ШИФР “ТРАФІК”

Конкурсна робота

на тему:

“АЛГОРИТМ ОПТИМАЛЬНОГО МІСЬКОГО МАРШРУТУ”

2018

Зміст

Вступ

Дане дослідження ставить за мету побудову алгоритму оптимального індивідуального маршруту для кожного транспортного засобу (ТЗ) у великому місті, миттєво здійснюючи корекцію маршруту при зміні дорожньої ситуації. Загалом процедура регулювання транспортних потоків здійснюється за рахунок динамічної взаємодії між центральним пунктом керування трафіком (ЦПКТ) та кожним ТЗ, що в свою чергу задав початкові та кінцеві координати маршруту. Процедура здійснюється в режимі реального часу. Певним чином запропонована технологія подібна до добре відомої системи GPS-навігації. Проте особливість представленої системи навігації полягає в тому, що дана система працює в режимі щосекундної корекції маршруту, передаючи кожному водієві голосові команди щодо руху по оптимальному маршруту. GPS-навігація, навпаки, констатує дорожню ситуація в місті постфактум, інформуючи водіїв про затори на міських трасах. Отже, на противагу GPS-технології, запропонована програма проводить детальний аналіз дорожньої ситуації на кожному перехресті та ділянках дороги між перехрестями по всьому місту та прокладає маршрут, враховуючи трафік-ситуації на кожен конкретний момент часу. Кінцевою метою цього дослідження є синхронізація потоків ТЗ, оптимальне використання усіх дорожніх шляхів міста, запобігання утворенню заторів та супровід кожного ТЗ до кінцевої точки маршруту в розрахунку на те, що затрачений на поїздку час буде мінімальним. При такому підході затори в містах взагалі виникати не будуть!

Розділ 1. Постановка проблеми

Це дослідження є продовженням роботи «Автоматизована система керування рухом транспортних засобів в межах міста» та її логічним доповненням [1]. У зазначеній роботі розглядалась проблема проїзду перехресть з використанням п’єзокристалічних датчиків.

Зазначимо, що найголовнішою проблемою на сьогоднішній день у великих містах є масові затори, що тягнуть за собою збільшення часу проведеного за кермом, запізнення на певні заходи та загалом втомленість водіїв, яке може призвести до дорожньо-транспортних пригод (ДТП). Тому виникає ряд запитань. Як уникнути такого негативного явища як затори? Як організувати проїзд кожного ТЗ, кількість яких у великому місті може сягати мільйонів? Як побудувати маршрут кожного автомобіля до кінцевого пункту прибуття, вказаного водієм, з розрахунком того, щоб поїздка зайняла найменше часу? Технологія, що пропонується у даному дослідженні, дозволяє ефективно вирішити поставленні проблеми та в певній мірі дає відповіді на поставлені запитання.

Розділ 2. Оцінка базових моделей

Вище описана проблема підіймається на один із вищих ступенів серед найрозповсюдженіших проблем світу. Її намагаються вирішити багато великих міжнародних компаній, що займаються розробкою технічних засобів організації дорожнього руху [2]. Технологія, що розглядається у даному дослідженні, напряму пов'язана із навігацією. Найбільш близькою до неї є система навігації типу GPS/GLONASS [3]. Кожного року різні компанії модифікують систему GPS-навігації. Сьогодні автомобільне навігаційне обладнання та придорожня навігаційна інфраструктура мають багато різних функціональних можливостей, зокрема можуть повідомляти про затори на вулицях міста та будувати маршрут об'їзду таких місць із збереженням мінімальної кількості витраченого для об'їзду часу. Інтеграція техніки з автомобілем стає дедалі глибшою, що дозволяє задавати кожному водієві голосові команди, які допомагають слідувати по оптимальному маршруту та не відхилятися від нього. Особливості виду взаємодії між дорожньою інфраструктурою та автомобілем детально описані в роботі [4]. Тут представлена технологія взаємодії між автомобілем та дорожньою інфраструктурою з допомогою так званих точок доступу (access points), розташованих вздовж автомобільної дороги та на перехрестях. Також близькою до нашого дослідження є робота [5]. В основі технології лежать чотири складових: світлофори, детектори черги (queue detectors), дорожні відеокамери і центральна контрольна система (central control system). Кожні дві секунди система контролює ситуацію щодо зміни фаз горіння світлофорів і завдяки цьому досягається оптимізація трафіку. В роботі [6] автори розглядають проблему розташування датчиків, тому що розташування їх суттєвим чином впливає на те, які транспортні потоки реєструються і тому можуть бути керованими. В дослідженні [7] застосовується модифікована модель стільникових автоматів (modified cellular automata model) Для вивчення процесів взаємодії між автомобілями проведено дослідженню процесів обгону шляхом вивчення даних про трафік ТЗ. Здійснено моделювання потоків ТЗ. В

дослідженні [8] застосовується «розумна» мережева імітаційна модель; в якості об’єкта досліджень використана центральна та західна частина міста Сінгапур. На карті міста відтворено автомобільні розв’язки, комерційні зони, перехрестя, автомагістралі, звичайні дороги і автобусні зупинки. Для того щоб провести імітаційне дослідження застосовувалась імітаційна модель PARAMICS. Для реєстрації потоків ТЗ дослідники використали камери спостереження (surveillance cameras) та петлеві детектори (loop detectors).

Ідея покращення трафіка, як можливий варіант, є координоване регулювання автомобільних потоків за допомогою бездротового зв'язку між автомобілями, як було запропоновано в роботі [9]. З розвитком автотранспорту все більших оборотів набуває теорія стільникових автоматів. В наведеній вище роботі моделювання рухом через перехрестя ведеться на платформі NetLogo, що є багатоагентним програмним середовищем для моделювання різних динамічних процесів.

Винахід [10] відноситься до інтелектуальної системи керування режимом роботи світлофорів через двосторонній обмін інформацією з ЦПКТ. Тут описаний процес перетворення інформації світлофорів на бездротові сигнали та реалізація інтелектуального управління світлофорною системою. Робота ЦПКТ полягає в наданні інформації водіям автомобілів, управління міськими дорогами та передача електронних карт маршрутів. Щоб водії своєчасно отримували інформацію про перемикання світлофорів, використано інтелектуальну систему управління роботою світлофорів на базі інтелектуального терміналу. Система світлофорів автоматично визначає кількість автомобілів на дорогах, щоб миттєво змінювати час проїзду для автомобілів різних напрямків. Система та спосіб, передбачені цим винаходом, допомагають водіям оптимізувати маршрути руху. Проблеми паркінгу як різновид проблеми трафіку розглянуті в дослідженні [11]. Технологія запропонована компанією Siemens. Система контролює завантаженість вулиць і передає інформацію автомобілістам, використовуючи для кожної окремої стоянки інформацію, зчитувану за допомогою наземних датчиків або на основі

кількості проданих паркувальних дозволів. В усіх випадках система направляє водіїв одразу ж на наявні місця для паркування, що дає змогу зменшити навантаження на трафік та запобігти переповненню вулиць. Має місце інтеграція в загальну систему керування трафіком. Така система дає змогу використовувати бази даних паркування для надання автомобілістам рекомендацій щодо маршрутизації вже при в'їзді в межі міста.

Робота [12] надає результати досліджень трафіку, які були проведенні в штаті Юта (США). Розглянута система показників ефективності регулювання руху потоків ТЗ на основі аналізу даних, які отриманні із спеціальних мікрохвильових датчиків. В дослідженні [13] для вирішення проблем, пов'язаних із дорожнім рухом, була використана рідинна динаміка. Таке дослідження представляє методологію для моделювання ситуацій, пов'язаних із дорожнім трафіком. Проблему заторів може полегшити розроблена теорія в [14] технологія. Тему взаємодії приватних автомобілів та громадського транспорту розглянуто в дослідженні [15]. Щоб розв'язати проблему такої взаємодії автори вводять поняття двотипних (бімодальних) міських мереж (bi-modal urban networks). Для організації ефективної взаємодії вказаних типів ТЗ вводяться у розгляд бімодальна Макроскопічна Фундаментальна Діаграма (МФД), яка моделює змішаний трафік ТЗ згаданих видів. За результатами видно, що ця теорія може: (а) зменшити затори в мережі; (б) поліпшити показники трафіку автобусів з точки зору часу проїзду маршруту руху; (в) знизити рівень скупченості ТЗ на критичних ділянках транспортної мережі. В якості об'єкта досліджень вибрана транспортна мережа Сан-Франціско.

В дослідженні [2] мова йде про організацію дорожнього руху за допомогою спеціальних пристроїв, зокрема п'єзокристалічних датчиків. П'єзоелектричний датчик являє собою пристрій, який використовує п'єзоелектричний ефект для вимірювання тиску, прискорення, температури, напруги або сили шляхом перетворення їх в електричний заряд. Детальніше про них можна дізнатись, наприклад, у роботі [18]. Використовуються повністю адаптивні системи управління трафіком. Повноцінна адаптивна технологія

управління трафіком (АТАК) – це операційна система, в якій оптимізуються до такі параметрами,як обсяг трафіку, черговість і нові умови в реальному часі з метою мінімізації часу проходження ТЗ та середніх зупинок в дорожній мережі. Більше можна дізнатись в [19]. Карта дорожньої щільності представлена у дослідженні [20]. Про роботу контролера сигналу руху EXPERTRA 2A9 можна дізнатись із [21]. З метою уникнення перевантажень, забезпечення пріоритету для аварійних автомобілів і скорочення середнього часу очікування (AWT) ТЗ на перехрестях створена система навігації, описана в [22]. Системне дослідження управління трафіком з використанням бездротових датчиків WSN та режим інтелектуального управління світлофором описані в [23].

Загалом, координована взаємодія між ТЗ та ЦПКТ із зворотнім зв’язком має особливі перспективи з огляду на існуючу тенденцію поширення автомобілів з так званими системами автопілота та появою на дорогах безпілотних ТЗ, що у майбутньому дозволить вивести керування транспортними потоками на кардинально новий, більш ефективний рівень зі скоординованими діями всіх учасників руху та упередженням проблемних ситуацій.

Розділ 3. Результати досліджень та їх імплементація

Основна мета нашого дослідження полягає в тому, щоб побудувати оптимальні маршрути для кожного автомобіля і, таким чином, синхронізувати потоки ТЗ. Іншими словами, ставиться задача провести кожен ТЗ по місту по оптимальному маршруту з урахуванням можливих змін маршруту, який коригується кожні 10 секунд. Система регулювання трафіку всієї сукупності автомобілів на вулицях міста повинна прокладати в режимі реального часу динамічний і оптимальний маршрут кожному, без виключення, автомобілю. При цьому ТЗ замовляє лише кінцеву точку маршруту ЦПКТ, а стартова позиція фіксується автоматично при підключенні ТЗ до ЦПКТ, який містить в собі центральний комп'ютер, що використовує комп'ютерну програму, зміст якої буде викладено нижче, та співпрацює з кожним ТЗ, передаючи водієві голосові команди щодо маршруту руху до кінцевого пункту, заявленого водієм, як при звичайній GPS-навігації. Особливістю є те, що програма аналізує ситуацію на кожному перехресті міста, на ділянках дороги між цими перехрестями та по всьому місту в цілому. Відповідно до цього аналізу програма прокладає оптимальний маршрут, враховуючи дорожній стан на тій чи іншій частині шляху у конкретний момент часу.

Взагалі виконання поставленого завдання можна реалізувати двома варіантами. Першим із них є реалізація у комп'ютерній програмі алгоритму Флойда-Воршела. Другим варіантом є реалізація алгоритму Дейкстри. Обидва способи можна використати для пошуку оптимального шляху. Розглянемо обидва варіанти більш детально у нижче викладеному матеріалі.

Технологія є автоматизованою інтелектуальною системою регуляції дорожнього руху у великих містах. Її можна розділити на два етапи. Перший етап детально викладений у роботі [1]. На цьому етапі здійснюється регулювання трафіку через одне перехрестя, що взаємодіє з сусіднім та завдяки

п’єзокристалічним датчикам регулює проїзд ТЗ через перехрестя, враховуючи завантаженість трафіку.

Реєстрація ТЗ, перетинаючих перехрестя, організована наступним чином (рис.1).

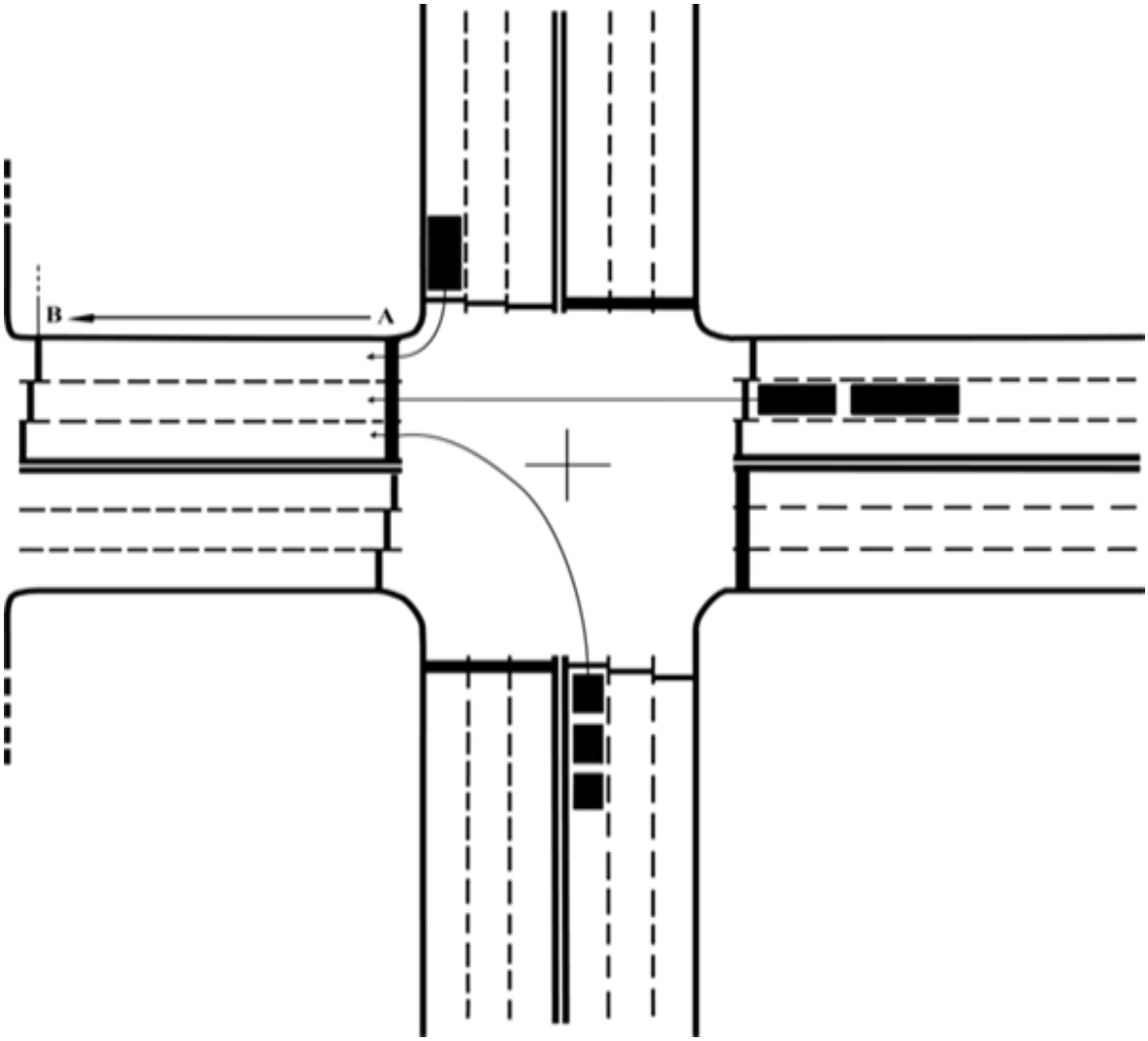

Рис.1. Умовне перехрестя А, з’єднане з сусіднім перехрестям В (показана лише його частина). Чорними прямокутниками зображені ТЗ, маршрути яких пролягають в напрямку А → В. Показані також вхідні (чорна смужка, також позначена літерою А) та вихідні датчики (три рядом розташовані смужки, позначені як В). Кожен вхідний та вихідний датчики з’єднані з ЦПКТ (на рисунку зображений лише один такий сигнальний провід, що виходить від одного із датчиків В – це суцільна лінія, що переходить в пунктирну).

Розглянемо технічну сторону реєстрації транспорту, проїжджаючого через перехрестя. П’єзокристалічні датчики монтуються в полотно проїзної

частини перпендикулярно до поздовжньої вісі зразу за стоп-лінією [24]. Пристрої реагують на тиск, спричинений автомобільною колісною парою. Вимірювально-обчислювальний комплекс (ВОК), що розташований в ЦПКТ, приймає електричний сигнал, спричинений стиском п'єзодатчика. Таким чином, відбувається реєстрація сигналів, що поступають з різних перехресть міста. Спектр сигналів, отримуваних з кожного перехрестя, пропорційний числу автомобілів, які перетнули його в різних напрямках. Проаналізуємо перший етам регулювання трафіку. Нехай маємо справу з хрестоподібним перехрестям, до якого застосуємо «розумну» технологію регулювання проїзду транспорту. Основний принцип світлофорного регулювання полягає в тому, щоб розподілити час світлофорного циклу, іншими словами період перемикання світлофору, пропорційно до завантаженості ТЗ напрямків на перехресті. Можуть бути окремі випадки коли необхідно пролонгувати час горіння зеленого світла в найбільш завантаженому напрямку. Схематично зрозуміти принцип дії «розумного» світлофора зрозумілий із рис. 1. Комп'ютерна програма створена для того щоб керувати роботою світлофорів на перехресті з таким розрахунком, щоб функція інтенсивності проїзду ТЗ через перехрестя $I = I(t)$ досягала максимуму (t – час). Взагалі період перемикання світлофору – це величина, що може складатися із таких складових.

$$T = t_H^G + t_H^R + t^Y + t_V^G + t_V^R + t^P, \qquad (1)$$

де t_H^G – час горіння зеленого світла у горизонтальному напрямку;

t_H^R – час горіння червоного світла у горизонтальному напрямку;

t^Y – час горіння жовтого світла;

t_V^G – час горіння зеленого світла у вертикальному напрямку;

t_V^R – час горіння червоного світла у вертикальному напрямку;

t^P – час горіння зеленого світла для пішоходів.

Тепер наведемо саму програму, що регулює проїзд ТЗ через хрестоподібне перехрестя.

```
package IC;
import java.util.Random;
import static java.lang.StrictMath.abs;

interface Lights{
    int REDH =0;
    int YELLOWH =1;
    int GREENH =2;
    int REDV =3;
    int YELLOWV =4;
    int GREENV =5;
    int GREENP = 6;
    int ERROR = -1;
  }
  class T implements Lights {
    private int delay;
    private static int light = REDH;
    T(int sec) {
      delay = 1000 * sec;
    }
    public int shift() {
      int count = (light++) % 7;
      try {
        switch (count) {
          case REDH:
            Thread.sleep(delay);
            break;
          case YELLOWH:
            Thread.sleep(delay / 3);
            break;
          case GREENH:
            Thread.sleep(delay / 2);
            break;
          case REDV:
            Thread.sleep(delay);
            break;
          case YELLOWV:
            Thread.sleep(delay / 3);
            break;
          case GREENV:
            Thread.sleep(delay / 2);
            break;
          case GREENP:
            Thread.sleep(delay );
            break;
        }
      } catch (Exception e) {
        return ERROR;
      }
      return count;
    }
  }
  class TrafficRegulator {
    static int T = 96;
    private static IC.T t = new IC.T(1);
    static Random gn1 = new Random();
    static Random gn2 = new Random();
    static Random gn3 = new Random();
    static Random gn4 = new Random();
      public static void main(String[] args) {
      double k =abs ((gn1.nextDouble()  +
gn2.nextDouble() ) /
            (gn3.nextDouble() + gn4.nextDouble()));
       double tg = 35;
      double tp = 23;
      double tyh = 2;
        double tgh = k * tg;
        double trh = (T - tyh - tgh- tp);
        double tgv = abs(2*tg -tgh);
        double trv = (T - trh);
        double tgp = 23;
        int tyv = 2;
        for (int j = 0; j < 7; j++)
          switch (t.shift()) {
            case Lights.REDH:
              System.out.println("red
horizontal!");
              System.out.format("%.1f%n",trh);
              break;
            case Lights.YELLOWH:
              System.out.println("yellow
horizontal!");
              System.out.println(tyh);
              break;
            case Lights.GREENH:
              System.out.println("green
horizontal!");
              System.out.format("%.1f%n",tgh);
              break;
            case Lights.REDV:
              System.out.println("red vertical!");
              System.out.format("%.1f%n",trv);
              break;
            case Lights.YELLOWV:
              System.out.println("yellow
vertical!");
              System.out.println(tyv);
              break;
            case Lights.GREENV:
              System.out.println("green
vertical!");
              System.out.format("%.1f%n",tgv);
              break;
            case Lights.GREENP:
              System.out.println("green
pedestrian!");
              System.out.println(tgp);
              break;
            case Lights.ERROR:
              System.out.println("Time error!");
              break;
            default:
              System.err.println("Unknown
light.");
              return;
          }
```

```
}                                                    }
```

В результаті роботи програми отримуємо:

```
red horizontal!
29,6
yellow horizontal!
2.0
green horizontal!
41,4
red vertical!
66,4
yellow vertical!
2
green vertical!
28,6
green pedestrian!
23.0
Process finished with exit code 0.
```

Розглянемо аналіз роботи цієї програми. Прикладом констант в програмі є величини такого типу REDH = 0 – червоне світло в горизонтальному напрямку. Принципово те, що програма аналізує завантаженість ТЗ напрямків перехрестя: чим більше завантажений напрямок – горизонтальний чи вертикальний, – то чим більше горітиме зелене світло для відповідного напрямку (звичайно в межах періоду перемикання світлофора). Щоб імітувати ситуації із реальними процесами, в програмі було включено чотири генератори випадкових величин, що імітують завантаженість перехрестя ТЗ. Нас цікавить відношення числа автомобілів, що розташовані на горизонтальному та вертикальному напрямках. В програмі таке відношення задається величиною коефіцієнта k. Варіація величини k імітує якраз зміни навантаженостей ТЗ. Тому спектр чисел на виході програми щораз змінюється випадковим чином. Змінюються, зокрема, величини, що задають час горіння зеленого світла у горизонтальному та вертикальному напрямках.

Виходячи з цього, презентована програма дає можливість суттєво покращити пропускну здатність перехрестя завдяки «розумному» режиму їх роботи в плані протяжності горіння різних фаз та їх пролонгації – останнє по

необхідності. Причому пролонгація може стосуватись не тільки часу горіння зеленого світла, а також і часу горіння червоного світла. Останнє можливе, наприклад, коли певний напрямок на перехресті основної та другорядної доріг не завантажений зовсім – звичайно зі сторони другорядного напрямку, – тоді на цей напрямок має постійно горіти червоне світло; зелене включається лише у випадку появи на вказаному напрямку хоча б одного ТЗ.

Зауважимо, що кожне окреме перехрестя працює у тісній взаємодії із сусіднім перехрестям. Щоб показати таку взаємодію на рис.1 зображена ділянка дороги між сусідніми перехрестями. Вказана взаємодія здійснюється з допомогою вхідних та вихідних датчиків. Вхідний датчик зображений на рис. 1 у вигляді суцільної чорної смужки та позначений літерою А, вихідний – літерою В. Датчик А реєструє автомобілі, які в'їжджають на ділянку дороги АВ з трьох можливих напрямків перехрестя А.

На другому ж етапі здійснюється регулювання дорожнього руху через все місто. Розглянемо цей етап регулювання трафіку, що і є основною метою дослідження. Оскільки всі перехрестя міста знаходяться під контролем ЦПКТ, то в цьому разі є можливість використання інтелектуальної технології регулювання проїзду ТЗ по заявленому маршруту. Стартову, а особливо фінішну позиції кожен водій, що приймає участь у трафіку, задає центральному пункту керування. В мегаполісах число автомобілів може сягати понад мільйон, а тому важливо знайти оптимальний маршрут руху для кожного ТЗ на основі використання отриманих даних з кожного перехрестя вважаючи, що вони змінюються дуже швидко. Це означає, що потрібно використовувати динамічну базу даних, яка спроможна оновлюватись, наприклад, кожні 5 секунд. Це означатиме, що зв'язки «ЦПКТ↔ Перехрестя» працюють в режимі реального часу, постійно оновлюючи дані про завантаженість ділянок дороги між перехрестями.

Таким чином, завдання полягає в тому, щоб побудувати оптимальний маршрут руху для кожного автомобіля, враховуючи ситуацію на транспортній мережі міста у кожен конкретний момент часу. ЦПКТ, працюючий на основі

пропонованої далі комп'ютерної програми, отримує пакети даних, що поступають з кожного перехрестя. У відповідь на вхідні дані ЦПКТ видає керуючі сигнали на світлофори кожного перехрестя. Тож всі перехрестя перебувають під контролем і ЦПКТ постійно володіє ситуацією щодо завантаження перехресть та дорожніх шляхів, що їх пов'язують. Таке керування дозволяє покращити проїзд через перехрестя та дає можливість прокласти маршрут кожному *i*-му ТЗ ($i = 1 \div N$, де N число автомобілів, що замовили маршрут ЦПКТ, тобто заявили свою пару (S_i, F_i). Водій – при необхідності – обирає також мову супроводу. Наприклад, водій-українець у Нью-Йорку може запросити в якості звукового гіда свою рідну мову.

Звернемо увагу, що з технічної точки зору ситуація виглядає так: програма на ЦПКТ працює з кожним ТЗ та розраховує для нього оптимальний маршрут руху і передає дані водієві на GPS-навігатор чи мобільний телефон, обладнаний спеціальним додатком. Розраховуючи оптимальний маршрут на даний момент часу, ЦПКТ веде водія по ньому, передаючи постійно інформацію щодо дій водія стосовно поворотів, розворотів, перелаштувань в іншу смугу і тому подібне. Дорожня ситуація у місті змінюється кожної миті і тому по мірі руху кожного ТЗ програма знаходиться у постійному пошуку оптимального маршруту та при появі труднощів на встановленому маршруті вона змінює маршрут руху водія на новий, але знову ж таки оптимальний. Так буде продовжуватись поки водій автомобіля *i* не прибуде до кінцевого пункту маршруту F_i.

Проїзд автомобіля по місту, в якому кожен світлофор оснащений описаною в [1] системою регулювання трафіком, здійснюється з допомогою алгоритму, який із всіх можливих варіантів маршруту між пунктами S_i і F_i обирає оптимальний. Тут оптимальність визначається з допомогою виконання наступної умови:

$$\sum_{h=1}^{f} (N_{A_h B_h} / n_{A_h B_h}) l_{A_h B_h} \to \min, \qquad (2)$$

де $N_{A_hB_h}$ – число ТЗ, що в'їжджають на ділянку дороги A_hB_h одного напрямку; $n_{A_hB_h}$ – число ТЗ, що виїжджають із ділянки дороги A_hB_h одного напрямку; h – індекс, що нумерує проїзну частину дороги одного напрямку руху вздовж маршруту; іншими словами, ребро графа між інцидентними (сусідніми) вузлами (перехрестями), що зчеплені між собою та утворюють простий ланцюг, з'єднуючи початкову S_i та кінцеву F_i координати, заявлені водієм автомобіля i. Символ f означає число смуг виду A_hB_h, які формують прокладений маршрут із мультиплікат типу (1). Співставимо фрагмент транспортної мережі міста (рис.2) з орієнтованим навантаженим графом (рис.3). Процедура прокладання маршруту здійснюється з допомогою програми, яка може бути двох варіантів. У першому варіанті ця програма буде використовувати алгоритм Флойда-Воршела [25-26], а у другому - алгоритм Дейкстри [27]. Для складання такої програми в цій роботі використовується мова програмування Python. Програма в обох її варіантах використовує в якості даних, які вводять в консолі, набір всіх значень виду (1), величини яких (для деякого моменту часу!) представлені на рис.3 біля ребер графа.

Дані алгоритми можуть знайти оптимальний шлях, але між ними є певна різниця.

В інформатиці алгоритм Флойда-Воршелла використовується для розв'язання задачі про найкоротший шлях у зваженому графі з додатними або від'ємними вагами ребер. При звичайній реалізації алгоритм видасть довжини (сумарні ваги) найкоротших шляхів між всіма парами вершин, хоча він не видасть інформацію про самі шляхи [26]. В той час коли алгоритм Дейкстри знаходить найкоротший шлях від однієї вершини графа до всіх інших вершин та працює тільки для графів без циклів від'ємної довжини [27].

Алгоритм Дейкстри, на відміну від алгоритму Флойда-Воршела з допомогою якого можна відшукати найкоротший маршрут між будь-якими двома вершинами графа, призначений для пошуку маршруту найкоротшої

довжини від заданої вершини графа до всіх інших [27]. Тому найбільш підходить для дослідження саме алгоритм Дейкстри [28-29]. Його ж і будемо розглядати більш детально. Програму із реалізацією обох алгоритмів можна буде побачити нижче.

Треба здійснити імітаційне моделювання перед технічним впровадженням пропонованого проекту, тобто провести імітаційну апробацію даної технології. Програма візуального імітаційного моделювання AnyLogic al 7.3. [30] є дуже перспективним методом апробації такого типу розробок: в ній представлено ряд імітаційних моделей, що регулюють дорожній трафік.

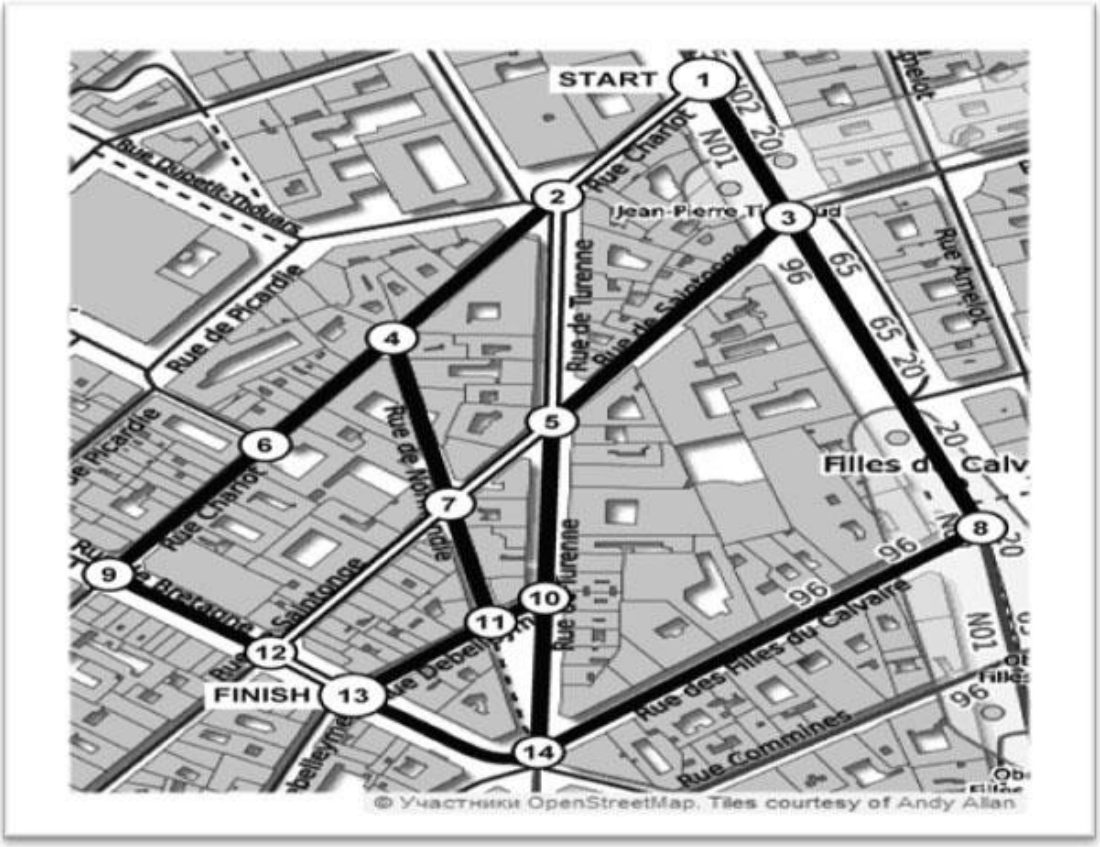

Рис. 2 Транспортна мережа одного із кварталів центрального району Парижа. Тут зроблені позначення перехресть (цифри від 1 до 14), що відповідають графу,представленому на рис.3. Перехрестя 1 відмічене надписом START, а перехрестя 13 визначено як кінцевий пункт поїздки – (FINISH).

AnyLogic 7.3 є ефективним інструментом для розробки об'ємних імітаційних моделей, складних анімацій, вбудовування моделей в різні IT оточення, створення (і подальшого використання) власних бібліотек для різних сфер застосування [30]. Найбільш ілюстративною та найбільш близькою до даної

теми є імітаційна модель Traffic Light Phases Optimization, яка візуально імітує проїзд ТЗ через три сусідні перехрестя. За допомогою цієї програми можна максимізувати інтенсивність проїзду ТЗ через кожне із зазначених вище перехресть за рахунок зміни вручну за допомогою слайдерів протяжності фаз горіння зеленого світла у взаємно перпендикулярних напрямках проїзних частин дороги. Тут важливо здійснювати таке регулювання не вручну, а на основі створеного алгоритму.

Розглянемо граф, представлений на рис.3. Він являє собою своєрідне відтворення транспортної мережі міста: ребра графа – це дороги між перехрестями, а вершини – це власне є перехрестя .

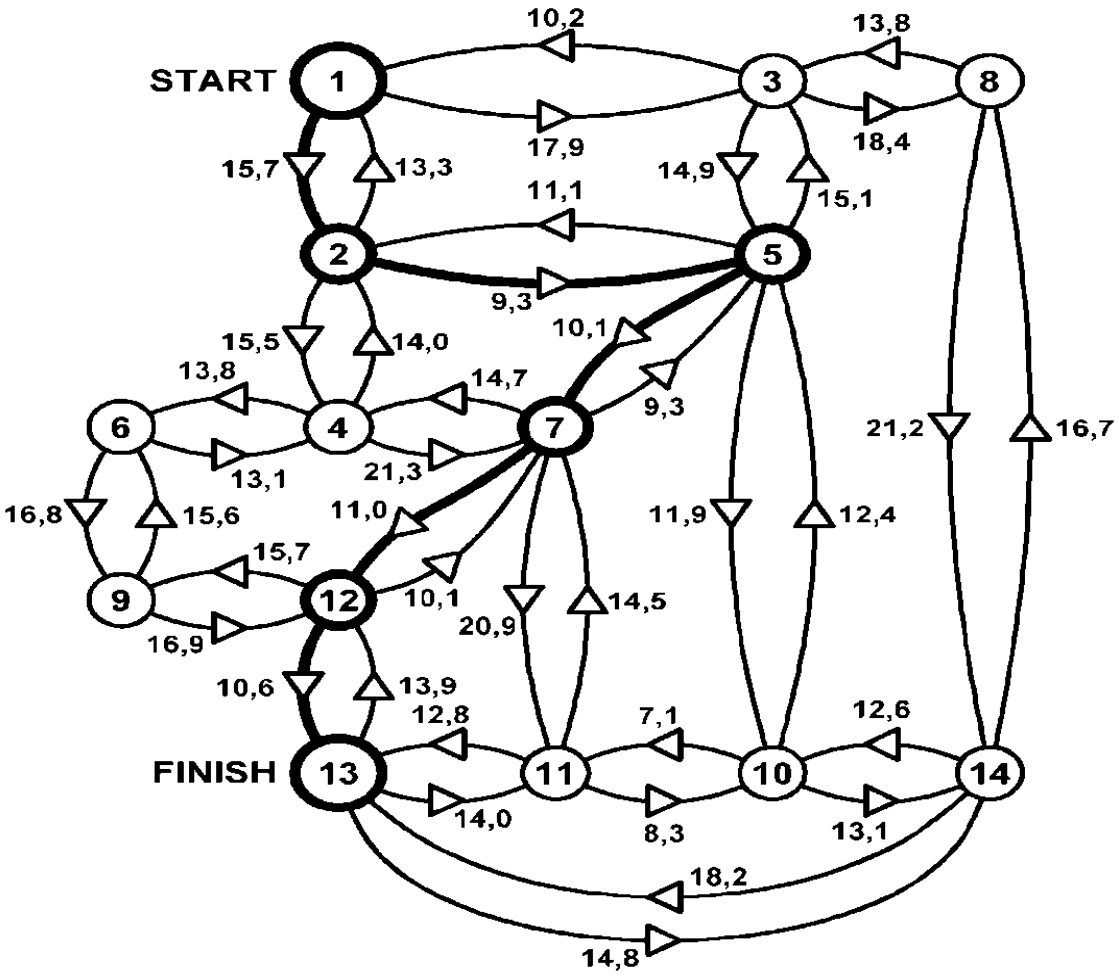

Рис.3.Зв’язний навантажений граф (орієнтований). Напрямки ребер задаються трикутниками, біля яких приведені ваги ребер. Написи «START» та «FINISH» відповідають вершинам графа відповідно 1 та 13 і символізують

собою початок та кінець подорожі конкретного ТЗ. Один із можливих маршрутів між пунктами 1 і 13 для певного моменту часу зображений жирною лінією. В реальності зображений жирною лінією маршрут може бути змінений в залежності від завантаженості ребер.

В даному випадку маємо справу з двореберним графом. Прокладання шляху руху по місту кожного ТЗ здійснюється за допомогою спрощеної програми, представленої знизу. Ця програма прокладає маршрут, не враховуючи тієї обставини, що «ділянка дороги одного напряму між сусідніми перехрестями» власне складається, як правило, з кількох смуг руху. Проте, не дивлячись на сказане спрощення, представлений тут алгоритм цілком придатний та спроможний здійснювати супровід кожного автомобіля – водієві лише потрібно буде самостійно здійснювати перестроювання на суміжні смуги руху, знаючи із GPS-карти ділянку дороги на інцидентному (сусідньому) плечі прокладеного маршруту. Це не створюватиме особливих перешкод.

Для порівняння нижче приведені фрагменти реалізації програми із алгоритмами Флойда-Воршела та алгоритмом Дейкстри.

3.1. Алгоритм Флойда-Воршела

```
#Function used as a decorator to provide a class like Singleton
def singleton(cls):
        instances = {}
        def getinstance():
                if cls not in instances:
                        instances[cls] = cls()
                return instances[cls]
        return getinstance
#The Singleton class that works with the algorithm Floyd Warshal
@singleton
class FloydWarshal:
        """
        Initializes the base matrix in the appropriate sizes
        numberofvertices - the number of vertices
        (the matrix will be created as numberofvertices x numberofvertices)
```

```
        """
        def __init__(self):
            self.numberofvertices = 0
            self.adjacencymatrix = None
            self.distancematrix = []
        #The method of the class that fills the matrix
        def setData(self, adjacencymatrix=[]):
            self.adjacencymatrix = adjacencymatrix
        def setNumberOfVertices(self, numberofvertices):
            self.numberofvertices = int(numberofvertices)
        #A class method that executes Floyd Worshell algorithm
        def algorithm(self):
            """Nulled i=j"""
            if len(self.adjacencymatrix) == self.numberofvertices:
                for i in range(len(self.adjacencymatrix)):
                    for j in range(len(self.adjacencymatrix[i])):
                        if i == j:
                            self.adjacencymatrix[i][j] = 0
            """Load matrix in to new matrix"""
            self.distancematrix = self.adjacencymatrix
            if len(self.adjacencymatrix) == self.numberofvertices:
                #Algorithm Floyd Warshal
                for k in range(self.numberofvertices):
                    for i in range(len(self.distancematrix)):
                        for j in range(len(self.distancematrix[i])):
                            if self.distancematrix[i][k] +
self.distancematrix[k][j] < self.distancematrix[i][j]:
                                self.distancematrix[i][j]=
self.distancematrix[i][k] + self.distancematrix[k][j]
```

3.2. Алгоритм Дейкстри

```
def Dijkstra(N, S, matrix):
    valid = [True]*N
    weight = [1000000]*N
    weight[S] = 0
    pred = {}
    for i in range(N):
        pred[i + 1] = []
    for i in range(N):
        min_weight = 1000001
        ID_min_weight = -1
        for j in range(len(weight)):
            if valid[j] and weight[j] < min_weight:
                min_weight = weight[j]
                ID_min_weight = j
        for j in range(N):
```

```
                if weight[ID_min_weight] + matrix[ID_min_weight][j] < weight[j]:
                    weight[j] = weight[ID_min_weight] + matrix[ID_min_weight][j]
                    pred[j + 1].append(ID_min_weight + 1)
            valid[ID_min_weight] = False
    return [weight, pred]

def main():
    while True:
        try:
            number = int(input("Введите количество вершин: "))
        except ValueError:
            print("Ошибка ввода попробуйте еще раз")
        else:
            break
    """Ввод Матрицы"""
    matrix = []
    print("Ввод матрицы")
    for i in range(number):
        matrix.append([])
        for j in range(number):
            while True:
                try:
                    matrix[i].append(float(input("MATRIX[%d][%d]: " % (i + 1, j + 1))))
                except ValueError:
                    print("Ошибка ввода попробуйте еще раз")
                else:
                    break
    print("Ваш масив:")
    print('   ', end='')
    for i in range(number):
        print('%d  ' % (i + 1), end='')
    print()
    for i in range(number):
        print('%d  ' % (i + 1), end='')
        for j in range(number):
            print('%.2f  ' % (matrix[i][j]), end='')
        print()

    while True:
        try:
            vershina = int(input("З якої вершиини розпочати обхід: "))
        except ValueError:
            print("Ошибка ввода попробуйте еще раз")
        else:
            if vershina <= 0 or vershina > number:
                print("вершина має бути від 1 до %i" %(number))
            else:
                break
    print("Результат роботи алгоритму Дейкстри:")
    print(Dijkstra(number, vershina - 1, matrix))

if __name__ == '__main__':
    main()
```

Програма, що керує трафіком, повинна вибрати оптимальний маршрут для кожного заданого маршруту, тобто для кожної пари (S_i, F_i) підібрати такий комплект із (2), який є простим ланцюгом.

Застосовуємо приведену вище програму до графа, який зображено на рис.3, вважаючи, що шуканий маршрут пролягає від точки 1 (START) до точки 13 (FINISH). Спектр даних приведений нижче означає наступне: ***1*** – номер вершини; ***14*** – число вершин графа; ***40*** – число ребер графа, від якої стартує маршрут. Далі приведено 40 тріад чисел, перше означає номер вихідної вершини, друге – номер вершини, до якої прокладається маршрут, третє – це дійсне число, що являє собою навантаженість ребра графа. Дані ми ці вводимо в консолі. Наголос на тому, що приведений спектр дійсних чисел (третя колонка) змінюється кожні 10 секунд. Дані числа являють собою

14 40 1
1 2 15.7
2 1 13.3
1 3 17.9
3 1 10.2
3 8 18.4
8 3 13.8
3 5 14.9
5 3 15.1
2 5 9.3
5 2 11.1
2 4 15.5
4 2 14.0
5 7 10.1
13 14 14.8
7 5 9.3
7 4 14.7
4 7 21.3
4 6 13.8
6 4 13.1
6 9 16.8
9 6 15.6
9 12 16.9
12 9 15.7
12 7 10.1
7 12 11.0
12 13 10.6
13 12 13.9
7 11 20.9
11 7 14.5
10 5 12.4
5 10 11.9
8 14 21.2
14 8 16.7
13 11 14.0
11 13 12.8
11 10 8.3
10 11 7.1
10 14 13.1
14 10 12.6
14 13 18.2

Результат роботи програми – спектр чисел (виділених жирним шрифтом), ці цифри представляють собою відстань від вершини 1 до інших 14 вершин. Після цього ідуть 14 рядків, що являють собою оптимальні маршрути від вершини 1 до інших вершин графа. Но нас цікавить маршрут виду «1→13».

Саме цей маршрут буде передавати водієві (на певний момент часу!) GPS-навігатор (при умові, що дорожня ситуація суттєво не зміниться).

0.0 15.7 17.9 31.2 25.0 45.0 35.1 36.3 61.8 36.9 44.0 46.1 *56.7* 50.0

1: 1

2: 1 2

3: 1→ 3

4: 1→ 2 →4

5: 1 →2→5

6: 1→ 2→ 4→ 6

7: 1→ 2→ 5→7

8: 1→ 3 →8

9: 1→ 2→ 4→ 6 →9

10: 1 → 2→ 5 → 10

11: 1→ 2→ 5→ 10 → 11

12: 1→ 2→ 5→ 7→ 12

13: ***1→ 2→ 5 → 7 → 12→ 13***

14: 1→ 2→ 5→ 10→ 14

Отримавши приведену вище послідовність чисел ***1→ 2→ 5 →7 →12→ 13***, програма в режимі GPS-навігації передає дані кожному водієві, причому ці дані, як і кожне ребро графа на рис.3, завдяки автоматизації кожного перехрестя в місті моніторяться кожні 10 секунд; інакше кажучи це дає зрозуміти, що ваги ребер постійно оновлюються у відповідності із даними, які отримуються із кожного перехрестя. В цьому випадку наша програма має перевагу над дорожніми картами Google, які працюють із запізненням, і тому констатують дорожню ситуацію постфактум, цим самим знецінює інформацію такого роду, оскільки це не дає можливості водієві своєчасно реагувати на зміну дорожньої обстановки по його дорожній карті. Якщо впроваджувати пропоновану систему, то це дозволить здійснити синхронізацію руху великої кількості автомобілів (наприклад, у місті Пекін на вулицях щоденно курсує понад 1 млн. ТЗ). Кожен водій який буде під'єднаний до ЦПКТ, буде отримувати вказівки щодо руху по маршруту. Більше водію немає про що турбуватись, тому що програма ЦПКТ буде передавати голосові команди, як це робить звичайний GPS-навігатор. Якщо на дорозі по якій їде водій буде зміна дорожньої обстановки (заблоковане перехрестя, ДТП по курсу і т.д.) то програма миттєво передає водієві новий розрахований, але оптимальний маршрут.

Розділ 4. Аналіз результатів та висновки дослідження

Під час проведення дослідження було вивчено багато наукових статей, враховуючи статті науковців різних країн. Робота за своїм характером є пошуковою та принесла багато цікавих ідей, одна із який втілена у викладеному вище матеріалі.

Була розроблена технологія побудови оптимальних маршрутів для транспортних засобів у великих містах, враховуючи взаємодію саме із кожним ТЗ, хоча їх кількість може сягати мільйонів.

Система регулювання трафіку у місті працює динамічно, у реальному часі, враховуючи зміну дорожньої ситуації кожних 10 секунд.

Програма, якою керується центральний пункт керування трафіком (ЦПКТ), реалізована на мові програмування Java і Python та співпрацює із водіями транспортних засобів через GPS-навігатор чи спеціальний додаток для мобільного телефону.

Технологія допоможе позбутись проблеми кожного великого міста – заторів та зробить для водіїв більш комфортне слідування по заявленому маршруту, завдяки прокладенню саме оптимального маршруту для кожного ТЗ та супроводу шляху голосовими командами із вказівками щодо руху по маршруту.

Дана технологія, при втіленні її у життя, значно покращить ситуації на вулицях міст та внесе неабиякий внесок у рух транспорту в усьому світі.

Перелік використаних джерел

1. Богуто Д.Г., Волинець В.І., Ніколюк П.К., Ніколюк П.П. Автоматизована система керування рухом транспортних засобів в межах міста / Д.Г. Богуто, В.І. Волинець, П.К. Ніколюк, П.П. Ніколюк // Вісник Харківського університету, серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління». – 2017.– №5. – С. 3-9.
2. A to Z List of Road Traffic Contractors - Road Traffic Technology [Електронний ресурс]. – Режим доступу: www.roadtraffic-technology.com/contractors/.
3. Как избавиться от пробок на дороге? - 1Gai [Електронний ресурс]. – Режим доступу: http://www.1gai.ru/publ/512991-kak-izbavitsya-ot-probok-na-doroge.
4. Pat.US20150153176 Korea, Vehicle, vehicle cloud system, and data dissemination system / Hyuk Lim, In Shick Kim, Ryangsoo Kim; applicant Gwangju Institute Of Science and Technology; publ. 04.06.2015.
5. Smart Traffic Management- Smarter Cambridge Transport [Електронний ресурс]. – Режим доступу: www.smartertransport.uk.
6. Rinaldi Marco, Viti Francesco. Exact and approximate route set generation for resilient partial observability in sensor location problems/Marco Rinaldi, Francesco Viti//Transportation Research Part B: Methodological. – 2017. –V.105. – № 11. – P. 86 – 119.
7. Gaurav Pandey, K. Ramachandra Rao, Dinesh Mohan. Modelling vehicular interactions for heterogeneous traffic flown using cellular automata with position preference/ Gaurav Pandey, K. Ramachandra Rao, Dinesh Mohan // J. Mod. Transport. – 2017. – V. 25. – №3. – P.163–177.
8. Memon A. A., Meng M., Wong Y. D., Lam S. H. Calibration of a rule-based intelligent network simulation model/ A. A.Memon, M. Meng, Y. D Wong., S. H. Lam // J. Mod. Transport. – 2016. – V. 24. – №1. – P.48–61.
9. Wu Wei, Liu Yang, Xu Yue, Wei Quanlun, Zhang Yi. Traffic Control Models Based on Cellular Automata for At-Grade Intersections in Autonomous Vehicle

Environment/ Wei Wu, Yang Liu, Yue Xu, Quanlun Wei, Yi Zhang // Journal of Sensors. –Volume 2017. – Article ID 9436054.– 6 pages.

10. Pat. CN104575066 China, Intelligent terminal based intelligent traffic light system and method/ Xu Chunmao Xu Jin; applicant Shanghai Bolter Digital Technology Co., LTD.; publ.29.04.2015.
11. Case studies for traffic solutions - Siemens [Електронний ресурс]. – Режим доступу: www.mobility.siemens.com/ solutions / case-studies-for-tr.
12. David K. Chang, Mitsuru Saito, Grant G. Schultz, Dennis L. Eggett. Use of Hi-resolution data for evaluating accuracy of traffic volume counts collected by microwave sensors/ K. David, Chang, Saito Mitsuru, Schultz G.Grant, Eggett L. Dennis // Journal of traffic and transportation engineering. – 2017. – V.4. – Is. 5. – P. 423-435.
13. Dazhi Sun,Jinpeng Lv,S. Travis Waller. In-depth analysis of traffic congestion using computational fluid dynamics (CFD) modeling meth/ Sun Dazhi, Lv Jinpeng, Waller S. Travis // Journal of Modern Transportation. – 2011.–V.1. – № 1. – P. 58-67.
13. Nair T.R. Gopalakrishnan, Sooda Kavitha. Comparison of Genetic Algorithm and Simulated Annealing Technique for Optimal Path Selection in Network Routing/ Gopalakrishnan Nair T.R., Kavitha Sooda // 2010. –arXiv: 1001.3920. Available at: http://arxiv.org/abs/1001.3920.
14. Ampountolas Konstantinos, Zheng Nan, Geroliminis Nikolas. Macroscopic modelling and robust control of bi-modal multi-region urban road networks/ Konstantinos Ampountolas, Nan Zheng, Nikolas Geroliminis // Transportation Research Part B: Methodological. – 2017. –V.104. – P.616–637.
15. Wikibooks.org, Алгоритм Дейкстри (реализация на Python) [Електронний ресурс]. Режим доступу: https://ru.wikibooks.org/wiki/Реализации_алгоритмов/Алгоритм_Дейкстры.
16. AnyLogic [Електронний ресурс]. – Режим доступу:https://www.anylogic.ru.
17. П’єзокристалічний датчик [Електронний ресурс]. Режим доступу: https://en. wikipedia.org/wiki/Piezoelectric_sensor.

18. Fully Adaptive Traffic Management Systems [Електронний ресурс]. – Режим доступу: http://isbak.istanbul/intelligent-transportation-systems/full-adaptive-traffic-management-system/.
19. Traffic density map [Електронний ресурс]. Режим доступу: http://isbak.istanbul/intelligent-transportation-systems/traffic-control-center/traffic-density-map/.
20. Traffic signalization systems [Електронний ресурс]. Режим доступу: http://isbak.istanbul/intelligent-transportation-systems/traffic-signalization-systems/.
21. A survey on urban traffic management system using WSN [Електронний ресурс]. – Режим доступу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4801535/.
23. Intelligent traffic light control [Електронний ресурс]. Режим доступу: https://www.ercim.eu/publication/Ercim_News/enw53/wiering.html
24. Виглеб Г. Датчики / Виглеб Герхард. -М.: Мир, 2014. – 236 с.
25. Алгоритм Флойда-Воршела (реалізація на Python) [Електронний ресурс].– Режим доступу: https://foxford.ru/wiki/informatika/algoritm-floyda.
26. Алгоритм Флойда-Воршела [Електронний ресурс].– Режим доступу: https://uk.wikipedia.org/wiki/ Алгоритм_Флойда_—_Воршелла.
27. Алгоритм Дейкстри [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/ Алгоритм_Дейкстри.
28. Алгоритм Дейкстри [Електронний ресурс]. Режим доступу: http://www.mathros.net.ua/znahodzhennja-najkorotshogo-shljahu-v-orijentovanomu-grafi-za-algorytmom-dejkstry.html.
29. Алгоритм Дейкстри [Електронний ресурс].– Режим доступу: https://prog-cpp.ru/deikstra/.
30. Anylogic Proffesional [Електронний ресурс]. Режим доступу: https://help.anylogic.ru/index.jsp?topic=%2Fcom.anylogic.help%2Fhtml%2Fwhatsnew%2FAnyLogic_Professional.html.