - weight cycles
    for (u, v), w in np.ndenumerate(weights):
        if distance[u] + w < distance[v]:
            raise ValueError("Graph contains a negative-weight cycle")

    return distance, predecessor
```



Алгоритм Форда-Белмана:
Вершины 7 достижима из вершины 4 за 3113.

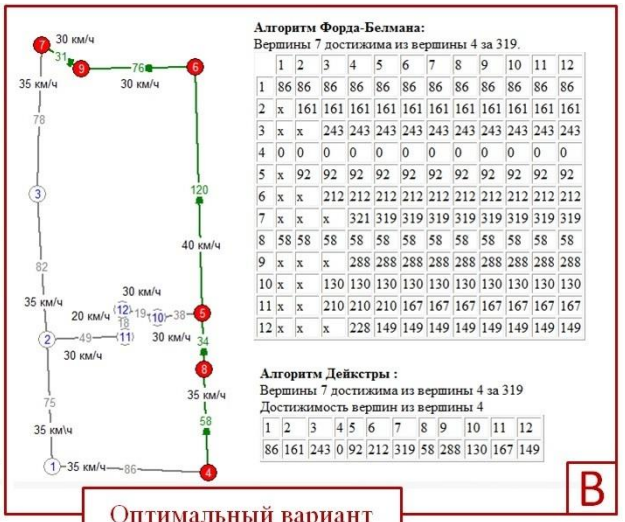
	1	2	3	4	5	6	7	8	9	10	11	12
1	833	833	833	833	833	833	833	833	833	833	833	833
2	x	1563	1563	1563	1563	1563	1563	1563	1563	1563	1563	1563
3	x	x	2358	2358	2358	2358	2358	2358	2358	2358	2358	2358
4	0	0	0	0	0	0	0	0	0	0	0	0
5	x	892	892	892	892	892	892	892	892	892	892	892
6	x	x	2226	2226	2226	2226	2226	2226	2226	2226	2226	2226
7	x	x	x	3113	3113	3113	3113	3113	3113	3113	3113	3113
8	560	560	560	560	560	560	560	560	560	560	560	560
9	x	x	x	2856	2856	2856	2856	2856	2856	2856	2856	2856
10	x	x	1205	1205	1205	1205	1205	1205	1205	1205	1205	1205
11	x	x	1967	1967	1967	1456	1456	1456	1456	1456	1456	1456
12	x	x	x	2064	1359	1359	1359	1359	1359	1359	1359	1359

Алгоритм Дейкстры :
Вершины 7 достижима из вершины 4 за 3113
Достижимость вершин из вершины 4

1	2	3	4	5	6	7	8	9	10	11	12
833	1563	2358	0	892	2226	3113	560	2856	1205	1456	1359

A

Неоптимальный вариант



Алгоритм Форда-Белмана:
Вершины 7 достижима из вершины 4 за 319.

	1	2	3	4	5	6	7	8	9	10	11	12
1	86	86	86	86	86	86	86	86	86	86	86	86
2	x	161	161	161	161	161	161	161	161	161	161	161
3	x	x	243	243	243	243	243	243	243	243	243	243
4	0	0	0	0	0	0	0	0	0	0	0	0
5	x	92	92	92	92	92	92	92	92	92	92	92
6	x	x	212	212	212	212	212	212	212	212	212	212
7	x	x	x	321	319	319	319	319	319	319	319	319
8	58	58	58	58	58	58	58	58	58	58	58	58
9	x	x	x	288	288	288	288	288	288	288	288	288
10	x	x	130	130	130	130	130	130	130	130	130	130
11	x	x	210	210	210	167	167	167	167	167	167	167
12	x	x	x	228	149	149	149	149	149	149	149	149

Алгоритм Дейкстры :
Вершины 7 достижима из вершины 4 за 319
Достижимость вершин из вершины 4

1	2	3	4	5	6	7	8	9	10	11	12
86	161	243	0	92	212	319	58	288	130	167	149

B

Оптимальный вариант

МЕТОДА ВЫБОРА ПРЕДПОЧТИТЕЛЬНОГО МАРШРУТА НА ОСНОВЕ СУБЪЕКТИВНЫХ КРИТЕРИЕВ

Путь предпочтительного следования пользователя— маршрут следования пользователя, пролегающий в области его предпочтительного следования, характеристики которого удовлетворяют его персональным предпочтениям.

Функция транспортной полезности пользователя Φ_{hid} —принимает предпочтительное значение B_t^{Pref}, в том случае, когда характеристики пути $Ch\ (\rho^s\ ;\ \rho^d)$ и свойства входящих в него объектов $Э(\rho^s;\ \rho^d)$ удовлетворяют персональным предпочтениям пользователя на всем протяжении пути из места отправления ρ^s в место прибытия ρ^d .

$$\Phi_{h_{id}} = \lim B_t^{h_{id}}\left(\rho^s_{h_{id}};\rho^d_{h_{id}}\right) = B_t^{Pref}\left(\rho^s_{h_{id}};\rho^d_{h_{id}}\right)$$

$$\left(Ch_{h_{id}}^{Pref} \overset{\Phi^{\Gamma}\left[\rho^s_{h_{id}};\rho^d_{h_{id}}\right]}{\cap} CH\right) = Э_{h_{id}}^{Pref}\left(\rho^s_{h_{id}};\rho^d_{h_{id}}\right) \quad (1)$$

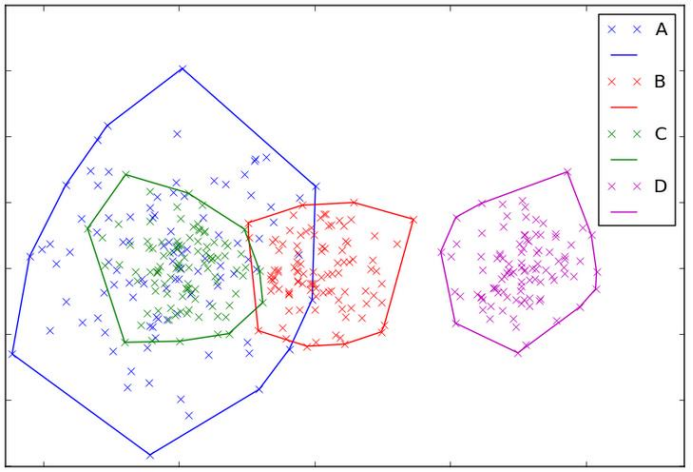
Поочередное пересечение множества предпочтений Ch_{hid}^{Pref} пользователя с множествами свойств $Ch(m_{Nm})$доступных маршрутов, даст область допустимых маршрутов следования M_{id}^{Ch}, а подсчет кардинального числа $Card(M^{id}_{Nm})$ определит степень их предпочтительности.

$$M_{id}^{Ch} = \left(Ch_{h_{id}}^{Pref} \bigcap_{[\rho^s_{h_{id}};\rho^d_{h_{id}}]} Ch(m_1) \right) \Rightarrow Card\left(M_1^{id}\right)$$

$$\cdots\cdots\cdots

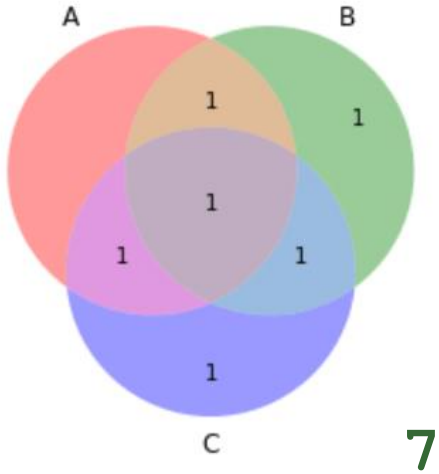
ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

```
import numpy as np
import matplotlib.nxutils
import scipy.spatial

def find_convex_hull(points):
  triangulation = scipy.spatial.Delaunay(points)
  unordered = list(triangulation.convex_hull)
  ordered = list(unordered.pop(0))
  while len(unordered) > 0:
    next = (i for i, seg in enumerate(unordered) if ordered[-1] in seg).next()
    ordered += [point for point in unordered.pop(next) if point != ordered[-1]]
  return points[ordered]
```




```
def inclusion(points_a, points_b):
  ch_a = find_convex_hull(points_a)
  return (1.0 * matplotlib.nxutils.points_inside_poly(points_b, ch_a)).mean()
```

```
from matplotlib import pyplot as plt
import numpy as np
from matplotlib_venn import venn3, venn3_circles
plt.figure(figsize=(4,4))
vd = venn3(subsets=(1, 1, 1, 1, 1, 1, 1), set_labels = ('A', 'B', 'C'))

#Move the numbers in the circles
vd.get_label_by_id("100").set_x(1.55)

plt.show()
```

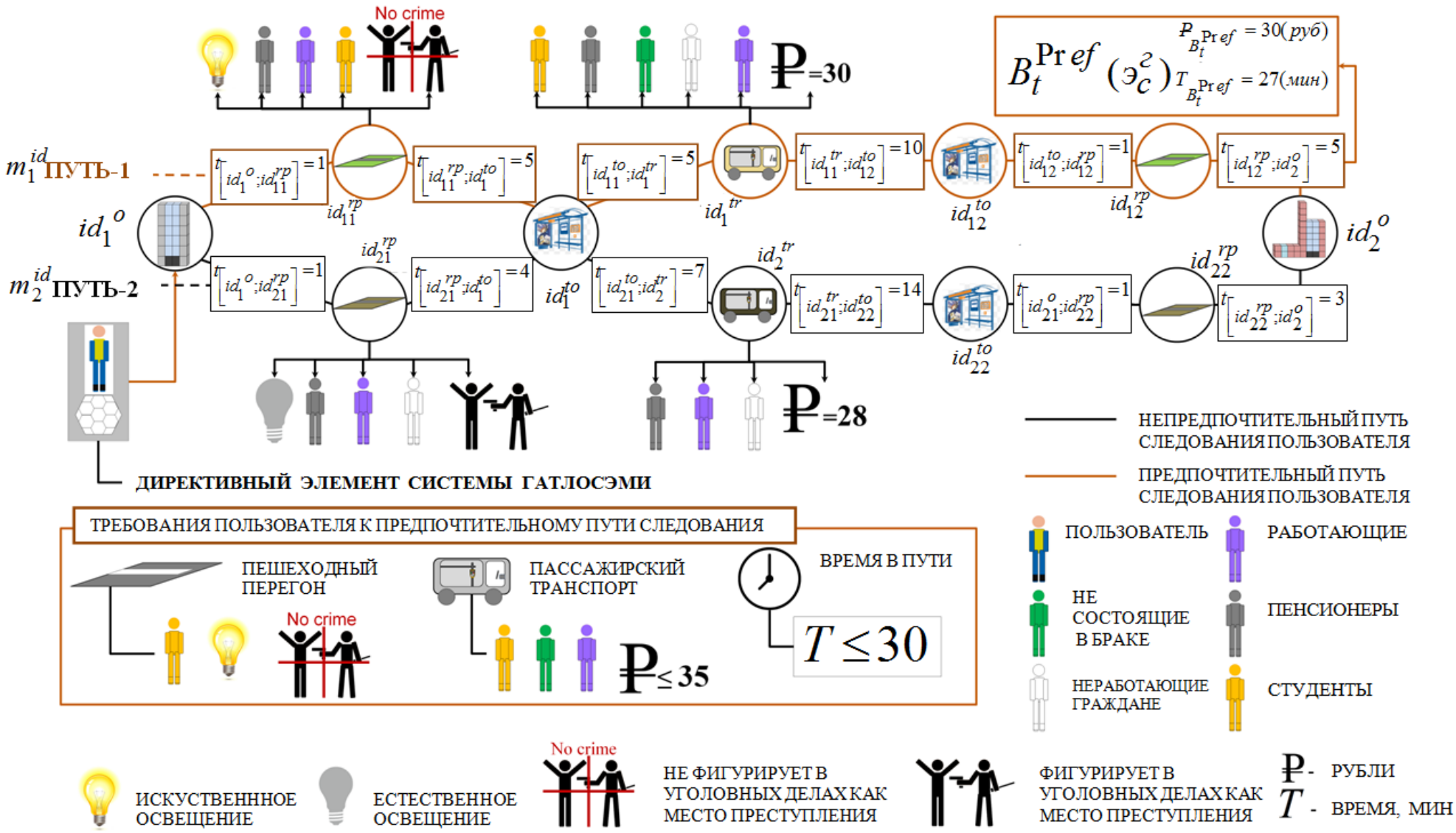


Пример 1. Пользователю с номером id необходимо из места отправления ρ^S прибыть в место назначения ρ^D по предпочтительному маршруту следования. Известно, что из места отправления в место назначения можно добраться 2 разными маршрутами: m_1^{id} (путь 1) и m_2^{id} (путь 2). Свойства маршрутов следования, на которых задан строгий порядок представлены в Таблице 1. Данные о городском населении приведены в соответствии с классификатором ОКИН (Общероссийский классификатор информации о населении). Требуется определить предпочтительный маршрут следования пользователя h_{id}?

Таблица 1. Свойства маршрутов следования

№	Свойства маршрута (СН)	**СН(m_1)**	**СН(m_2)**	**P [pref]**
P_1	Стоимость, руб	30	28	<35
P_2	Время, мин	27	30	<30
P_3	Криминальность	0	5	0
P_4	Искусственное освещение	есть	нет	есть
P_5	Пенсионеры	15%	60%	<20%
P_6	Студенты	60%	5%	>50%
P_7	Не состоящие в браке	70%	10%	>60%
P_8	Работающие	30%	45%	<40%

ПУТЬ ПРЕДПОЧТИТЕЛЬНОГО СЛЕДОВАНИЯ ПОЛЬЗОВАТЕЛЯ



Решение. Определим предпочтительный путь следования по формуле, как поочередное пересечение множества предпочтений пользователя с множества свойств маршрута следования. Расчеты сведем в Таблицу 2.

Таблица 2. Определение предпочтительного маршрута следования.

№	Свойства маршрута (СН)	$Ch_{h_{id}}^{\Pr ef} \cap Ch(m_1)$	$Ch_{h_{id}}^{\Pr ef} \cap Ch(m_2)$	P [pref]
P_1	Стоимость, руб	■	■	<35
P_2	Время, мин	■	□	<30
P_3	Криминальность	■	□	0
P_4	Искусственное освещение	■	□	есть
P_5	Пенсионеры	■	■	<20%
P_6	Студенты	■	□	>50%
P_7	Не состоящие в браке	□	□	>60%
P_8	Работающие	□	■	<40%
	Подсчет мощности множества	$Card\left(m_1^{id}\right)=6$	$Card\left(m_2^{id}\right)=3$	

! $M_{id}^{\Pr ef} = \max\left[Card\left(m_1^{id}\right), Card\left(m_2^{id}\right)\right] = \max[6;2] = 6 \Rightarrow M_{id}^{\Pr ef} = m_1^{id}$

СЕЛЕКЦИОННАЯ ФИЛЬТРАЦИЯ СВОЙСТВ МАРШРУТОВ

Выбор маршрута следования, обладающего одними определенными свойствами и не обладающего другими.

В основу селекционной фильтрации маршрутов целесообразно положить общий метод просеивания Сильва – Сильвестра

Пусть элементы $A_i \subset A$ обладают свойством P_i, $i=1,2,\ldots,n$, тогда подмножество $A_{i_1} \bigcap A_{i_2} \bigcap \ldots \bigcap A_{i_k}$ - обладает свойством $P_{i_1} \wedge P_{i_2} \wedge \ldots \wedge P_{i_k}$. Таким образом, если элементы А могут обладать n различными свойствами, то число элементов A, обладающих k указанными свойствами и не обладающих n-k задаются оператором f_{sift}:

$$f_{sift} : Card\left(A_1 \bigcap \ldots \bigcap A_k \bigcap \overline{A}_{k+1} \bigcap \overline{A}_{k+2} \bigcap \ldots \bigcap \overline{A}_n\right) = Card\left(A_1 \bigcap \ldots \bigcap A_k\right) - Card\left(A_1 \bigcap \ldots \bigcap A_k \bigcap A_{k+1}\right) -$$
$$\ldots - Card\left(A_1 \bigcap \ldots \bigcap A_k \bigcap A_n\right) + Caard\left(A_1 \bigcap \ldots \bigcap A_k \bigcap A_{k+1} \bigcap A_{k+2}\right) + \ldots$$
$$+ Card\left(A_1 \bigcap \ldots \bigcap A_k \bigcap A_{n-1} \bigcap A_n\right) + (-1)^{n-k} Card\left(A_1 \bigcap \ldots \bigcap A_k \bigcap A_{k+1} \bigcap A_{k+2} \bigcap \ldots \bigcap A_n\right) \quad (4)$$

где A – конечное множество; $A_1 \subset A$, $\overline{A_1}$ - дополнение A_1 по отношению к A; $Card\,A$ – число элементов в множестве А.

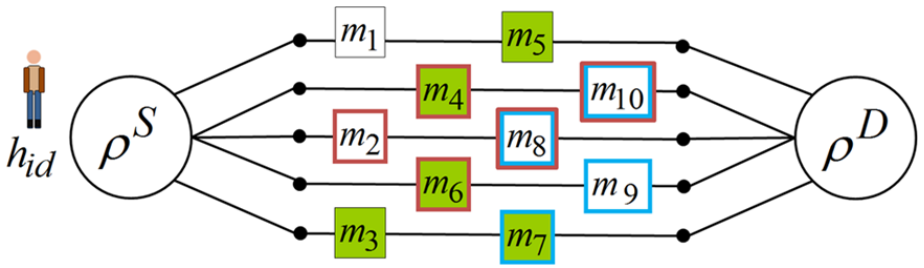
В случае не выделенных множеств, когда множества не выделяются, а фиксируется только число свойств, которыми обладают их элементы, оператор f_{sift} имеет вид:

$$W(k) = w(k) - C_{k+1}^{1} w(k+1) + C_{k+2}^{2} w(k+2) - \ldots + (-1)^{n-k} C_{n}^{n-k} w(n) \quad (5)$$

где $W(k)$ - число элементов в множестве A удовлетворяющих k свойствам.

Рассмотрим работу селекционного фильтра свойств маршрутов жителя на следующем примере.

Пример 2. Пользователь h_{id} с номером id необходимо из места отправления ρ^S прибыть в место назначения ρ^D по предпочтительному маршруту следования. Известно, что из места отправления в место назначения можно добраться 10 разными маршрутами. Предпочтения пользователя h_{id} следующие: стоимость поездки должна быть меньше 40 руб., $P_1 < 40\,руб$; время поездки должно быть меньше 1 часа, $P_2 < 1ч$; на маршруте не должно быть зарегистрировано ни одного криминального случая, $P_3 = 0$.



Требуется определить предпочтительный маршрут следования пользователя с использованием селекционного фильтра.

Решение: Дано множество маршрутов M такое что: $M = \{m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8, m_9, m_{10}\}$, где $\{m\}$- множество различных маршрутов обладающих свойствами (Таблица 3).

Таблица 3. Свойство транспортных средств

№	Свойства маршрута	Количество маршрутов									
		m_1	**m_2**	**m_3**	**m_4**	**m_5**	**m_6**	**m_7**	**m_8**	**m_9**	**m_{10}**
P_1	**Стоимость < 40руб**		●		●		●		●		●
P_2	**Время поездки>1 ч**							☼	☼	☼	☼
P_3	**Криминальность**			◯	◯	◯	◯	◯			

Подсчитаем число маршрутов M, обладающих свойством: $P_1 \wedge$ не $P_2 \wedge$, не P_3, т.е. $P_1 \wedge \overline{P}_2 \wedge \overline{P}_3$.

Обозначим подмножества, соответствующие свойствам P_1, P_2, P_3, через A_1, A_2, A_3. Тогда, по формулам (4) и (5) имеем:

$$Card\left(A_1 \cap \overline{A}_2 \cap \overline{A}_3\right) = Card\left(A_1\right) - Card\left(A_1 \cap A_2\right) - Card\left(A_1 \cap A_3\right) + Card\left(A_1 \cap A_2 \cap A_3\right)$$

Просеиваем M через P_1: $Card(A_1) = 5$

Просеиваем A_1 через P_2 и P_3: $Card(A_1 \cap A_2) = 2$, $Card(A_1 \cap A_3) = 2$

Просеиваем $A_1 \cap A_2$ через P_3: $Card(A_1 \cap A_2 \cap A_3) = 0$

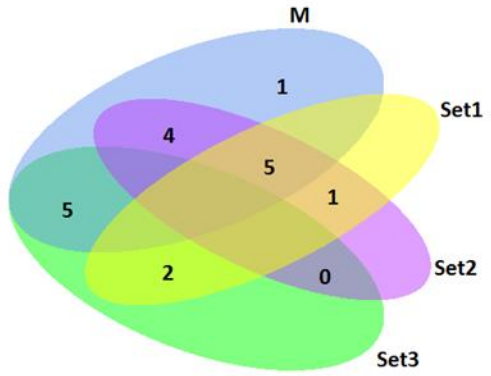
В итоге: $Card(A_1 \cap \overline{A}_2 \cap \overline{A}_3) = 5 - 2 - 2 + 0 = 1$

Таким образом, через процедуру просеивания мы определили, что только маршрут m_2 обладает свойством P_1 и не обладает свойствами P_2 и P_3, то есть является наиболее безопасным маршрутом и предпочтительным по стоимостному и возрастному критериям.

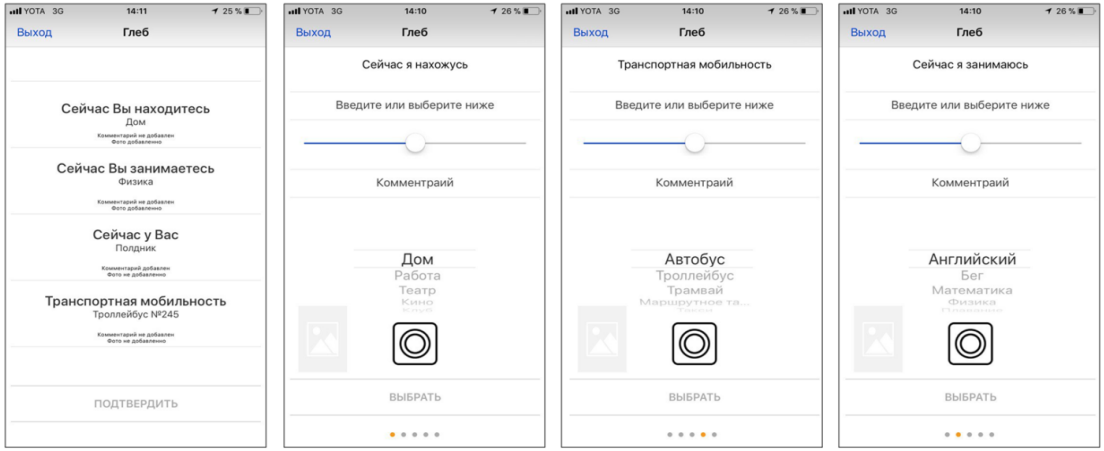
ПРОГРАММНАЯ РЕАЛИЗАЦИЯ (ПРИМЕР2)

```
#of items with m properties:
E(m) = sum { (-1)^(r-m) * Choose(r,m) * W(r) }
Where:
W(r) = summation of all intersections of size i, or formally:
W(r) = sum { W(p_k_1,...,p_k_r) | 1<=k_1<k_2<...<k_r<=n }
W(pk1,...,pkr) = |X_k1 [intersection] X_k2 [intersection] .... [intersection] X_kr|
```

```
from matplotlib_venn import venn3, venn3_circles
import numpy as np
M= set(['m1', 'm2','m3', 'm4', 'm5', 'm6', 'm7', 'm8', 'm9','m10'])
set1 = set(['m2', 'm4', 'm6', 'm8','m10'])
set2 = set(['m7', 'm8', 'm9', 'm10'])
set3 = set(['m3', 'm4', 'm5', 'm6', 'm7'])

venn3([M, set1, set2, set3], ('M', 'Set1', 'Set2', 'Set3'))

plt.show()
```



МОБИЛЬНЫЙ ПЕРСОНАЛЬНЫЙ ПЛАНИРОВЩИК



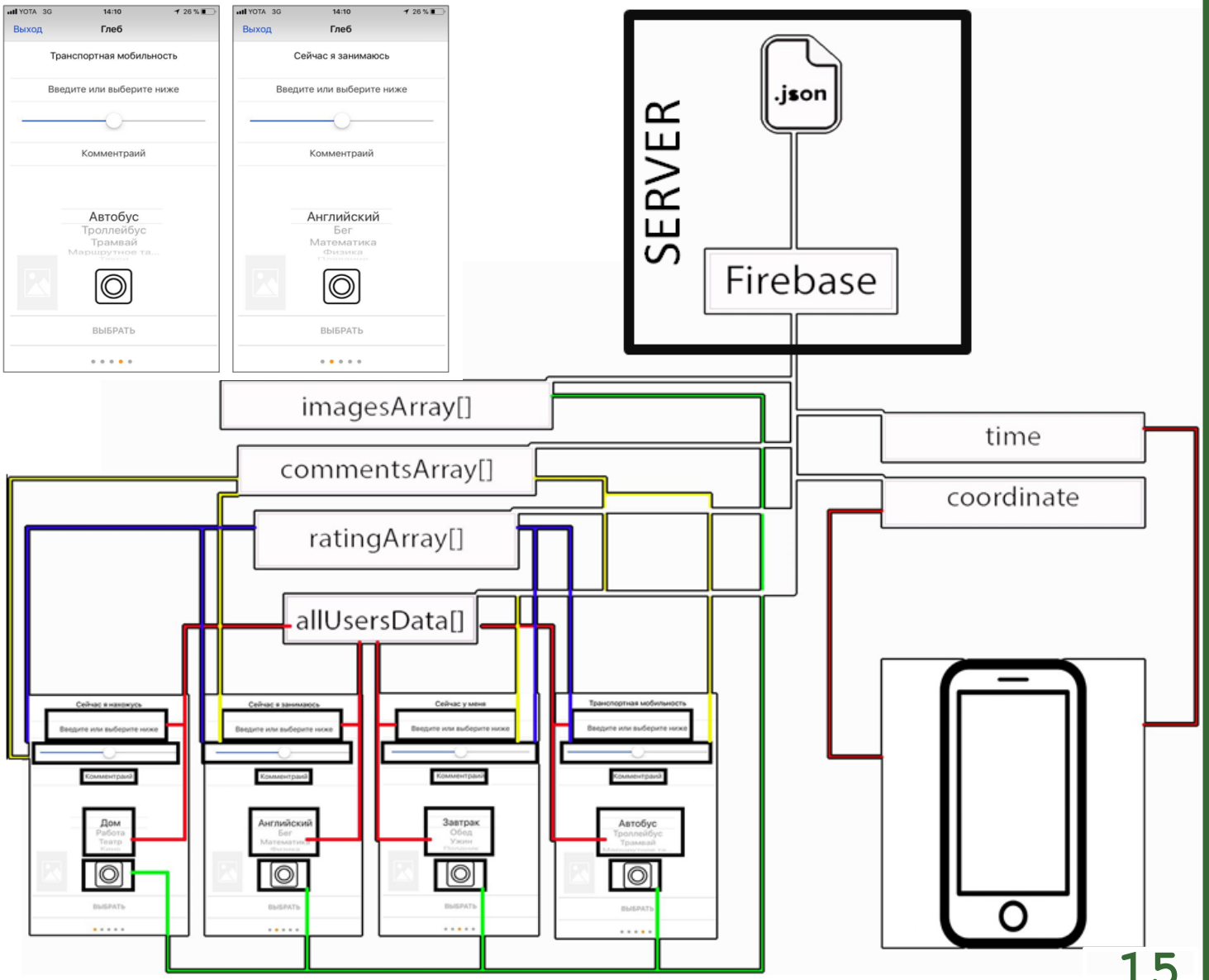
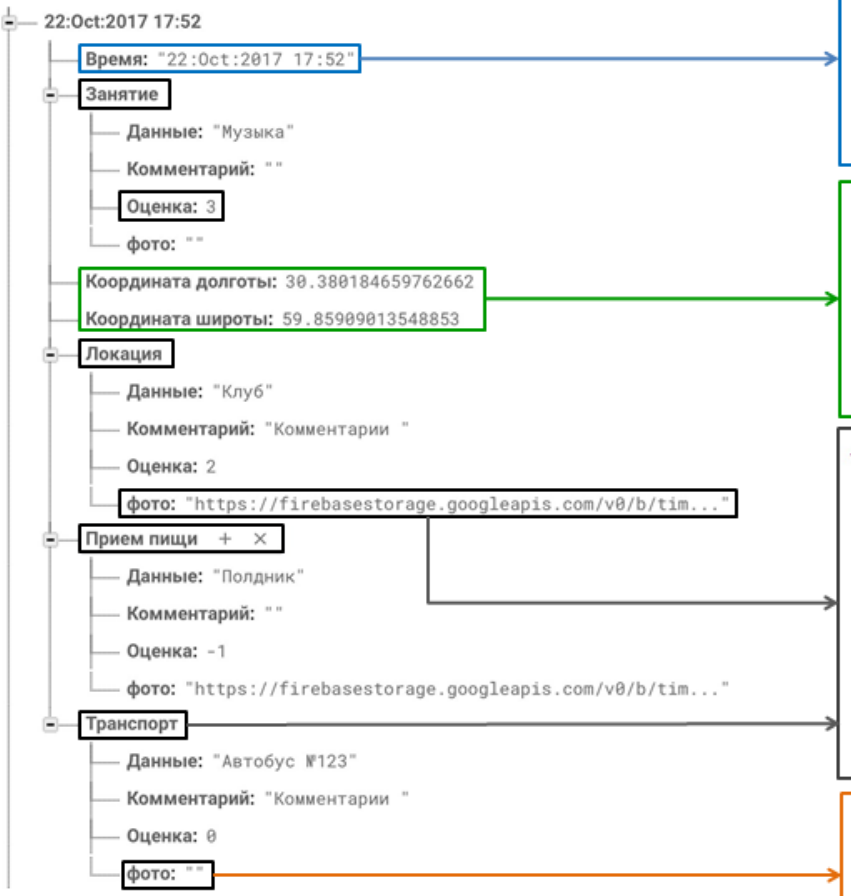


СХЕМА МОБИЛЬНОГО ПРИЛОЖЕНИЯ

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ В СРЕДЕ XCODE




```
let seconds = NSDate().timeIntervalSince1970
let time = NSDate(timeIntervalSince1970: seconds)
let dateFormater = DateFormatter()
dateFormater.timeZone = NSTimeZone(name: "EET") as! TimeZone
// dateFormater.timeZone = "America/New_York"
dateFormater.dateFormat = "dd:MMM:yyyy HH:mm"
```

```
func locationManager(_ manager: CLLocationManager, didUpdateLocations locations: [CLLocation]) {
    let location = locations[0]

    let myLocation:CLLocationCoordinate2D = CLLocationCoordinate2DMake(location.coordinate.latitude,
        location.coordinate.longitude)

    coordinate = CLLocation(latitude: myLocation.latitude, longitude: myLocation.longitude)

}
```

```
func handleChoiceButton(){



    pageController?.allUsersData[(pageController?.pageControl.currentPage)!] = ""
    pageController?.allUsersData[(pageController?.pageControl.currentPage)!] = textField.text!
    textField.text = ""

    print((pageController?.pageControl.currentPage)!)
    pageController?.commentsArray[(pageController?.pageControl.currentPage)!] = commentTextField.text!
    commentTextField.text = ""

    pageController?.ratingArray[(pageController?.pageControl.currentPage)!]  = Int(placeRating.value)
    placeRating.value = 0

    pageController?.handleArray()
```

```
func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info:
    [String : Any]) {
    if let pickedImage = info[UIImagePickerControllerOriginalImage] as? UIImage {
        imagesArray[pageControl.currentPage] = pickedImage
        print(pageControl.currentPage)
        imagesArrayFlag[pageControl.currentPage] = true
        DispatchQueue.main.async(execute: {
            self.collectionView.reloadData()
        });
    }
    picker.dismiss(animated: true, completion: nil)
```

Отправка данных на сервер

```
let values = ["Локация": ["Данные": allUsersData[0], "Комментарий": commentsArray[0], "фото":
    imagesArrayUrl[0], "Оценка": ratingArray[0]], "Занятие": ["Данные": allUsersData[1],
    "Комментарий": commentsArray[1], "фото": imagesArrayUrl[1], "Оценка": ratingArray[1]], "Прием
    пищи": ["Данные": allUsersData[2], "Комментарий": commentsArray[2], "фото": imagesArrayUrl[2],
    "Оценка": ratingArray[2]], "Транспорт": ["Данные": allUsersData[3], "Комментарий":
    commentsArray[3], "фото": imagesArrayUrl[3], "Оценка": ratingArray[3]], "Время":
    dateFormater.string(from: time as Date), "Координата долготы": coordinate.coordinate.longitude,
    "Координата широты": coordinate.coordinate.latitude] as [String : Any]
```

Выводы

В настоящей работе разработано алгоритмическое построение пути субъективного предпочтительного следования.

В основу метода определения предпочтительного пути была положена теоретико-множественная операция пересечения, а в основу селекционной фильтрации маршрутов был положен общий метод просеивания Сильва – Сильвестра. Адекватность разработанных методов была продемонстрирована на практических примерах. Таким образом, впервые осуществлен переход к персонифицированному учету транспортной мобильностью населения.

Реализация рекомендательного управления на основе выбора субъективных предпочтительных маршрутов следования позволяет перейти создания персональных городских навигационных систем поддержки принятия решений в городской мобильности.

Спасибо за внимание!

ЛАБОРАТОРИЯ РАЗВИТИЯ ИНТЕЛЛЕКТУАЛЬНЫХ ТРАНСПОРТНЫХ СИСТЕМ
Кандидат технических наук, старший научный сотрудник,
Селиверстов Ярослав Александрович,
Email: seliverstov-yr@mail.ru

РАБОТА ВЫПОЛНЕНА В ФГБУН «ИНСТИТУТ ПРОБЛЕМ ТРАНСПОРТА ИМ. Н.С.СОЛОМЕНКО РОССИЙСКОЙ АКАДЕМИИ НАУК»
Адрес: 199178, Россия, Санкт-Петербург, В.О. 12-я линия, дом 13.
т.факс: (812) 323-2954
Официальный сайт: www.iptran.ru