
Moscow Institute of Physics and Technology

Phystech School of Applied Mathematics and Informatics

Major of study: Applied Mathematics and Informatics (bachelor program)

Specialization: Computer Science/Информатика

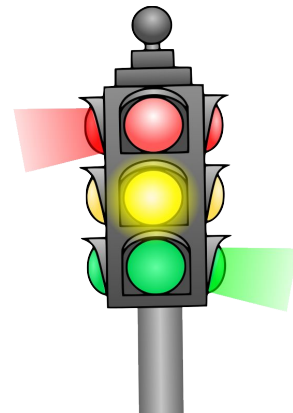
Graph Theory Application for Traffic Light Control at Crossroads

Student: Abo Alnaser Mohamad

Scientific supervisor: Daynyak Aleksandr Borisovich

Contents

- Formulation of the problem
- Describing the algorithm
- Mathematical model for setting the optimal total waiting time
- Software simulation
- Summary and results




Formulation of the problem

Traffic jams at traffic lights at crossroads

In order to solve such an issue:

1- Building new roads.

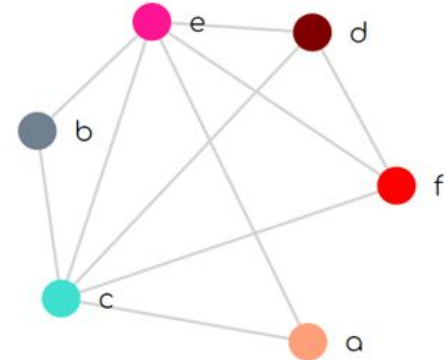
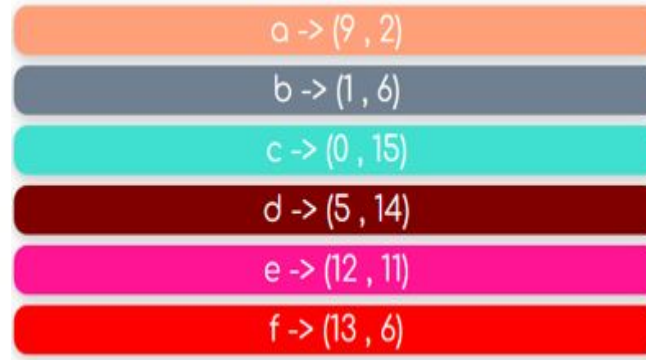
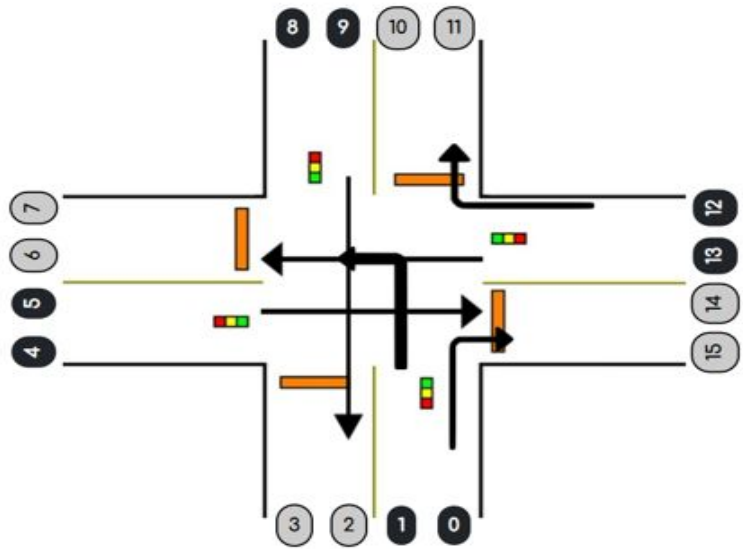
2- Model and install an efficient traffic management system on the existing roads.

 Graph theory can be applied on traffic lights at crossroads for collecting all the information about the traffic intersection compatibility into a graph called the **compatibility graph**.

Goals:

- Developing an algorithm for modeling the compatibility graph.
- Making a mathematical model for setting the optimal total waiting time.

Compatibility graph



Streams that can flow in the intersection simultaneously without causing crashes are called **compatible**.

Cliques:

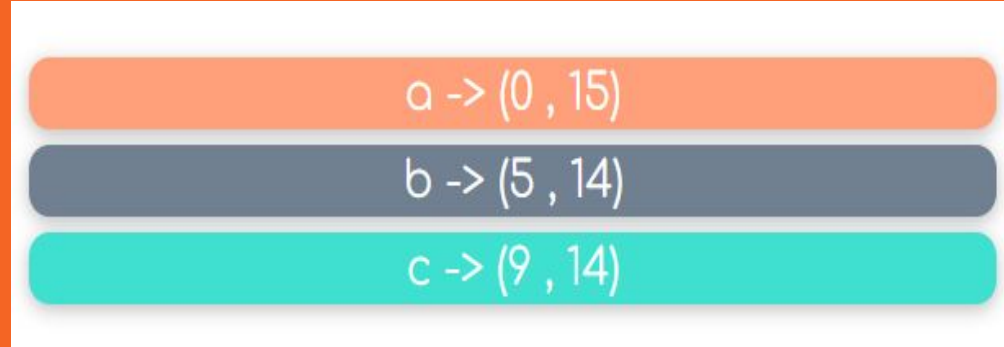
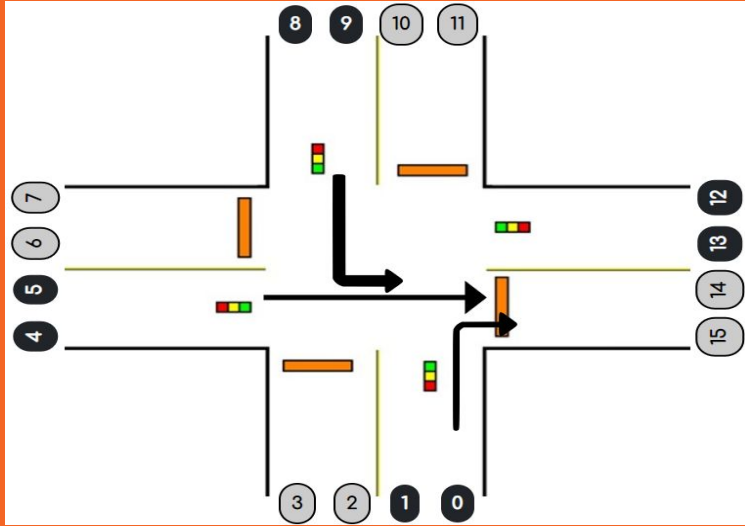
$$K_1 = \{c, d, e, f\}$$

$$K_2 = \{a, c, e\}$$

$$K_3 = \{b, c, e\}$$

Describing the algorithm

1. Pairing traffic flows with different endpoints:

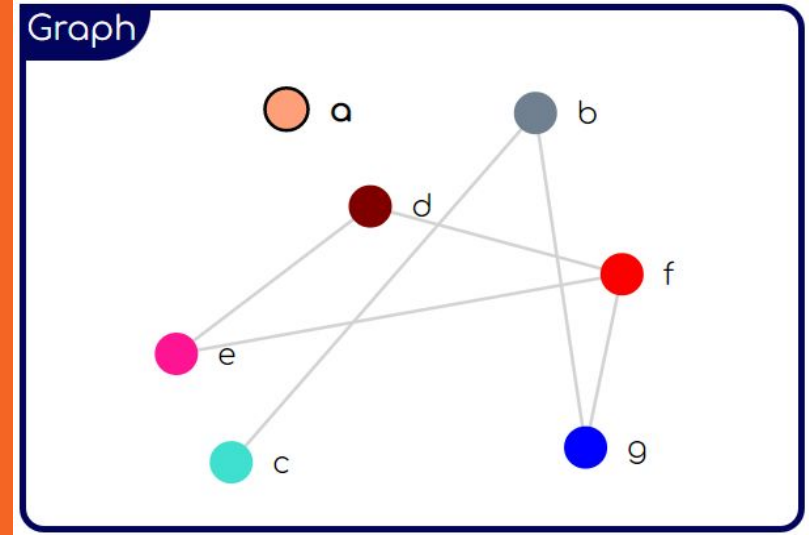
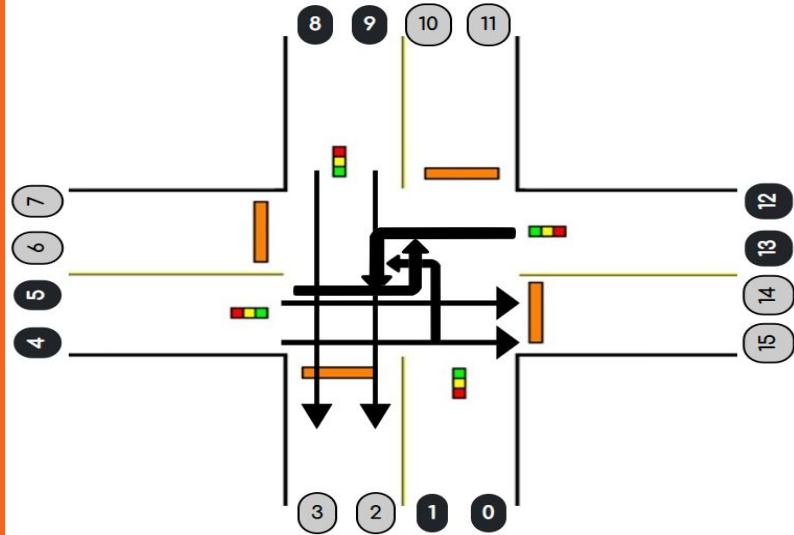


$[[[5, 14], [0, 15]], [[9, 14], [0, 15]]]$

The algorithm iterates over these values and pairs them into a single array if and only if they have different endpoints. Eventually, all of the generated arrays that contain paired traffic flows with different endpoints are added to a new single array u .

Describing the algorithm

2. Generating all possibilities of traffic intersections:



For instance, consider the two sets $A = \{[1, 6]\}$ and $B = \{[8,3], [9,2], [5,14], [4,15], [5,10], [13,2]\}$

$A \times B = \{ [[1,6], [8,3]], [[1,6], [9,2]], [[1,6], [5,14]], [[1,6], [4,15]], [[1,6], [5,10]], [[1,6], [13,2]] \}$

We keep generating all possible intersections and eventually we add all of them into a new single array v .

Describing the algorithm

3. Generating all compatible pairs of traffic flows:

a -> (9 , 2)

b -> (1 , 6)

c -> (0 , 15)

d -> (5 , 14)

e -> (12 , 11)

f -> (13 , 6)

At this point, the algorithm applies the **combinations without repetition** of our traffic flows to get all different groups of traffic flows.

```
z = [  
  [[9, 2], [1, 6]], [[9, 2], [0, 15]], [[9, 2], [5, 14]], [[9, 2], [12, 11]],  
  [[9, 2], [13, 6]], [[1, 6], [0, 15]], [[1, 6], [5, 14]], [[1, 6], [12, 11]],  
  [[1, 6], [13, 6]], [[0, 15], [5, 14]], [[0, 15], [12, 11]], [[0, 15], [13, 6]],  
  [[5, 14], [12, 11]], [[5, 14], [13, 6]], [[12, 11], [13, 6]]  
]
```

Describing the algorithm

4. Generating all maximal cliques and modeling the compatibility graph:

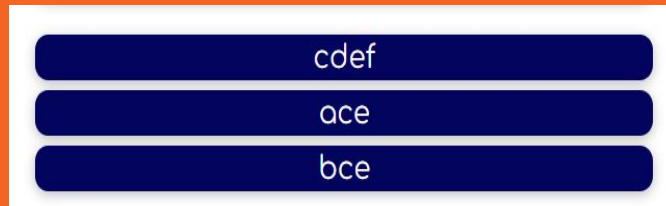
➤ $z = [$
[[9, 2], [0, 15]], [[9, 2], [12, 11]], [[1, 6], [0, 15]], [[1, 6], [12, 11]],
[[0, 15], [5, 14]], [[0, 15], [12, 11]], [[0, 15], [13, 6]],
[[5, 14], [12, 11]], [[5, 14], [13, 6]], [[12, 11], [13, 6]]
]

**All pairs of compatible
traffic flows**

➤ $z = [$
[[9, 2], [0, 15], [12, 11]], [[9, 2], [12, 11], [0, 15]],
[[1, 6], [0, 15], [12, 11]], [[1, 6], [12, 11]],
[[0, 15], [5, 14], [12, 11], [13, 6]], [[0, 15], [12, 11], [13, 6]],
[[0, 15], [13, 6], [12, 11]], [[5, 14], [12, 11], [13, 6]],
[[5, 14], [13, 6]], [[12, 11], [13, 6]]
]

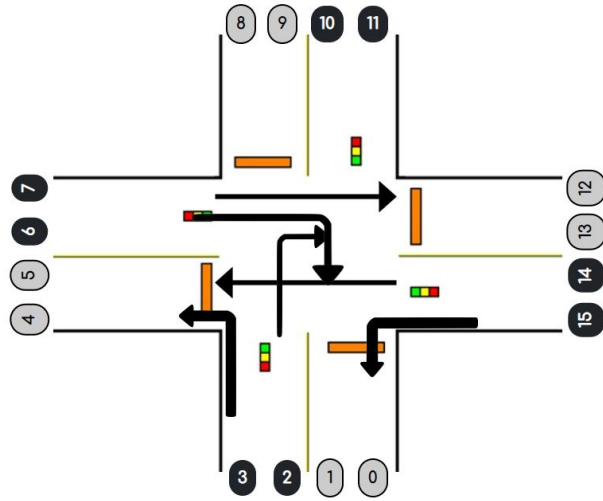
All possible cliques

➤ $z = [$
[[0, 15], [5, 14], [12, 11], [13, 6]],
[[9, 2], [0, 15], [12, 11]],
[[1, 6], [0, 15], [12, 11]]
]



**All possible maximal
cliques**

Mathematical model for setting the optimal total waiting time



From our compatibility graph we can say that it has 3 cliques:

$$K_1 = \{a, b, d, e\}$$

$$K_2 = \{a, c, d, e\}$$

$$K_3 = \{a, d, e, f\}$$

In clique 1, the traffic flows **a, b, d, e** has the green light to flow, then, in clique 2, traffic flows **a, c, d, e** has the green light and so on.

Let's assign a duration d_i to each clique k_i :

$$K_1 = \{a, b, d, e\} \rightarrow d_1$$

$$K_2 = \{a, c, d, e\} \rightarrow d_2$$

$$K_3 = \{a, d, e, f\} \rightarrow d_3$$

Mathematical model for setting the optimal total waiting time

b has the red light during the flow of these cliques **K2** and **K3**, so **b**'s total waiting time is **d2 + d3**.

Also, **a**'s, **c**'s, **d**'s, **e**'s and **f**'s total waiting time is 0, **d1 + d3**, 0, 0 and **d1 + d2** respectively.

So, we can have the mathematical model **Z** which is the total waiting time for all traffic flows:

$$Z = 0 + (d_2 + d_3) + (d_1 + d_3) + 0 + 0 + (d_1 + d_2) = 2d_1 + 2d_2 + 2d_3$$

Let's suppose that the minimum flow time for a traffic flow is 20 seconds,

Now, we can reduce **Z** subject to

$$d_2 + d_3 \geq 20$$

$$d_1 + d_3 \geq 20$$

$$d_1 + d_2 \geq 20$$

The example above is an illustration that ignores the traffic flow volume which is the number of cars that cross a certain point in a certain road within a certain time unit and the time durations that are given to traffic flows are created equally.

Software

As a part of this work, a web application software was created and developed in order to simulate the process of modeling the compatibility graph.

We found out that such a software hasn't been implemented before and a decision was made to create and develop it in order to model optimal and effective traffic intersections, avoid mistakes that may happen while modeling the graph and save the time spent on this process

Summary and results

- During the study, it was found that graph theory can be applied to the traffic light problem at crossroads in order to organize the movement and to avoid traffic accidents and also to minimize the total waiting time.
- We assigned a duration for each traffic stream and eventually we made a mathematical model of the optimal total waiting time and in order to minimize it later.
- A web application and algorithm were developed for this traffic light problem which can save time for modeling the compatibility graph. The algorithm showed efficiency in modeling the compatibility graph so that it only connects traffic streams (vertices) by an edge if and only if they can flow concurrently without causing collisions.
- We're planning to develop the algorithm so that it can also assign a flow time for each traffic stream and try to minimize the total waiting time depending on the volume of each stream.

**Thanks for your
attention!**