Nonlinear Programming Assignment 2022

SC42056 Optimization for Systems and Control

Maria de Neves de Fonseca, 5611679
Qingyi Ren, 5684803

October 2022

Delft University of Technology

I. Discrete-time state space model

In this assignment, we consider traffic signal control, the main control measure used in cities. The intersection of a traffic network, shown in Figure 1, will be used to implement the control strategy.

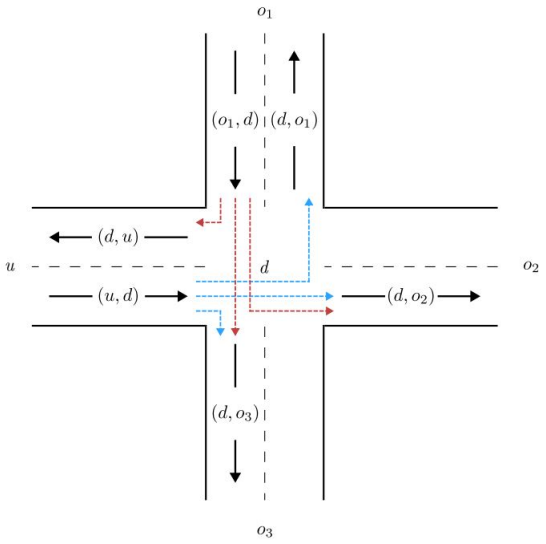

Fig. 1. Intersection in an urban traffic network

The traffic signal control of node d is considered in this assignment, where traffic flows entering link (u, d) via node u, as well as link (o_1, d) via node o_1, are given a priori. As a consequence, traffic flow dynamics of links (u, d) and (o_1, d) will be taken into account in the following. The discrete-time state space model that predicts the number of vehicles on link (u, d) and link (o_1, d) for the next simulation cycle $k + 1$ can be defined as follows:

$$x(k + 1) = f(x(k), u(k))$$
$$y(k) = g(x(k))$$

where f and g are vector-valued, nonlinear functions.

The states $x(k)$ of the model are defined as follows:

$$x(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \\ x_5(k) \\ x_6(k) \\ x_7(k) \\ x_8(k) \end{bmatrix} = \begin{bmatrix} n_{u,d}(k) \\ n_{o_1,d}(k) \\ q_{u,d,o_1}(k) \\ q_{u,d,o_2}(k) \\ q_{u,d,o_3}(k) \\ q_{o_1,d,u}(k) \\ q_{o_1,d,o_2}(k) \\ q_{o_1,d,o_3}(k) \end{bmatrix}$$

The inputs $u(k)$ of the model are represented by the green-time lengths ($g_{u,d,i}(k)$ and $g_{o_1,d,i}(k)$), which can be defined as:

$$u(k) = \begin{bmatrix} u_1(k) \\ u_2(k) \\ u_3(k) \\ u_4(k) \\ u_5(k) \\ u_6(k) \end{bmatrix} = \begin{bmatrix} g_{u,d,o_1}(k) \\ g_{u,d,o_2}(k) \\ g_{u,d,o_3}(k) \\ g_{o_1,d,u}(k) \\ g_{o_1,d,o_2}(k) \\ g_{o_1,d,o_3}(k) \end{bmatrix}$$

This brings the following discrete-time state space model:

$$x(k+1) = \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \\ x_5(k+1) \\ x_6(k+1) \\ x_7(k+1) \\ x_8(k+1) \end{bmatrix} = \begin{bmatrix} n_{u,d}(k+1) \\ n_{o_1,d}(k+1) \\ q_{u,d,o_1}(k+1) \\ q_{u,d,o_2}(k+1) \\ q_{u,d,o_3}(k+1) \\ q_{o_1,d,u}(k+1) \\ q_{o_1,d,o_2}(k+1) \\ q_{o_1,d,o_3}(k+1) \end{bmatrix} = \begin{bmatrix} x_1(k) + (\alpha_{u,d}^{enter}(k) - (\alpha_{u,d,o_1}^{leave}(k) + \alpha_{u,d,o_2}^{leave}(k) + \alpha_{u,d,o_3}^{leave}(k))) \cdot c \\ x_2(k) + (\alpha_{o_1,d}^{enter}(k) - (\alpha_{o_1,d,u}^{leave}(k) + \alpha_{o_1,d,o_2}^{leave}(k) + \alpha_{o_1,d,o_3}^{leave}(k))) \cdot c \\ x_3(k) + (\alpha_{u,d,o_1}^{arrive}(k) - \alpha_{u,d,o_1}^{leave}(k)) \cdot c \\ x_4(k) + (\alpha_{u,d,o_2}^{arrive}(k) - \alpha_{u,d,o_2}^{leave}(k)) \cdot c \\ x_5(k) + (\alpha_{u,d,o_3}^{arrive}(k) - \alpha_{u,d,o_3}^{leave}(k)) \cdot c \\ x_6(k) + (\alpha_{o_1,d,u}^{arrive}(k) - \alpha_{o_1,d,u}^{leave}(k)) \cdot c \\ x_7(k) + (\alpha_{o_1,d,o_2}^{arrive}(k) - \alpha_{o_1,d,o_2}^{leave}(k)) \cdot c \\ x_8(k) + (\alpha_{o_1,d,o_3}^{arrive}(k) - \alpha_{o_1,d,o_3}^{leave}(k)) \cdot c \end{bmatrix}$$

$$\Leftrightarrow x(k+1) = x(k) + \begin{bmatrix} (\alpha_{u,d}^{enter}(k) - (\alpha_{u,d,o_1}^{leave}(k) + \alpha_{u,d,o_2}^{leave}(k) + \alpha_{u,d,o_3}^{leave}(k))) \\ (\alpha_{o_1,d}^{enter}(k) - (\alpha_{o_1,d,u}^{leave}(k) + \alpha_{o_1,d,o_2}^{leave}(k) + \alpha_{o_1,d,o_3}^{leave}(k))) \\ (\alpha_{u,d,o_1}^{arrive}(k) - \alpha_{u,d,o_1}^{leave}(k)) \\ (\alpha_{u,d,o_2}^{arrive}(k) - \alpha_{u,d,o_2}^{leave}(k)) \\ (\alpha_{u,d,o_3}^{arrive}(k) - \alpha_{u,d,o_3}^{leave}(k)) \\ (\alpha_{o_1,d,u}^{arrive}(k) - \alpha_{o_1,d,u}^{leave}(k)) \\ (\alpha_{o_1,d,o_2}^{arrive}(k) - \alpha_{o_1,d,o_2}^{leave}(k)) \\ (\alpha_{o_1,d,o_3}^{arrive}(k) - \alpha_{o_1,d,o_3}^{leave}(k)) \end{bmatrix} \cdot c$$

with

$$\alpha_{u,d,o_1}^{leave}(k) = \min\left(\frac{\beta_{u,d,o_1} \cdot \mu_{u,d,o_1} \cdot u_1(k)}{c}, \frac{x_3(k)}{c} + \alpha_{u,d,o_1}^{arrive}(k), \frac{C_{d,o_1}(k)}{c}\right)$$

$$\alpha_{u,d,o_2}^{leave}(k) = \min\left(\frac{\beta_{u,d,o_2} \cdot \mu_{u,d,o_2} \cdot u_2(k)}{c}, \frac{x_4(k)}{c} + \alpha_{u,d,o_2}^{arrive}(k), \frac{C_{d,o_2}(k)}{c}\right)$$

$$\alpha_{u,d,o_3}^{leave}(k) = \min\left(\frac{\beta_{u,d,o_3} \cdot \mu_{u,d,o_3} \cdot u_3(k)}{c}, \frac{x_5(k)}{c} + \alpha_{u,d,o_3}^{arrive}(k), \frac{C_{d,o_3}(k)}{c}\right)$$

$$\alpha_{o_1,d,u}^{leave}(k) = \min\left(\frac{\beta_{o_1,d,u} \cdot \mu_{o_1,d,u} \cdot u_4(k)}{c}, \frac{x_6(k)}{c} + \alpha_{o_1,d,u}^{arrive}(k), \frac{C_{d,u}(k)}{c}\right)$$

$$\alpha_{o_1,d,o_2}^{leave}(k) = \min\left(\frac{\beta_{o_1,d,o_2}\cdot\mu_{o_1,d,o_2}\cdot u_5(k)}{c}, \frac{x_7(k)}{c} + \alpha_{o_1,d,o_2}^{arrive}(k), \frac{C_{d,o_2}(k)}{c}\right)$$

$$\alpha_{o_1,d,o_3}^{leave}(k) = \min\left(\frac{\beta_{o_1,d,o_3}\cdot\mu_{o_1,d,o_3}\cdot u_6(k)}{c}, \frac{x_8(k)}{c} + \alpha_{o_1,d,o_3}^{arrive}(k), \frac{C_{d,o_3}(k)}{c}\right)$$

$$\alpha_{u,d,i}^{arrive}(k) = \beta_{u,d,i}\cdot\alpha_{u,d}^{arrive}(k) \quad \text{for } i \in O_{u,d}$$

$$\alpha_{u,d}^{arrive}(k) = (1-\gamma_{u,d}(k))\cdot\alpha_{u,d}^{enter}(k-\tau_{u,d}(k)) + \gamma_{u,d}(k)\cdot\alpha_{u,d}^{enter}(k-\tau_{u,d}(k)-1)$$

$$\tau_{u,d} = \text{floor}\left\{\frac{(C_{u,d}(k)-q_{u,d})\cdot l_{veh}}{N_{u,d}^{lane}\cdot v_{u,d}^{lane}\cdot c}\right\}$$

$$\gamma_{u,d} = \text{frac}\left\{(C_{u,d}(k)-q_{u,d})\cdot l_{veh}, \quad N_{u,d}^{lane}\cdot v_{u,d}^{lane}\cdot c\right\}$$

$$q_{u,d} = \sum_{i\in O_{u,d}} q_{u,d,i}(k) = x_3(k) + x_4(k) + x_5(k)$$

$$\alpha_{o_1,d,j}^{arrive}(k) = \beta_{o_1,d,j}\cdot\alpha_{o_1,d}^{arrive}(k) \quad \text{for } j \in O_{o_1,d}$$

$$\alpha_{o_1,d}^{arrive}(k) = (1-\gamma_{o_1,d}(k))\cdot\alpha_{o_1,d}^{enter}(k-\tau_{o_1,d}(k)) + \gamma_{o_1,d}(k)\cdot\alpha_{o_1,d}^{enter}(k-\tau_{o_1,d}(k)-1)$$

$$\tau_{o_1,d} = \text{floor}\left\{\frac{(C_{o_1,d}(k)-q_{o_1,d})\cdot l_{veh}}{N_{o_1,d}^{lane}\cdot v_{o_1,d}^{lane}\cdot c}\right\}$$

$$\gamma_{o_1,d} = \text{frac}\left\{(C_{o_1,d}(k)-q_{o_1,d})\cdot l_{veh}, \quad N_{o_1,d}^{lane}\cdot v_{o_1,d}^{lane}\cdot c\right\}$$

$$q_{o_1,d} = \sum_{j\in O_{o_1,d}} q_{o_1,d,j}(k) = x_6(k) + x_7(k) + x_8(k)$$

For links (u,d), (o_1,d) maximum capacities are assumed to be constant in this assignment, and they are computed using link parameters:

$$C_{u,d} = \frac{N_{u,d}^{lane}\cdot l_{u,d}}{l_{veh}}$$

$$C_{o_1,d} = \frac{N_{o_1,d}^{lane}\cdot l_{o_1,d}}{l_{veh}}$$

$$\alpha_{u,d}^{enter}(k) = \begin{cases} 2300 + 10E_1 & \text{[veh/h]} \quad \text{if} \quad k \le 20 \\ 1800 + 10E_2 & \text{[veh/h]} \quad \text{if} \quad 20 < k \le 40 \\ 2100 + 10E_3 & \text{[veh/h]} \quad \text{if} \quad k > 40 \end{cases}$$

$$\alpha_{o_1,d}^{enter}(k) = 1800 + 10\cdot E_1 \quad \text{[veh/h]} \quad \forall k$$

and it is given that

c	$N_{u,d}^{lane}$	$v_{u,d}^{free}$	l_{veh}	$l_{u,d}$	β_{u,d,o_1}	β_{u,d,o_2}	β_{u,d,o_3}	$\mu_{u,d}$
60/3600 $[h]$	3	60 $[\frac{km}{h}]$	7e-3 $[km]$	4 $[km]$	0.4	0.3	0.3	1800 $[\frac{veh}{h}]$

TABLE I
PARAMETERS FOR LINK (u, d)

c	$N_{o_1,d}^{lane}$	$v_{o_1,d}^{free}$	l_{veh}	$l_{o_1,d}$	$\beta_{o_1,d,u}$	β_{o_1,d,o_2}	β_{o_1,d,o_3}	$\mu_{o_1,d}$
60/3600 $[h]$	3	50 $[\frac{km}{h}]$	7e-3 $[km]$	4 $[km]$	0.3	0.4	0.3	1700 $[\frac{veh}{h}]$

TABLE II
PARAMETERS FOR LINK (o_1, d)

$$C_{d,u}(k) = \begin{cases} 20 - \frac{E_3}{2} & \text{[veh]} \quad \text{if} \quad k \leq 40 \\ 20 + \frac{E_3}{2} & \text{[veh]} \quad \text{if} \quad k > 40 \end{cases}$$

$$C_{d,o_1}(k) = C_{d,o_2}(k) - \frac{2 \cdot k}{E_1 + 1} \quad \text{[veh]} \quad \forall k$$

$$C_{d,o_2}(k) = \begin{cases} 10 + \frac{E_2}{2} & \text{[veh]} \quad \text{if} \quad k \leq 20 \\ 10 + \frac{E_2}{2} - \frac{2 \cdot k}{E_1 + 1} & \text{[veh]} \quad \text{if} \quad 20 < k \leq 35 \\ 20 + \frac{E_2}{2} & \text{[veh]} \quad \text{if} \quad 35 < k \leq 45 \\ 20 + \frac{E_3}{2} + \frac{2 \cdot k}{E_1 + 1} & \text{[veh]} \quad \text{if} \quad k > 45 \end{cases}$$

$$C_{d,o_3}(k) = \begin{cases} 10 - \frac{E_3}{2} & \text{[veh]} \quad \text{if} \quad k \leq 20 \\ 10 + \frac{E_3}{2} & \text{[veh]} \quad \text{if} \quad k > 20 \end{cases}$$

Note that all capacities $C_{i,j}(.)$ must have nonnegative value. Therefore, we assumed $C_{i,j}(.)$ equal to zero if the expression given above returns a negative value for the combination of parameters E1, E2, E3.

The total time spent (TTS) by the drivers on links (u, d) and (o_1, d) is taken as the output of the system and it can be defined as:

$$y(k) = (n_{u,d}(k) + n_{o_1,d}(k)) \cdot c = [c \quad c \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]x(k) = (x_1(k) + x_2(k)) \cdot c$$

which depends only indirectly on the input values.

Note: We chose to define $\alpha_{u,d}^{enter}$ and $\alpha_{o_1,d}^{enter}$ as being 0 for the first iteration $k = 0$ since in that moment we have no cars in the links or in the queues.

II. TIME LENGTH OF THE GREEN LIGHT THAT MINIMIZES THE TOTAL TIME SPENT

A. Problem formulation - optimising over the whole hour horizon

Using the model built in section I, we can now formulate the optimization problem to find the time length of the green light at node d for every cycle that minimizes the TTS by the drivers on links (u, d) and (o_1, d) for the following hour (i.e., for the period $[0, 60c]\forall k$). From our understanding, for this problem, since we have perfect knowledge of the future inputs and capacities of the traffic network, it is equivalent to optimise the green light such that it minimizes the value of TTS for every cycle, or to optimise once for the whole hour horizon. For this problem, we opted to solve and optimise for the entire hour at once for a vector $u_1(k) \in \mathrm{R}^{1\times 60}$. An alternative will be provided showing that we obtained the same results for the optimisation for every cycle, and therefore, since no drastic improvements were made (or very little) we continue analysing the case for optimising the whole hour.

For this, we assume that drivers can always turn right at the intersection, that the green-light duration for going straight and turning left is the same, and that the green-light durations of the the green-light

phase when the vehicles in link (u, d) can cross the intersection, and the red-light phase when the vehicles in link (u, d) are not allowed to cross the intersection are complementary. The same consideration applies to link $(o1, d)$. It is also assumed that at $k = 0$ there is no traffic, i.e., the numbers of vehicles in links (u, d) and (d, o_1) are equal to 0, and the numbers of vehicles waiting in the queues are 0 as well. For simplicity, we consider the green time as continuous variables constrained in the interval $[15s, 45s]$. Since we have that all parameters must be expressed with the same units (h, km, veh), the time interval for the values of the green time is divided by 3600 so it appears in hours. All these assumptions are equivalent to the following minimization problem:

$$\min_{u(k)} \sum_{k=0}^{60c} y(k) = \min_{u(k)} \sum_{k=0}^{60c} (n_{u,d}(k) + n_{o_1,d}(k)) \cdot c = \min_{u(k)} \sum_{k=0}^{60c} (x_1(k) + x_2(k)) \cdot c$$

We also know that

$$n_{u,d}(k+1) = n_{u,d}(k) + (\alpha_{u,d}^{enter}(k) - (\alpha_{u,d,o_1}^{leave}(k) + \alpha_{u,d,o_2}^{leave}(k) + \alpha_{u,d,o_3}^{leave}(k))) \cdot c$$
$$n_{o_1,d}(k+1) = n_{o_1,d}(k) + (\alpha_{o_1,d}^{enter}(k) - (\alpha_{o_1,d,u}^{leave}(k) + \alpha_{o_1,d,o_2}^{leave}(k) + \alpha_{o_1,d,o_3}^{leave}(k))) \cdot c$$

This shows that optimising $u(k)$, implying optimising also $\alpha_{u,d,i}^{leave}(k)$ for $i \in O_{u,d}$ and $\alpha_{o_1,d,j}^{leave}(k)$ for $j \in O_{o_1,d}$, actually optimises the number of cars for the next step. Thus, we have that our minimisation problem can be written as

$$\min_{u(k)} \sum_{k=0}^{60c} y(k+1) = \min_{u(k)} \sum_{k=0}^{60c} (n_{u,d}(k+1) + n_{o_1,d}(k+1)) \cdot c = \min_{u(k)} \sum_{k=0}^{60c} (x_1(k+1) + x_2(k+1)) \cdot c \quad (1)$$

s.t.

$$u_3(k) = u_4(k) = c$$
$$u_1(k) = u_2(k)$$
$$u_5(k) = u_6(k)$$
$$u_5(k) = c - u_2(k)$$
$$15/3600 \leq u(k) \leq 45/3600$$

B. Alternative formulation - optimising for each step

As referred before, an alternative would be optimising for each cycle which would transform the minimisation problem from the subsection above into the following:

$$\min_{u(k)} y(k+1) = \min_{u(k)} (n_{u,d}(k+1) + n_{o_1,d}(k+1)) \cdot c = \min_{u(k)} (x_1(k+1) + x_2(k+1)) \cdot c \quad (2)$$

s.t.

$$u_3(k) = u_4(k) = c$$
$$u_1(k) = u_2(k)$$
$$u_5(k) = u_6(k)$$
$$u_5(k) = c - u_2(k)$$
$$15/3600 \leq u(k) \leq 45/3600$$

To obtain the final TTS in this case we would sum in the end all the TTS obtained for the 60 different optimisation problems. If we were in a real life situation this would be the best approach to use since in reality we wouldn't know or be able to predict road incidents like accidents for instance or obstructions. So it would make sense to optimise for smaller time steps like each minute instead of optimising for the entirety of one hour at once, since for this case we can only optimize based on the current situation. However, we assume that we perfectly know the future inputs and capacities of the traffic network, as they are specified in section I. Thus, from our understanding, in this situation we would rather optimise for the entire hour already since we have information about the things that usually can be unpredictable on the roads.

C. Notes on convexity

Looking at the objective function presented in (2) we can note an interesting thing. The problem can be extended to

$$\min_{u(k)} y(k+1) = \min_{u(k)} (n_{u,d}(k) + (\alpha_{u,d}^{enter}(k) - \alpha_{u,d}^{leave}(k)) \cdot c + n_{o_1,d}(k) + (\alpha_{o_1,d}^{enter}(k) - \alpha_{o_1,d}^{leave}(k)) \cdot c) \cdot c$$

with

$$\alpha_{u,d}^{leave}(k) = \alpha_{u,d,o_1}^{leave}(k) + \alpha_{u,d,o_2}^{leave}(k) + \alpha_{u,d,o_3}^{leave}(k)$$
$$\alpha_{o_1,d}^{leave}(k) = \alpha_{o_1,d,u}^{leave}(k) + \alpha_{o_1,d,o_2}^{leave}(k) + \alpha_{o_1,d,o_3}^{leave}(k)$$

If we optimise each time for a certain step $k+1$, then we know that the values from previous steps are already optimal. Then we can assume that the values of $n_{u,d}(k)$ and $n_{o_1,d}(k)$ are independent of the optimisation variable for this iteration and cannot change. Note also that the values for $\alpha_{u,d}^{enter}(k)$ and $\alpha_{o_1,d}^{enter}(k)$ are also constants that are predefined before. These values can be ignored for the minimisation problem and added up in the end to get the final result (in theory). So now we have that

$$\min_{u(k)} y(k+1) = \min_{u(k)} -(\alpha_{u,d,o_1}^{leave}(k) + \alpha_{u,d,o_2}^{leave}(k) + \alpha_{u,d,o_3}^{leave}(k) + \alpha_{o_1,d,u}^{leave}(k) + \alpha_{o_1,d,o_2}^{leave}(k) + \alpha_{o_1,d,o_3}^{leave}(k)) \cdot c^2 \quad (3)$$

These functions are represented as

$$\alpha_{u,d,o_1}^{leave}(k) = \min\left(\frac{\beta_{u,d,o_1} \cdot \mu_{u,d,o_1} \cdot u_1(k)}{c}, \frac{x_3(k)}{c} + \alpha_{u,d,o_1}^{arrive}(k), \frac{C_{d,o_1}(k)}{c}\right)$$
$$\alpha_{u,d,o_2}^{leave}(k) = \min\left(\frac{\beta_{u,d,o_2} \cdot \mu_{u,d,o_2} \cdot u_2(k)}{c}, \frac{x_4(k)}{c} + \alpha_{u,d,o_2}^{arrive}(k), \frac{C_{d,o_2}(k)}{c}\right)$$
$$\alpha_{u,d,o_3}^{leave}(k) = \min\left(\frac{\beta_{u,d,o_3} \cdot \mu_{u,d,o_3} \cdot u_3(k)}{c}, \frac{x_5(k)}{c} + \alpha_{u,d,o_3}^{arrive}(k), \frac{C_{d,o_3}(k)}{c}\right)$$
$$\alpha_{o_1,d,u}^{leave}(k) = \min\left(\frac{\beta_{o_1,d,u} \cdot \mu_{o_1,d,u} \cdot u_4(k)}{c}, \frac{x_6(k)}{c} + \alpha_{o_1,d,u}^{arrive}(k), \frac{C_{d,u}(k)}{c}\right)$$
$$\alpha_{o_1,d,o_2}^{leave}(k) = \min\left(\frac{\beta_{o_1,d,o_2} \cdot \mu_{o_1,d,o_2} \cdot u_5(k)}{c}, \frac{x_7(k)}{c} + \alpha_{o_1,d,o_2}^{arrive}(k), \frac{C_{d,o_2}(k)}{c}\right)$$
$$\alpha_{o_1,d,o_3}^{leave}(k) = \min\left(\frac{\beta_{o_1,d,o_3} \cdot \mu_{o_1,d,o_3} \cdot u_6(k)}{c}, \frac{x_8(k)}{c} + \alpha_{o_1,d,o_3}^{arrive}(k), \frac{C_{d,o_3}(k)}{c}\right)$$

We can understand how our set behaves by analysing these functions. Even though we cannot say anything about convexity for minimum functions (since we cannot be sure if they are convex or concave), we can reformulate these functions by applying $\min f(x) = \max -f(-x)$.

$$\alpha^{leave}_{u,d,o_1}(k) = -\max\left(-\frac{\beta_{u,d,o_1}\cdot\mu_{u,d,o_1}\cdot u_1(k)}{c}, -\frac{x_3(k)}{c} - \alpha^{arrive}_{u,d,o_1}(k), -\frac{C_{d,o_1}(k)}{c}\right)$$

$$\alpha^{leave}_{u,d,o_2}(k) = -\max\left(-\frac{\beta_{u,d,o_2}\cdot\mu_{u,d,o_2}\cdot u_2(k)}{c}, -\frac{x_4(k)}{c} - \alpha^{arrive}_{u,d,o_2}(k), -\frac{C_{d,o_2}(k)}{c}\right)$$

$$\alpha^{leave}_{u,d,o_3}(k) = -\max\left(-\frac{\beta_{u,d,o_3}\cdot\mu_{u,d,o_3}\cdot u_3(k)}{c}, -\frac{x_5(k)}{c} - \alpha^{arrive}_{u,d,o_3}(k), -\frac{C_{d,o_3}(k)}{c}\right)$$

$$\alpha^{leave}_{o_1,d,u}(k) = -\max\left(-\frac{\beta_{o_1,d,u}\cdot\mu_{o_1,d,u}\cdot u_4(k)}{c}, -\frac{x_6(k)}{c} - \alpha^{arrive}_{o_1,d,u}(k), -\frac{C_{d,u}(k)}{c}\right)$$

$$\alpha^{leave}_{o_1,d,o_2}(k) = -\max\left(-\frac{\beta_{o_1,d,o_2}\cdot\mu_{o_1,d,o_2}\cdot u_5(k)}{c}, -\frac{x_7(k)}{c} - \alpha^{arrive}_{o_1,d,o_2}(k), -\frac{C_{d,o_2}(k)}{c}\right)$$

$$\alpha^{leave}_{o_1,d,o_3}(k) = -\max\left(\frac{-\beta_{o_1,d,o_3}\cdot\mu_{o_1,d,o_3}\cdot u_6(k)}{c}, -\frac{x_8(k)}{c} - \alpha^{arrive}_{o_1,d,o_3}(k), -\frac{C_{d,o_3}(k)}{c}\right)$$

Once again, since for the separate optimisation problems we can assume that the values from the previous steps are already optimal, we can also look at $\alpha^{arrive}_{o_1,d,j}$, $q_{o_1,d,j}$ for $j \in O_{o_1,d}$, and $\alpha^{arrive}_{u,d,i}$, $q_{u,d,i}$ for $i \in O_{u,d}$ as unchangeable values independent from the optimisation variable for this iteration.

We can then notice that the terms inside the $\max()$ are either constants which do not affect convexity or linear functions which can be considered convex sets. The $\max()$ function conserves convexity if the terms inside of it are convex. Therefore, in this case all the $\max()$ functions are convex. The sum of convex sets is also a convex set. Thus, we have that $\alpha^{leave}_{u,d} = \sum_{i\in O_{u,d}} \alpha^{leave}_{u,d,i}$ and $\alpha^{leave}_{o_1,d} = \sum_{j\in O_{o_1,d}} \alpha^{leave}_{o_1,d,j}$ are both convex leading to convexity of (3) and, thus, of (2).

We could try to extend this analysis to the objective function provided in (1). However, if we look at this problem as a global optimisation problem for the entire hour we notice that we cannot assume the values from the previous cycle as optimal (or as constants), ignoring their effect on the objective function. Since the optimisation is run at once for a 60 minutes cycle we have interdependence of functions and recursive behaviour. We believe that because of this we cannot assume convexity for the problem stated in (1) since the next cycle $k+1$ is dependent on all previous cycles. In the previous case for the objective function in (2) we are in the presence of 60 separate optimisation problems while for (1) we have one global optimisation problem where each cycle is dependent of previous ones.

As a side note: the same assumptions about simplification of the objective function can be made for the objective function (1). However, this was tried in Matlab and the results for the TTS were slightly worse (TTS=1.1640e+03 $[veh \cdot h]$ for method displayed in (3)). Also the simulation for the minimisation problem stated in (2) was run providing very similar results to the ones for the minimisation problem in (1) (TTS=1.157825e+03 $[veh \cdot h]$). Therefore, we continue discussing the problem using the objective function in (1).

D. Algorithm selection

Clearly from the state-space representation, we are in the presence of a nonlinear optimization problem with respect to the input variable $u(k)$. The equality constraints can be inserted inside the objective function in terms of just $u_1(k)$ and the cycle time c as

$$u_2(k) = u_1(k)$$
$$u_3(k) = u_4(k) = c$$
$$u_5(k) = c - u_1(k)$$
$$u_6(k) = c - u_1(k)$$

As for the four inequality constraints, these can be set as lower and upper bounds for $u(k)$. Bearing all these in mind, we can now decide which algorithm to use to solve this problem. From the list of Matlab solvers from the optimization toolbox, the most appropriate one for our specific problem is `fmincon`, which finds the minimum of a problem specified by

$$\min_x f(x) \quad \text{such that} \begin{cases} c(x) \leq 0 \\ ceq(x) = 0 \\ A \cdot x \leq b \\ Aeq \cdot x = beq \\ lb \leq x \leq ub \end{cases}$$

where b and beq are vectors, A and Aeq are matrices, $c(x)$ and $ceq(x)$ are functions that return vectors, and $f(x)$ is a function that returns a scalar. $f(x)$, $c(x)$, and $ceq(x)$ can be nonlinear functions. x, lb, and ub can be passed as vectors or matrices [2]. In our case, since we can substitute the constraints into the problem itself, we are left with only the lower and upper bound constraints that can be defined straight away in the `fmincon` funtion in Matlab.

`fmincon` has five algorithm options: 'interior-point', 'trust-region-reflective', 'sqp', 'sqp-legacy' and 'active-set' [3].

1) 'interior-point' is recommended for large-scale problems (which is not our case).
2) 'sqp' satisfies bounds at all iterations. The algorithm can recover from NaN or Inf results and it is not a large-scale algorithm.
3) 'sqp-legacy' is similar to 'sqp', but usually is slower and uses more memory.
4) 'active-set' can take large steps, which adds speed. The algorithm is effective on some problems with nonsmooth constraints.
5) 'trust-region-reflective' requires a gradient to be supplied in the objective function, and allows only bounds or linear equality constraints, but not both. Within these limitations, the algorithm handles both large sparse problems and small dense problems efficiently.

Knowing this, the 'sqp' algorithm was chosen, since we are not in the presence of a large-scale problem (we only optimise for one variable - length of the green light time), it is not necessary to calculate the gradient to use this algorithm, and it can satisfy bounds at all iterations.

III. Optimizing the traffic flows by choosing different starting points for green time lengths of link (u, d)

A. Results

Now we use the chosen optimization algorithm to optimize the traffic flows by choosing two different starting points for the green time lengths of link (u, d): $g_{u,d,i}(k) = 15s \quad \forall k$ and $g_{u,d,i}(k) = 45s \quad \forall k$. We can do this by providing a starting vector $x_0 \in \mathrm{R}^{1\times 60}$ with the desired initial value for all entries. Similarly, both lower and upper bounds are also vectors of dimension $\mathrm{R}^{1\times 60}$.

```
lb = ones(1,61)*15/3600;
ub = ones(1,61)*45/3600;

x0 = lb; % or x0 = ub;

rng default % For reproducibility
options = optimoptions('fmincon','Algorithm','sqp','MaxFunctionEvaluations',8e4,'
    TolFun',5e-2);
[x1,fval1,exitflag1,output1] = fmincon(@(x1)myfun(x1,C_u_d,C_o1_d,C_d_u,C_d_o3,C_d_o2
    ,C_d_o1,alpha_enter_o1_d,alpha_enter_u_d),x0,[],[],[],[],lb,ub,[],options)
```

In a time-span of 1h, the evaluations were carried out every 60 seconds (or every minute), which brings a total of 60 evaluations (since k goes from 0 to 60c). The simulations for the green light time length are shown bellow in Figure 2 for the entire simulation cycle for both links.

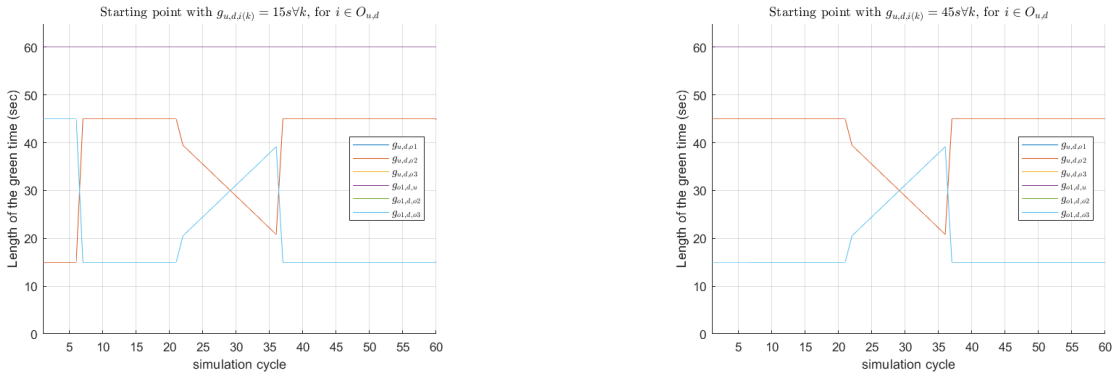

Fig. 2. Plots of inputs $u(k)$ for optimised problem with starting points $x_0 = 15s$ (right) and $x_0 = 45s$ (left) through one hour

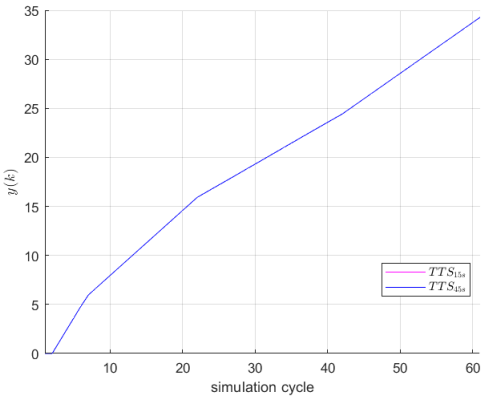

Fig. 3. Plot of the TTS for optimised problem with starting points $x_0 = 15s$ and $x_0 = 45s$ through one hour

B. Discussion of the results

It can be seen from the plots shown in Figure 2 that for the first simulation time steps, neither solution moves from its starting point. Of course, this makes sense since for $k = 0$ the number of vehicles in both links is equal to 0, as well as the number of vehicles waiting in the queues. Note that also at this point the vehicles are still traveling on the respective links, not having yet entered any queue. Thus, we can conclude that no cars are ready to leave yet, and, therefore, no optimisation is necessary.

For the rest of the simulation, there is not a significant difference between the two obtained solutions, in fact they are pretty much the same. By looking at the plots for the TTS for both starting points, we found that both plots completely overlap, no matter which value we have for x_0.

TTS_{15s}	TTS_{45s}
1.1578257e+03 $[veh \cdot h]$	1.1578253e+03 $[veh \cdot h]$

TABLE III
TTS VALUES OVER THE ENTIRE SIMULATION PERIOD FOR DIFFERENT CASES

From III we can check that the TTS values only differ around the seventh decimal place after the decimal point, which is negligible. This small difference can be due to how the SQP algorithm works. However, for any starting point the function value will always be around the neighbourhood of 1157.8 $[veh \cdot h]$. These results raise again some suspicion on whether our problem can be assumed convex or not, since for convex problems, the same solution should be obtained regardless of the starting point.

If we look at the problem from a simulation point of view, when using solvers from optimisation toolboxes as with Matlab we cannot be 100% sure our solution is the global/local optimum because depending on the starting point or step size of the solvers we can end up in different places inside the function. However, as seen in our case we will always end up in the neighbourhood of a global/local minimum so our solution will be also in the neighbourhood of the global/local optimum. So for this case, a way to improve our solutions could be trying several starting points and see if that makes any difference on the result or change the step size of the solver itself. Also playing with the tolerances regarding the constraints or the objective function can be helpful to achieve better results.

As to prove if the solution obtained is the global optimum let's bring back again the two possible formulations for this problem. In section II C. we have proven that if we formulate this problem as 60 separate optimisation problems with objective functions as in (2) we are optimising over a convex set. Thus, we can straight away conclude that the solution found is the global optimum since convex functions only have one minimum. However, we have also discussed that looking at the problem as one global optimisation problem with objective function (1) it becomes hard to conclude if this is also convex. For this case we cannot be sure if the solution obtained is the global or just a local optimum since we do not know the behaviour of the function.

As an experiment, we plotted the value of the objective function for the no-control case for several values of the green light time, meaning the green light time is kept constant for the entire simulation - Figure 4. From this plot we can check that our objective function has in fact only one global minimum since it decreases for any green light time below 32s and starts increasing after that.

Even thought we couldn't mathematically prove that the solution obtained is a global optimum, based on the results from simulations we could possible derive that the obtained solution is close to the global optimum.

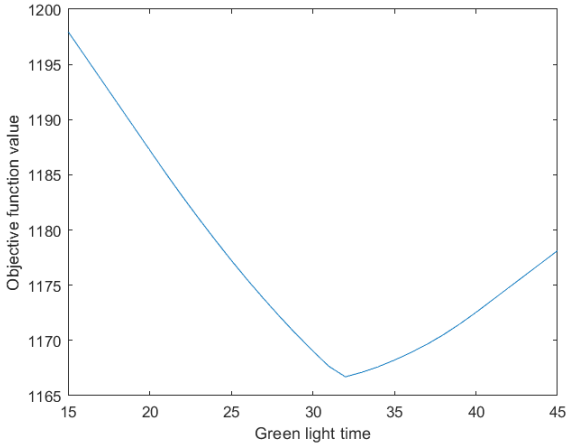

Fig. 4. Value of the objective function for the no-control case for several values of the green light time

IV. Simulation for different starting points and for non-control case

To better understand the effect of optimisation, now we compare our results from section III with the ones obtained for the no-control case ($g_{u,d,i}(k) = g_{o_1,d,i}(k) = 30s \forall k$, for $i \in O_{u,d}$ and $j \in O_{o_1,d}$). To compare the behaviour of the system for all methods, the plots for the states, inputs and TTS values are shown. To apply the no-control method, the green light time length is kept constant as $u_1(k) = 30s \quad \forall k$.

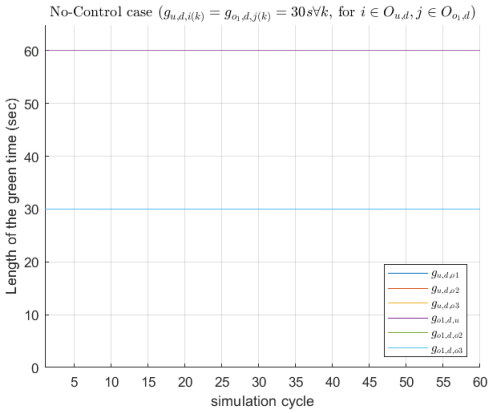

Fig. 5. Plots of inputs $u(k)$ for no-control problem with $u(k) = 30s$

A. Simulation and discussion of the results

For simplicity, only the plot of $u_1(k)$ is going to be shown in the next sections to better understand the behaviour of the green light for the several methods.

The TTS values over the entire simulation period were obtained for the different cases (including the no-control one) and are presented in table IV.

When looking at the plot of the TTS for all three cases we can see that they do not vary much from each other. This shows that due to the formulation of this problem, if we want to optimise the TTS by changing the values of the green light time, we can simply set $u_1(k) = 30s$ which will provide already a very close result to the optimal one.

TTS_{15s}	TTS_{45s}	TTS_{30s}
1.1578257e+03 $[veh \cdot h]$	1.1578253e+03$[veh \cdot h]$	1.1690455e+03$[veh \cdot h]$

TABLE IV

TTS VALUES OVER THE ENTIRE SIMULATION PERIOD FOR DIFFERENT CASES

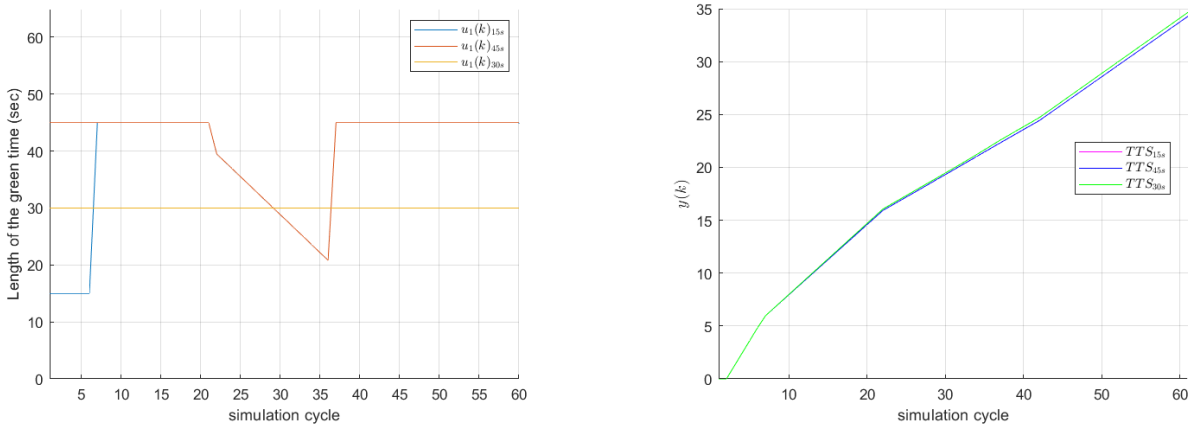

Fig. 6. Plots of inputs $u_1(k)$ (left) and objective function value (right) for optimised problem with starting points $x_0 = 15s$ and $x_0 = 45s$ and for no control case through one hour

On one hand, when looking at the graph of the number of vehicles in link (u, d) for the different methods, we can check that having a constant $u_1(k) = 30s$ results in a larger number of vehicles present in the link when compared to the optimised values. This makes sense, since for the optimised values we see that the green light time increases around $k = 6$. On the other hand, by increasing this green light time for link (u, d), we decrease it for the link (o_1, d), which results in an increase of the number of vehicles in link (o_1, d) when compared to the no-control case.

Obviously the queue length of the sub-stream turning to link (d, i) for $i \in O_{u,d}$ will also be larger for the no-control case when compared to the optimised values. Also, makes sense that the queue length of the sub-stream turning to link (d, j) for $j \in O_{o_1,d}$ is larger for the optimised case comparing to for the no-control case. This also happened due to the same explanation as before. If the number of vehicles in a link increases then the queue length of the sub-stream turning to other link also increases.

For either case, the queue lengths of the sub-streams turning to the right are the shortest since the green light time does not have influence in the lanes turning right.

As a side note, as stated before, we can see that increasing the value of the green light time for the optimised case decreases the number of vehicles in link (u, d) while increases the number of vehicles in link (o_1, d), which already has a higher number of vehicles. So we can see this optimisation method does not prioritise the lane with more vehicles in it. So link (u, d) has less number of vehicles and larger green light times at the expense of the higher number of vehicles in link (o_1, d). This happens since we are optimising for the TTS, so as long as TTS by the cars in each link is minimised, we do not care about the vehicles disposition in the respective links. In real life this would be something that should be taken into account since it becomes a problem about fairness for all drivers.

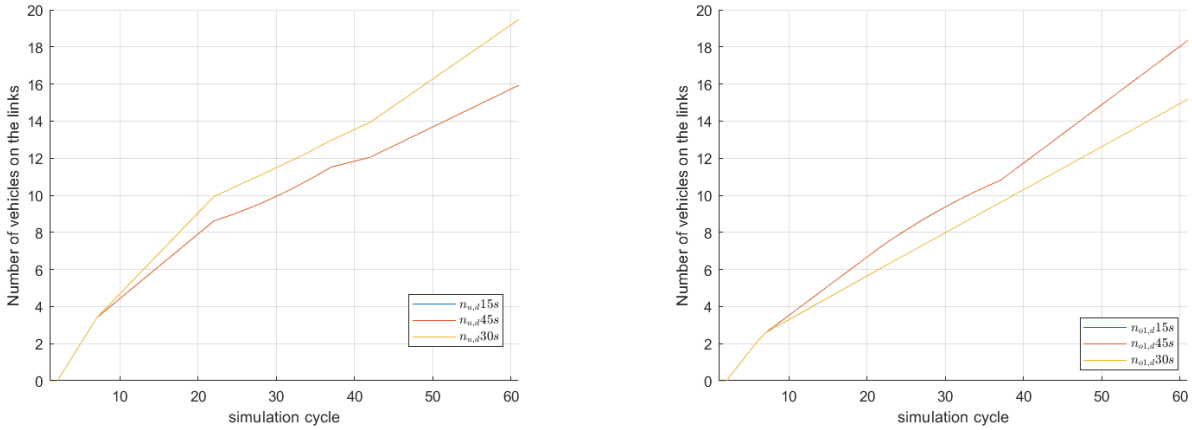

Fig. 7. Plots of states $n_{u,d}(k)$ (left) and $n_{o_1,d}(k)$ (right) for optimised problem with starting points $x_0 = 15s$ and $x_0 = 45s$ and for no control case through one hour

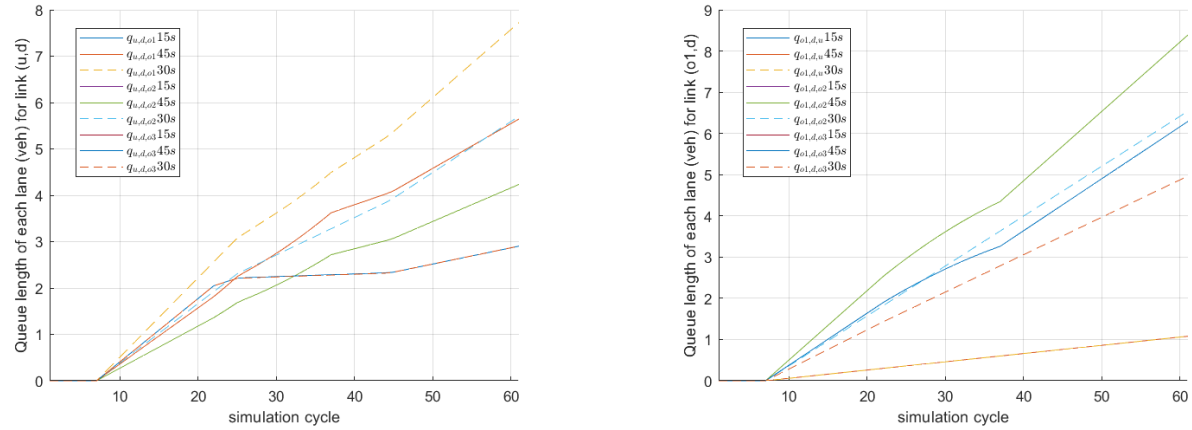

Fig. 8. Plots of states $q_{u,d,i}(k)$ for $i \in O_{u,d}$ (left) and $q_{o_1,d,j}(k)$ for $j \in O_{o_1,d}$(right) for optimised problem with starting points $x_0 = 15s$ and $x_0 = 45s$ and for no control case through one hour

V. Changing optimization variables to integer values

A. Genetic algorithm

Now the green light times are limited to the given discrete time set $\{15s, 20s, 25s, 30s, 35s, 40s, 45s\}$. To solve this problem, we use an optimization algorithm that can directly deal with integer optimization variables - genetic algorithm. However, these variables need to be sequential integers, which is not the case for our discrete time set. To avoid this issue we convert the given discrete time set to $u_1'(k) \in \{3, 4, 5, 6, 7, 8, 9\}$, and set the lower bound as 3s and upper bound as 9s for the initial problem. We also need to adapt the problem formulated in sections I and II as follows:

$$\alpha^{leave}_{u,d,o_1}(k) = \min\left(\frac{\beta_{u,d,o_1} \cdot \mu_{u,d,o_1} \cdot 5u_1'(k)}{c}, \frac{x_3(k)}{c} + \alpha^{arrive}_{u,d,o_1}(k), \frac{C_{d,o_1}(k)}{c}\right)$$

$$\alpha^{leave}_{u,d,o_2}(k) = \min\left(\frac{\beta_{u,d,o_2} \cdot \mu_{u,d,o_2} \cdot 5u_2'(k)}{c}, \frac{x_4(k)}{c} + \alpha^{arrive}_{u,d,o_2}(k), \frac{C_{d,o_2}(k)}{c}\right)$$

$$\alpha^{leave}_{u,d,o_3}(k) = \min\left(\frac{\beta_{u,d,o_3}\cdot\mu_{u,d,o_3}\cdot 5u'_3(k)}{c}, \frac{x_5(k)}{c} + \alpha^{arrive}_{u,d,o_3}(k), \frac{C_{d,o_3}(k)}{c}\right)$$

$$\alpha^{leave}_{o_1,d,u}(k) = \min\left(\frac{\beta_{o_1,d,u}\cdot\mu_{o_1,d,u}\cdot 5u'_4(k)}{c}, \frac{x_6(k)}{c} + \alpha^{arrive}_{o_1,d,u}(k), \frac{C_{d,u}(k)}{c}\right)$$

$$\alpha^{leave}_{o_1,d,o_2}(k) = \min\left(\frac{\beta_{o_1,d,o_2}\cdot\mu_{o_1,d,o_2}\cdot 5u'_5(k)}{c}, \frac{x_7(k)}{c} + \alpha^{arrive}_{o_1,d,o_2}(k), \frac{C_{d,o_2}(k)}{c}\right)$$

$$\alpha^{leave}_{o_1,d,o_3}(k) = \min\left(\frac{\beta_{o_1,d,o_3}\cdot\mu_{o_1,d,o_3}\cdot 5u'_6(k)}{c}, \frac{x_8(k)}{c} + \alpha^{arrive}_{o_1,d,o_3}(k), \frac{C_{d,o_3}(k)}{c}\right)$$

Note that the values of $u'(k)$ are multiplied by 5 for calculating $\alpha^{leave}_{o_1,d,j}(k)$ for $j \in O_{o_1,d}$ and $\alpha^{leave}_{u,d,i}(k)$ for $i \in O_{u,d}$ which are then used for the objective function as defined in section I. Also note that $u'(k)$ needs to be divided by 3600 in order to get the values in hours instead of seconds.

$$\min_{u'(k)} \sum_{k=0}^{60c} y(k+1) = \min_{u'(k)} \sum_{k=0}^{60c} (n_{u,d}(k+1) + n_{o_1,d}(k+1))\cdot c = \min_{u'(k)} \sum_{k=0}^{60c} (x_1(k+1) + x_2(k+1))\cdot c$$

s.t.

$$u'_3(k) = u'_4(k) = c$$
$$u'_1(k) = u'_2(k)$$
$$u'_5(k) = u'_6(k)$$
$$u'_5(k) = c - u'_2(k)$$
$$3 \le u'(k) \le 9$$

B. Matlab simulation and results

We can now use the the Matlab solver `ga()` that uses the genetic algorithm as optimization method. By using the syntax `x=ga(fun,nvars,A,b,Aeq,beq,lb,ub,nonlcon,intcon,options)`, we can obtain a integer solution for our optimization variables set in `intcon`. In Matlab, we confine our input to integer values, by setting `intcon=1:61`, which includes the entirety of our input array.

```
lb=3*ones(1,61);
ub=9*ones(1,61);

opts = optimoptions('ga','ConstraintTolerance',1e-6, 'MaxStallGenerations',1000,'
   MaxGenerations',1000, 'PlotFcn', @gaplotbestf);
rng default % For reproducibility
[x4,fval4,eflag4,outpt4] = ga(@(x4)integer_sol(x4,C_u_d,C_o1_d,C_d_u,C_d_o3,C_d_o2,
   C_d_o1,alpha_enter_o1_d,alpha_enter_u_d),61,[],[],[],[],lb,ub,[],1:61,opts)
```

Again, we optimise for an entire hour instead of for each step. So we solve this constrained integer problem for all 60 variables of $u_1(k)$ at once. The simulation results for the green-time lengths ($g_{u,d,i}(k)$ for $i \in O_{u,d}$ and $g_{o_1,d,j}(k)$ for $j \in O_{o_1,d}$), the number of vehicle in the links $n_{u,d}(k)$ and $n_{o_1,d}(k)$, and the number of vehicles waiting in the queue of links (d,u) and (d,o_1) are shown in Figure 9 separately.

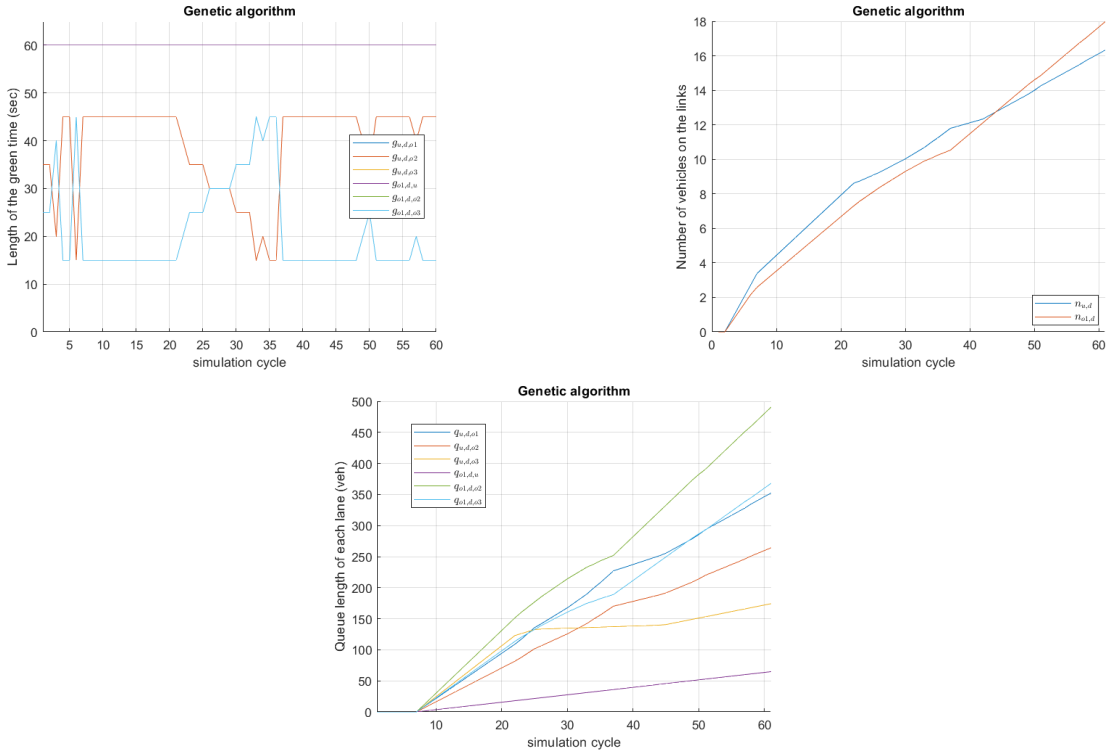

Fig. 9. Plots of inputs $u(k)$, number of vehicles in the links and queue length of each lane

C. *Comparison between the different approaches*

Once more, the TTS times for the different approaches over the entire simulation time were computed and the following values were obtained:

TTS_{15s}	TTS_{45s}	TTS_{30s}	TTS_{ga}
1.1578257e+03$[veh \cdot h]$	1.1578253e+03 $[veh \cdot h]$	1.1690455e+03 $[veh \cdot h]$	1.1584092e+03$[veh \cdot h]$

TABLE V

TTS VALUES OVER THE ENTIRE SIMULATION PERIOD FOR DIFFERENT APPROACHES

For the genetic algorithm, in the beginning of the simulation cycle we see that since we are in the presence of a discrete set, the optimised values for the green light time vary between 15s and 45s. Once cars start moving on the links and entering the queues in each lane, we see that the green light time obtained from the genetic algorithm follows quite well the behaviour from the first optimisation method from section III. Noticing this, the other plots for the number of vehicles in each link and for the queue lengths for each lane make sense, since they are similar to the ones from the first optimised result for the different starting points.

When looking at the plot of the TTS for all four cases we can see that again they do not vary much from each other which means that all methods/algorithms provide results that are very close to the optimum case. However, even though the results for the genetic algorithm are better than the ones for the no-control case, they are still a bit worse compared to the first optimisation method. This might be due to the randomness of the Matlab solver and the discrete set we were given, which results in some jumps between integer values along the simulation cycle that can provide in the end a slightly different result. We tried to change the initial population matrix of the genetic algorithm to several values from

our discrete set and the results actually never diverge much from the ones from the first optimisation algorithm. This can show how close to optimal our results from the first optimisation algorithm are.

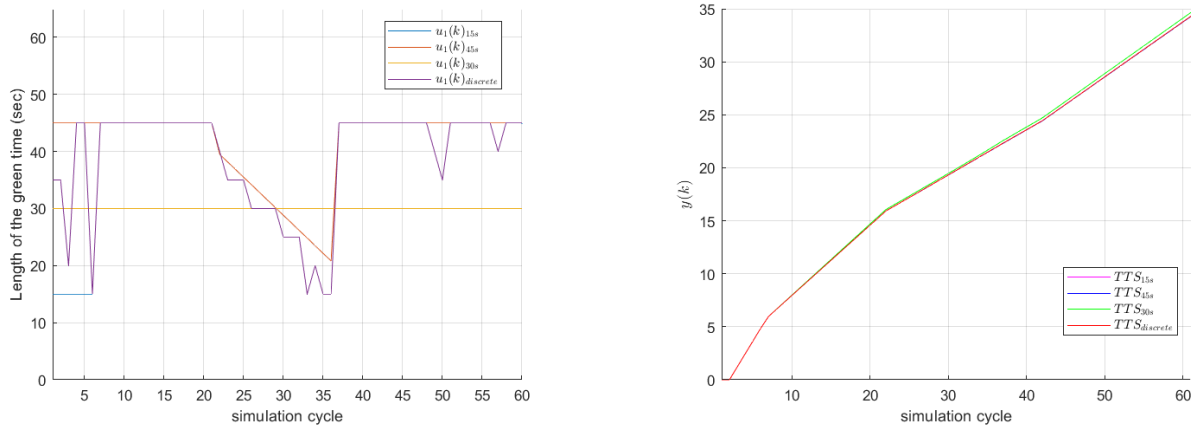

Fig. 10. Plots of inputs $u_1(k)$ (left) and objective function value (right) for optimised problem with starting points $x_0 = 15s$ and $x_0 = 45s$, for no control case and for genetic algorithm through one hour

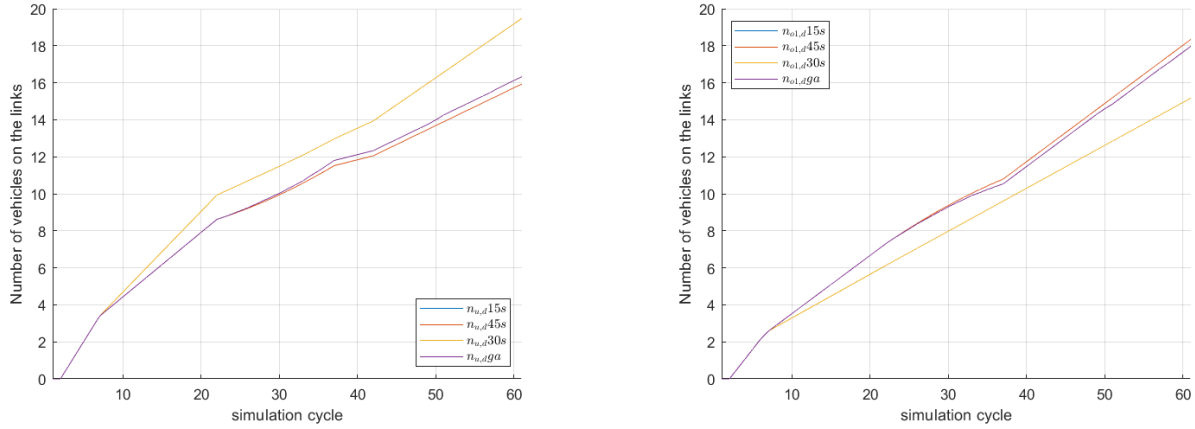

Fig. 11. Plots of states $n_{u,d}(k)$ (left) and $n_{o_1,d}(k)$ (right) for optimised problem with starting points $x_0 = 15s$ and $x_0 = 45s$, for no control case and for genetic algorithm through one hour

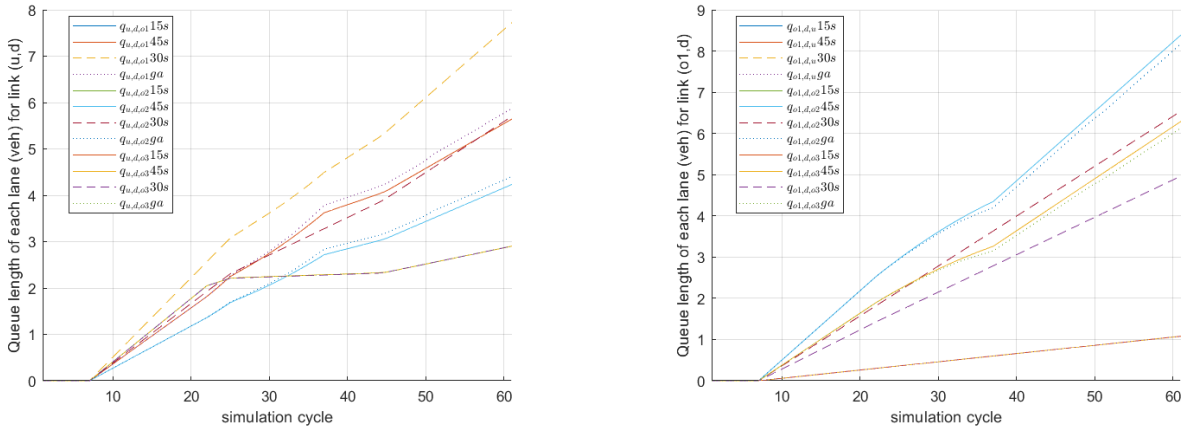

Fig. 12. Plots of states $q_{u,d,i}(k)$ for $i \in O_{u,d}$ (left) and $q_{o_1,d,j}(k)$ for $j \in O_{o_1,d}$(right) for optimised problem with starting points $x_0 = 15s$ and $x_0 = 45s$, for no control case and for genetic algorithm through one hour

REFERENCES

[1] Optimization for Systems and Control (SC42056). (2020, October 17). Retrieved from https://www.dcsc.tudelft.nl/ bdeschutter/osc

[2] LinearInequalityConstraintExample. (2022, October 18). Retrieved from https://www.mathworks.com/help/optim/ug/fmincon.html

[3] Choosing the Algorithm - MATLAB Simulink. (2022, October 18). Retrieved from https://www.mathworks.com/help/optim/ug/choosing-the-algorithm.htmlbsbwxm7

APPENDIX

A. *Problem formulation from Section II A.*

```
clear all
close all

%% parameters
global func;
% parameters for link (u,d)
c = 60/3600;
N_u_d = 3;
v_u_d = 60;
l_veh = 7/1000;
l_u_d = 4;
beta_u_d_o1 = 0.4;
beta_u_d_o2 = 0.3;
beta_u_d_o3 = 0.3;
mu_u_d = 1800;

% parameters for link (o1,d)
N_o1_d = 3;
v_o1_d = 50;
l_o1_d = 4;
beta_o1_d_u = 0.3;
beta_o1_d_o2 = 0.4;
beta_o1_d_o3 = 0.3;
mu_o1_d = 1700;
```

```
E_1 = 8+6;
E_2 = 7+0;
E_3 = 9+3;


C_u_d = (N_u_d*l_u_d)/l_veh;
C_o1_d = (N_o1_d*l_o1_d)/l_veh;

%------------ alpha enter (u,d) -------------%
alpha_enter_u_d(1:21) = 2300 + 10*E_1;
alpha_enter_u_d(22:41) = 1800 + 10*E_2;
alpha_enter_u_d(42:61) = 2100 + 10*E_3;

alpha_enter_o1_d(1:61) = 1800+10*E_1;

%------------ Cd,u-------------%
C_d_u(1:41) = 20-E_3/2;
C_d_u(42:61) = 20+E_3/2;

%------------ Cd,o2-------------%
C_d_o2(1:21) = 10+E_2/2;
C_d_o2(22:36) = 10+E_2/2-2*(21:35)/(E_1+1);
C_d_o2(37:46) = 20+E_2/2;
C_d_o2(47:61) = 20+E_3/2+2*(46:60)/(E_1+1);

%------------ Cd,o1-------------%
C_d_o1(1:61)=C_d_o2(1:61)-2*(0:60)/(E_1+1);

%------------ Cd,o3-------------%
C_d_o3(1:21) = 10-E_3/2;
C_d_o3(22:61) = 10+E_3/2;

for k = 1:61
    if C_d_o3(k) < 0
        C_d_o3(k) = 0;
    end

    if C_d_o2(k) < 0
        C_d_o2(k) = 0;
    end

    if C_d_o1(k) < 0
        C_d_o1(k) = 0;
    end

    if C_d_u(k) < 0
        C_d_u(k) = 0;
    end
end

%% question 3
lb = ones(1,61)*15/3600;
ub = ones(1,61)*45/3600;

rng default % For reproducibility
```

```
options = optimoptions('fmincon','Algorithm','sqp','MaxFunctionEvaluations',8e4,'
    TolFun',5e-4, 'ConstraintTolerance', 1e-4);
[x1,fval1,exitflag1,output1] = fmincon(@(x1)myfun(x1,C_u_d,C_o1_d,C_d_u,C_d_o3,C_d_o2
    ,C_d_o1,alpha_enter_o1_d,alpha_enter_u_d),lb,[],[],[],[],lb,ub,[],options)

figure()
hold on
grid on
plot(x1(1:61)*3600);
plot(x1(1:61)*3600);
plot(ones(1,61)*c*3600);
plot(ones(1,61)*c*3600);
plot(c*3600-x1(1:61)*3600);
plot(c*3600-x1(1:61)*3600);
xlim([1 60]);
ylim([0 65]);
legend('$g_{u,d,o1}$','$g_{u,d,o2}$', '$g_{u,d,o3}$', '$g_{o1,d,u}$','$g_{o1,d,o2}$',
    '$g_{o1,d,o3}$','interpreter','latex','location',"best")
ylabel('Length of the green time (sec)');
xlabel('simulation cycle');
title('Starting point with $g_{u,d,i(k)}=15s \forall k$, for $i\in O_{u,d}$','
    interpreter','latex')
hold off;

figure()
plot(func,"Color",'m')
xlim([1 61]);
ylabel('$y(k)$','interpreter','latex')
xlabel('simulation cycle');
title('Starting point with $g_{u,d,i(k)}=15s \forall k$, for $i\in O_{u,d}$','
    interpreter','latex')

global n_u_d n_o1_d q_u_d_o1 q_u_d_o2 q_u_d_o3 q_o1_d_u q_o1_d_o2 q_o1_d_o3;

figure();
hold on
grid on
plot(n_u_d(1:end)*c);
plot(n_o1_d(1:end)*c);
xlim([1 61]);
legend('$n_{u,d}$', '$n_{o1,d}$','interpreter','latex','location',"best")
ylabel('Number of vehicles on the links');
xlabel('simulation cycle');
title('Starting point with $g_{u,d,i(k)}=15s \forall k$, for $i\in O_{u,d}$','
    interpreter','latex')
hold off;

figure();
hold on
grid on
plot(q_u_d_o1(1:end));
plot(q_u_d_o2(1:end));
plot(q_u_d_o3(1:end));
plot(q_o1_d_u(1:end));
plot(q_o1_d_o2(1:end));
plot(q_o1_d_o3(1:end));
```

```
xlim([1 61]);
legend('$q_{u,d,o1}$', '$q_{u,d,o2}$', '$q_{u,d,o3}$', '$q_{o1,d,u}$', '$q_{o1,d,o2}$
    ', '$q_{o1,d,o3}$','interpreter','latex','location',"best")
ylabel('Queue length of each lane (veh)');
xlabel('simulation cycle');
title('Starting point with $g_{u,d,i(k)}=15s \forall k$, for $i\in O_{u,d}$','
    interpreter','latex')
hold off;

rng default % For reproducibility
[x2,fval2,exitflag2,output2] = fmincon(@(x2)myfun(x2,C_u_d,C_o1_d,C_d_u,C_d_o3,C_d_o2
    ,C_d_o1,alpha_enter_o1_d,alpha_enter_u_d),ub,[],[],[],[],lb,ub,[],options)

figure()
hold on
grid on
plot(x2(1:61)*3600);
plot(x2(1:61)*3600);
plot(ones(1,61)*c*3600);
plot(ones(1,61)*c*3600);
plot(c*3600-x2(1:61)*3600);
plot(c*3600-x2(1:61)*3600);
xlim([1 60]);
ylim([0 65]);
legend('$g_{u,d,o1}$','$g_{u,d,o2}$', '$g_{u,d,o3}$', '$g_{o1,d,u}$','$g_{o1,d,o2}$',
    '$g_{o1,d,o3}$','interpreter','latex','location',"best")
ylabel('Length of the green time (sec)');
xlabel('simulation cycle');
title('Starting point with $g_{u,d,i(k)}=45s \forall k$, for $i\in O_{u,d}$','
    interpreter','latex')
hold off;

figure()
plot(func,"Color",'m')
xlim([1 61]);
ylabel('$y(k)$','interpreter','latex')
xlabel('simulation cycle')
title('Starting point with $g_{u,d,i(k)}=45s \forall k$, for $i\in O_{u,d}$','
    interpreter','latex')

global n_u_d n_o1_d q_u_d_o1 q_u_d_o2 q_u_d_o3 q_o1_d_u q_o1_d_o2 q_o1_d_o3;

figure();
hold on
grid on
plot(n_u_d(1:end)*c);
plot(n_o1_d(1:end)*c);
xlim([1 61]);
legend('$n_{u,d}$', '$n_{o1,d}$','interpreter','latex','location',"best")
ylabel('Number of vehicles on the links');
xlabel('simulation cycle');
title('Starting point with $g_{u,d,i(k)}=45s \forall k$, for $i\in O_{u,d}$','
    interpreter','latex')
hold off;

figure();
```

```
hold on
grid on
plot(q_u_d_o1(1:end));
plot(q_u_d_o2(1:end));
plot(q_u_d_o3(1:end));
plot(q_o1_d_u(1:end));
plot(q_o1_d_o2(1:end));
plot(q_o1_d_o3(1:end));
xlim([1 61]);
legend('$q_{u,d,o1}$', '$q_{u,d,o2}$', '$q_{u,d,o3}$', '$q_{o1,d,u}$', '$q_{o1,d,o2}$
    ', '$q_{o1,d,o3}$','interpreter','latex','location',"best")
ylabel('Queue length of each lane (veh)');
xlabel('simulation cycle');
title('Starting point with $g_{u,d,i(k)}=45s \forall k$, for $i\in O_{u,d}$','
    interpreter','latex')
hold off;

%% question 4
global nocont;
rng default % For reproducibility
[x3,fval3,exitflag3,output3] = fmincon(@(x3)nocontrol(x3,C_u_d,C_o1_d,C_d_u,C_d_o3,
    C_d_o2,C_d_o1,alpha_enter_o1_d,alpha_enter_u_d),ones(1,61)*30/3600,[],[],[],[],lb,
    ub,[],options)

figure()
hold on
grid on
plot(x3(1:61)*3600);
plot(x3(1:61)*3600);
plot(ones(1,61)*c*3600);
plot(ones(1,61)*c*3600);
plot(c*3600-x3(1:61)*3600);
plot(c*3600-x3(1:61)*3600);
xlim([1 60]);
ylim([0 65]);
legend('$g_{u,d,o1}$','$g_{u,d,o2}$', '$g_{u,d,o3}$', '$g_{o1,d,u}$','$g_{o1,d,o2}$',
    '$g_{o1,d,o3}$','interpreter','latex','location',"best")
ylabel('Length of the green time (sec)');
xlabel('simulation cycle');
title('No-Control case ($g_{u,d,i(k)}=g_{o_1,d,j(k)}=30s \forall k$, for $i\in O_{u,d
    },j \in O_{o_1,d}$)','interpreter','latex')
hold off;

figure()
plot(nocont,"Color",'m')
xlim([1 61]);
title('No-Control case ($g_{u,d,i(k)}=g_{o_1,d,j(k)}=30s \forall k$, for $i\in O_{u,d
    },j \in O_{o_1,d}$)','interpreter','latex')
ylabel('$y(k)$','interpreter','latex')
xlabel('simulation cycle');

global n_u_d n_o1_d q_u_d_o1 q_u_d_o2 q_u_d_o3 q_o1_d_u q_o1_d_o2 q_o1_d_o3;

figure();
hold on
grid on
```

```
plot(n_u_d(1:end)*c);
plot(n_o1_d(1:end)*c);
xlim([1 61]);
legend('$n_{u,d}$', '$n_{o1,d}$','interpreter','latex','location',"best")
ylabel('Number of vehicles on the links');
xlabel('simulation cycle');
title('No-Control case ($g_{u,d,i(k)}=g_{o_1,d,j(k)}=30s \forall k$, for $i\in O_{u,d
    },j \in O_{o_1,d}$)','interpreter','latex')
hold off;

figure();
hold on
grid on
plot(q_u_d_o1(1:end));
plot(q_u_d_o2(1:end));
plot(q_u_d_o3(1:end));
plot(q_o1_d_u(1:end));
plot(q_o1_d_o2(1:end));
plot(q_o1_d_o3(1:end));
xlim([1 61]);
legend('$q_{u,d,o1}$', '$q_{u,d,o2}$', '$q_{u,d,o3}$', '$q_{o1,d,u}$', '$q_{o1,d,o2}$
    ', '$q_{o1,d,o3}$','interpreter','latex','location',"best")
ylabel('Queue length of each lane (veh)');
xlabel('simulation cycle');
title('No-Control case ($g_{u,d,i(k)}=g_{o_1,d,j(k)}=30s \forall k$, for $i\in O_{u,d
    },j \in O_{o_1,d}$)','interpreter','latex')
hold off;

%% question 5
lb=3*ones(1,61);
ub=9*ones(1,61);
opts = optimoptions('ga','ConstraintTolerance',1e-6, 'MaxStallGenerations',1000,'
    MaxGenerations',1000, 'PlotFcn', @gaplotbestf);
rng default % For reproducibility
[x4,fval4,eflag4,outpt4] = ga(@(x4)integer_sol(x4,C_u_d,C_o1_d,C_d_u,C_d_o3,C_d_o2,
    C_d_o1,alpha_enter_o1_d,alpha_enter_u_d),61,[],[],[],[],lb,ub,[],1:61,opts)

global integ n_u_d n_o1_d q_u_d_o1 q_u_d_o2 q_u_d_o3 q_o1_d_u q_o1_d_o2 q_o1_d_o3;

figure()
hold on
grid on
plot(x4(1:61)*5);
plot(x4(1:61)*5);
plot(ones(1,61)*c*3600);
plot(ones(1,61)*c*3600);
plot(c*3600-x4(1:61)*5);
plot(c*3600-x4(1:61)*5);
xlim([1 60]);
ylim([0 65]);
legend('$g_{u,d,o1}$','$g_{u,d,o2}$', '$g_{u,d,o3}$', '$g_{o1,d,u}$','$g_{o1,d,o2}$',
    '$g_{o1,d,o3}$','interpreter','latex','location',"best")
ylabel('Length of the green time (sec)');
xlabel('simulation cycle');
title('Genetic algorithm')
hold off;
```

```
figure()
plot(integ,"Color",'m')
xlim([1 61]);
title('Genetic algorithm')
ylabel('$y(k)$','interpreter','latex')
xlabel('simulation cycle');

figure();
hold on
grid on
plot(n_u_d(1:end)*c);
plot(n_o1_d(1:end)*c);
xlim([0 61]);
legend('$n_{u,d}$', '$n_{o1,d}$','interpreter','latex','location',"best")
ylabel('Number of vehicles on the links');
xlabel('simulation cycle');
title('Genetic algorithm')
hold off;

figure();
hold on
grid on
plot(q_u_d_o1(1:end));
plot(q_u_d_o2(1:end));
plot(q_u_d_o3(1:end));
plot(q_o1_d_u(1:end));
plot(q_o1_d_o2(1:end));
plot(q_o1_d_o3(1:end));
xlim([1 61]);
legend('$q_{u,d,o1}$', '$q_{u,d,o2}$', '$q_{u,d,o3}$', '$q_{o1,d,u}$', '$q_{o1,d,o2}$
   ', '$q_{o1,d,o3}$','interpreter','latex','location',"best")
ylabel('Queue length of each lane (veh)');
xlabel('simulation cycle');
title('Genetic algorithm')
hold off;
%% functions used

function i = integer_sol(x,C_u_d,C_o1_d,C_d_u,C_d_o3,C_d_o2,C_d_o1,alpha_enter_o1_d,
   alpha_enter_u_d)

    global integ n_u_d n_o1_d q_u_d_o1 q_u_d_o2 q_u_d_o3 q_o1_d_u q_o1_d_o2 q_o1_d_o3
   ;

    % parameters for link (u,d)
    c = 60/3600;
    N_u_d = 3;
    v_u_d = 60;
    l_veh = 7/1000;
    beta_u_d_o1 = 0.4;
    beta_u_d_o2 = 0.3;
    beta_u_d_o3 = 0.3;
    mu_u_d = 1800;

    % parameters for link (o1,d)
    N_o1_d = 3;
```

```
v_o1_d = 50;
beta_o1_d_u = 0.3;
beta_o1_d_o2 = 0.4;
beta_o1_d_o3 = 0.3;
mu_o1_d = 1700;

x = x*5/3600; % for values between 15s and 45s

for k = 1:60

    n_u_d(k==1) = 0;
    n_o1_d(k==1) = 0;

    q_u_d_o1(k==1) = 0;
    q_u_d_o2(k==1) = 0;
    q_u_d_o3(k==1) = 0;

    q_o1_d_u(k==1) = 0;
    q_o1_d_o2(k==1) = 0;
    q_o1_d_o3(k==1) = 0;

    alpha_enter_u_d(k==1) = 0;
    alpha_enter_o1_d(k==1) = 0;

    integ(1) = 0;

    tau_u_d(k) = floor((C_u_d-q_u_d_o1(k)-q_u_d_o2(k)-q_u_d_o3(k))*l_veh/(N_u_d*
v_u_d*c));
    tau_o1_d(k) = floor((C_o1_d-q_o1_d_u(k)-q_o1_d_o2(k)-q_o1_d_o3(k))*l_veh/(
N_o1_d*v_o1_d*c));

    gamma_u_d(k) = rem((C_u_d-q_u_d_o1(k)-q_u_d_o2(k)-q_u_d_o3(k))*l_veh, N_u_d*
v_u_d*c)/(N_u_d*v_u_d*c);
    gamma_o1_d(k) = rem((C_o1_d-q_o1_d_u(k)-q_o1_d_o2(k)-q_o1_d_o3(k))*l_veh,
N_o1_d*v_o1_d*c)/(N_o1_d*v_o1_d*c);

    if k-tau_u_d(k)<=0
        alpha_arrive_u_d(k) = 0;
    else
        if k-tau_u_d(k)-1<=1
           alpha_arrive_u_d(k) = 0;
        elseif  k-tau_u_d(k)==2
           alpha_arrive_u_d(k) = (1-gamma_u_d(k))*alpha_enter_u_d(2);
        else
           alpha_arrive_u_d(k) = (1-gamma_u_d(k))*alpha_enter_u_d(k-tau_u_d(k))+
gamma_u_d(k)*alpha_enter_u_d(k-tau_u_d(k)-1);
        end
    end

    if k-tau_o1_d(k)<=0
        alpha_arrive_o1_d(k) = 0;
    else
        if k-tau_o1_d(k)-1<1
           alpha_arrive_o1_d(k) = 0;
        elseif  k-tau_o1_d(k)==2
           alpha_arrive_o1_d(k) = (1-gamma_o1_d(k))*alpha_enter_o1_d(2);
```

```
            else
               alpha_arrive_o1_d(k) = (1-gamma_o1_d(k))*alpha_enter_o1_d(k-tau_o1_d(k
    ))+gamma_o1_d(k)*alpha_enter_o1_d(k-tau_o1_d(k)-1);
            end
        end

        alpha_leave_u_d_o1(k) = min([beta_u_d_o1*mu_u_d*x(k)/c, q_u_d_o1(k)/c +
    beta_u_d_o1*alpha_arrive_u_d(k), C_d_o1(k)/c]);
        alpha_leave_u_d_o2(k) = min([beta_u_d_o2*mu_u_d*x(k)/c, q_u_d_o2(k)/c +
    beta_u_d_o2*alpha_arrive_u_d(k), C_d_o2(k)/c]);
        alpha_leave_u_d_o3(k) = min([beta_u_d_o3*mu_u_d, q_u_d_o3(k)/c + beta_u_d_o3*
    alpha_arrive_u_d(k), C_d_o3(k)/c]);

        alpha_leave_o1_d_u(k) = min([beta_o1_d_u*mu_o1_d, q_o1_d_u(k)/c + beta_o1_d_u
    *alpha_arrive_o1_d(k), C_d_u(k)/c]);
        alpha_leave_o1_d_o2(k) = min([beta_o1_d_o2*mu_o1_d*(c-x(k))/c, q_o1_d_o2(k)/c
     + beta_o1_d_o2*alpha_arrive_o1_d(k), C_d_o2(k)/c]);
        alpha_leave_o1_d_o3(k) = min([beta_o1_d_o3*mu_o1_d*(c-x(k))/c, q_o1_d_o3(k)/c
     + beta_o1_d_o3*alpha_arrive_o1_d(k), C_d_o3(k)/c]);

        n_u_d(k+1)=n_u_d(k)+(alpha_enter_u_d(k)-(alpha_leave_u_d_o1(k)+
    alpha_leave_u_d_o2(k)+alpha_leave_u_d_o3(k)))*c;
        n_o1_d(k+1)=n_o1_d(k)+(alpha_enter_o1_d(k)-(alpha_leave_o1_d_u(k)+
    alpha_leave_o1_d_o2(k)+alpha_leave_o1_d_o3(k)))*c;

        q_u_d_o1(k+1) = q_u_d_o1(k)+(beta_u_d_o1*alpha_arrive_u_d(k)-
    alpha_leave_u_d_o1(k))*c;
        q_u_d_o2(k+1) = q_u_d_o2(k)+(beta_u_d_o2*alpha_arrive_u_d(k)-
    alpha_leave_u_d_o2(k))*c;
        q_u_d_o3(k+1) = q_u_d_o3(k)+(beta_u_d_o3*alpha_arrive_u_d(k)-
    alpha_leave_u_d_o3(k))*c;

        q_o1_d_u(k+1) = q_o1_d_u(k)+(beta_o1_d_u*alpha_arrive_o1_d(k)-
    alpha_leave_o1_d_u(k))*c;
        q_o1_d_o2(k+1) = q_o1_d_o2(k)+(beta_o1_d_o2*alpha_arrive_o1_d(k)-
    alpha_leave_o1_d_o2(k))*c;
        q_o1_d_o3(k+1) = q_o1_d_o3(k)+(beta_o1_d_o3*alpha_arrive_o1_d(k)-
    alpha_leave_o1_d_o3(k))*c;

        integ(k+1) = (n_u_d(k+1)+n_o1_d(k+1))*c;
    end

    i = sum(integ);
end

function m = nocontrol(x,C_u_d,C_o1_d,C_d_u,C_d_o3,C_d_o2,C_d_o1,alpha_enter_o1_d,
    alpha_enter_u_d)

    global nocont n_u_d n_o1_d q_u_d_o1 q_u_d_o2 q_u_d_o3 q_o1_d_u q_o1_d_o2
    q_o1_d_o3;


    x = ones(1,61)*30/3600;

    % parameters for link (u,d)
    c = 60/3600;
```

```
N_u_d = 3;
v_u_d = 60;
l_veh = 7/1000;
beta_u_d_o1 = 0.4;
beta_u_d_o2 = 0.3;
beta_u_d_o3 = 0.3;
mu_u_d = 1800;

% parameters for link (o1,d)
N_o1_d = 3;
v_o1_d = 50;
beta_o1_d_u = 0.3;
beta_o1_d_o2 = 0.4;
beta_o1_d_o3 = 0.3;
mu_o1_d = 1700;

for k = 1:60

    n_u_d(k==1) = 0;
    n_o1_d(k==1) = 0;

    q_u_d_o1(k==1) = 0;
    q_u_d_o2(k==1) = 0;
    q_u_d_o3(k==1) = 0;

    q_o1_d_u(k==1) = 0;
    q_o1_d_o2(k==1) = 0;
    q_o1_d_o3(k==1) = 0;

    alpha_enter_u_d(k==1) = 0;
    alpha_enter_o1_d(k==1) = 0;

    nocont(1) = 0;


    tau_u_d(k) = floor((C_u_d-q_u_d_o1(k)-q_u_d_o2(k)-q_u_d_o3(k))*l_veh/(N_u_d*
v_u_d*c));
    tau_o1_d(k) = floor((C_o1_d-q_o1_d_u(k)-q_o1_d_o2(k)-q_o1_d_o3(k))*l_veh/(
N_o1_d*v_o1_d*c));

    gamma_u_d(k) = rem((C_u_d-q_u_d_o1(k)-q_u_d_o2(k)-q_u_d_o3(k))*l_veh, N_u_d*
v_u_d*c)/(N_u_d*v_u_d*c);
    gamma_o1_d(k) = rem((C_o1_d-q_o1_d_u(k)-q_o1_d_o2(k)-q_o1_d_o3(k))*l_veh,
N_o1_d*v_o1_d*c)/(N_o1_d*v_o1_d*c);

    if k-tau_u_d(k)<=0
        alpha_arrive_u_d(k) = 0;
    else
        if k-tau_u_d(k)-1<=1
           alpha_arrive_u_d(k) = 0;
        elseif  k-tau_u_d(k)==2
           alpha_arrive_u_d(k) = (1-gamma_u_d(k))*alpha_enter_u_d(2);
        else
           alpha_arrive_u_d(k) = (1-gamma_u_d(k))*alpha_enter_u_d(k-tau_u_d(k))+
gamma_u_d(k)*alpha_enter_u_d(k-tau_u_d(k)-1);
        end
```

```
        end

        if k-tau_o1_d(k)<=0
            alpha_arrive_o1_d(k) = 0;
        else
            if k-tau_o1_d(k)-1<1
               alpha_arrive_o1_d(k) = 0;
            elseif  k-tau_o1_d(k)==2
               alpha_arrive_o1_d(k) = (1-gamma_o1_d(k))*alpha_enter_o1_d(2);
            else
               alpha_arrive_o1_d(k) = (1-gamma_o1_d(k))*alpha_enter_o1_d(k-tau_o1_d(k
   ))+gamma_o1_d(k)*alpha_enter_o1_d(k-tau_o1_d(k)-1);
            end
        end

        alpha_leave_u_d_o1(k) = min([beta_u_d_o1*mu_u_d*x(k)/c, q_u_d_o1(k)/c +
   beta_u_d_o1*alpha_arrive_u_d(k), C_d_o1(k)/c]);
        alpha_leave_u_d_o2(k) = min([beta_u_d_o2*mu_u_d*x(k)/c, q_u_d_o2(k)/c +
   beta_u_d_o2*alpha_arrive_u_d(k), C_d_o2(k)/c]);
        alpha_leave_u_d_o3(k) = min([beta_u_d_o3*mu_u_d, q_u_d_o3(k)/c + beta_u_d_o3*
   alpha_arrive_u_d(k), C_d_o3(k)/c]);

        alpha_leave_o1_d_u(k) = min([beta_o1_d_u*mu_o1_d, q_o1_d_u(k)/c + beta_o1_d_u
   *alpha_arrive_o1_d(k), C_d_u(k)/c]);
        alpha_leave_o1_d_o2(k) = min([beta_o1_d_o2*mu_o1_d*(c-x(k))/c, q_o1_d_o2(k)/c
    + beta_o1_d_o2*alpha_arrive_o1_d(k), C_d_o2(k)/c]);
        alpha_leave_o1_d_o3(k) = min([beta_o1_d_o3*mu_o1_d*(c-x(k))/c, q_o1_d_o3(k)/c
    + beta_o1_d_o3*alpha_arrive_o1_d(k), C_d_o3(k)/c]);

        n_u_d(k+1)=n_u_d(k)+(alpha_enter_u_d(k)-(alpha_leave_u_d_o1(k)+
   alpha_leave_u_d_o2(k)+alpha_leave_u_d_o3(k)))*c;
        n_o1_d(k+1)=n_o1_d(k)+(alpha_enter_o1_d(k)-(alpha_leave_o1_d_u(k)+
   alpha_leave_o1_d_o2(k)+alpha_leave_o1_d_o3(k)))*c;

        q_u_d_o1(k+1) = q_u_d_o1(k)+(beta_u_d_o1*alpha_arrive_u_d(k)-
   alpha_leave_u_d_o1(k))*c;
        q_u_d_o2(k+1) = q_u_d_o2(k)+(beta_u_d_o2*alpha_arrive_u_d(k)-
   alpha_leave_u_d_o2(k))*c;
        q_u_d_o3(k+1) = q_u_d_o3(k)+(beta_u_d_o3*alpha_arrive_u_d(k)-
   alpha_leave_u_d_o3(k))*c;

        q_o1_d_u(k+1) = q_o1_d_u(k)+(beta_o1_d_u*alpha_arrive_o1_d(k)-
   alpha_leave_o1_d_u(k))*c;
        q_o1_d_o2(k+1) = q_o1_d_o2(k)+(beta_o1_d_o2*alpha_arrive_o1_d(k)-
   alpha_leave_o1_d_o2(k))*c;
        q_o1_d_o3(k+1) = q_o1_d_o3(k)+(beta_o1_d_o3*alpha_arrive_o1_d(k)-
   alpha_leave_o1_d_o3(k))*c;

        nocont(k+1) = (n_u_d(k+1)+n_o1_d(k+1))*c;
    end

    m = sum(nocont);
end

function f = myfun(x,C_u_d,C_o1_d,C_d_u,C_d_o3,C_d_o2,C_d_o1,alpha_enter_o1_d,
   alpha_enter_u_d)
```

```
global func n_u_d n_o1_d q_u_d_o1 q_u_d_o2 q_u_d_o3 q_o1_d_u q_o1_d_o2 q_o1_d_o3;

% parameters for link (u,d)
c = 60/3600;
N_u_d = 3;
v_u_d = 60;
l_veh = 7/1000;
beta_u_d_o1 = 0.4;
beta_u_d_o2 = 0.3;
beta_u_d_o3 = 0.3;
mu_u_d = 1800;

% parameters for link (o1,d)
N_o1_d = 3;
v_o1_d = 50;
beta_o1_d_u = 0.3;
beta_o1_d_o2 = 0.4;
beta_o1_d_o3 = 0.3;
mu_o1_d = 1700;

for k = 1:60

    n_u_d(k==1) = 0;
    n_o1_d(k==1) = 0;

    q_u_d_o1(k==1) = 0;
    q_u_d_o2(k==1) = 0;
    q_u_d_o3(k==1) = 0;

    q_o1_d_u(k==1) = 0;
    q_o1_d_o2(k==1) = 0;
    q_o1_d_o3(k==1) = 0;
    alpha_enter_u_d(k==1) = 0;
    alpha_enter_o1_d(k==1) = 0;
    func(1) = 0;

    tau_u_d(k) = floor((C_u_d-q_u_d_o1(k)-q_u_d_o2(k)-q_u_d_o3(k))*l_veh/(N_u_d*
v_u_d*c));
    tau_o1_d(k) = floor((C_o1_d-q_o1_d_u(k)-q_o1_d_o2(k)-q_o1_d_o3(k))*l_veh/(
N_o1_d*v_o1_d*c));

    gamma_u_d(k) = rem((C_u_d-q_u_d_o1(k)-q_u_d_o2(k)-q_u_d_o3(k))*l_veh, N_u_d*
v_u_d*c)/(N_u_d*v_u_d*c);
    gamma_o1_d(k) = rem((C_o1_d-q_o1_d_u(k)-q_o1_d_o2(k)-q_o1_d_o3(k))*l_veh,
N_o1_d*v_o1_d*c)/(N_o1_d*v_o1_d*c);

    if k-tau_u_d(k)<=0
        alpha_arrive_u_d(k) = 0;
    else
        if k-tau_u_d(k)-1<=1
           alpha_arrive_u_d(k) = 0;
        elseif  k-tau_u_d(k)==2
           alpha_arrive_u_d(k) = (1-gamma_u_d(k))*alpha_enter_u_d(2);
        else
           alpha_arrive_u_d(k) = (1-gamma_u_d(k))*alpha_enter_u_d(k-tau_u_d(k))+
```

```
gamma_u_d(k)*alpha_enter_u_d(k-tau_u_d(k)-1);
        end
    end

    if k-tau_o1_d(k)<=0
        alpha_arrive_o1_d(k) = 0;
    else
        if k-tau_o1_d(k)-1<1
           alpha_arrive_o1_d(k) = 0;
        elseif  k-tau_o1_d(k)==2
           alpha_arrive_o1_d(k) = (1-gamma_o1_d(k))*alpha_enter_o1_d(2);
        else
           alpha_arrive_o1_d(k) = (1-gamma_o1_d(k))*alpha_enter_o1_d(k-tau_o1_d(k
))+gamma_o1_d(k)*alpha_enter_o1_d(k-tau_o1_d(k)-1);
        end
    end

    alpha_leave_u_d_o1(k) = min([beta_u_d_o1*mu_u_d*x(k)/c, q_u_d_o1(k)/c +
beta_u_d_o1*alpha_arrive_u_d(k), C_d_o1(k)/c]);
    alpha_leave_u_d_o2(k) = min([beta_u_d_o2*mu_u_d*x(k)/c, q_u_d_o2(k)/c +
beta_u_d_o2*alpha_arrive_u_d(k), C_d_o2(k)/c]);
    alpha_leave_u_d_o3(k) = min([beta_u_d_o3*mu_u_d, q_u_d_o3(k)/c + beta_u_d_o3*
alpha_arrive_u_d(k), C_d_o3(k)/c]);

    alpha_leave_o1_d_u(k) = min([beta_o1_d_u*mu_o1_d, q_o1_d_u(k)/c + beta_o1_d_u
*alpha_arrive_o1_d(k), C_d_u(k)/c]);
    alpha_leave_o1_d_o2(k) = min([beta_o1_d_o2*mu_o1_d*(c-x(k))/c, q_o1_d_o2(k)/c
 + beta_o1_d_o2*alpha_arrive_o1_d(k), C_d_o2(k)/c]);
    alpha_leave_o1_d_o3(k) = min([beta_o1_d_o3*mu_o1_d*(c-x(k))/c, q_o1_d_o3(k)/c
 + beta_o1_d_o3*alpha_arrive_o1_d(k), C_d_o3(k)/c]);

    n_u_d(k+1)=n_u_d(k)+(alpha_enter_u_d(k)-(alpha_leave_u_d_o1(k)+
alpha_leave_u_d_o2(k)+alpha_leave_u_d_o3(k)))*c;
    n_o1_d(k+1)=n_o1_d(k)+(alpha_enter_o1_d(k)-(alpha_leave_o1_d_u(k)+
alpha_leave_o1_d_o2(k)+alpha_leave_o1_d_o3(k)))*c;

    q_u_d_o1(k+1) = q_u_d_o1(k)+(beta_u_d_o1*alpha_arrive_u_d(k)-
alpha_leave_u_d_o1(k))*c;
    q_u_d_o2(k+1) = q_u_d_o2(k)+(beta_u_d_o2*alpha_arrive_u_d(k)-
alpha_leave_u_d_o2(k))*c;
    q_u_d_o3(k+1) = q_u_d_o3(k)+(beta_u_d_o3*alpha_arrive_u_d(k)-
alpha_leave_u_d_o3(k))*c;

    q_o1_d_u(k+1) = q_o1_d_u(k)+(beta_o1_d_u*alpha_arrive_o1_d(k)-
alpha_leave_o1_d_u(k))*c;
    q_o1_d_o2(k+1) = q_o1_d_o2(k)+(beta_o1_d_o2*alpha_arrive_o1_d(k)-
alpha_leave_o1_d_o2(k))*c;
    q_o1_d_o3(k+1) = q_o1_d_o3(k)+(beta_o1_d_o3*alpha_arrive_o1_d(k)-
alpha_leave_o1_d_o3(k))*c;


    %func(k+1) = -c^2*(alpha_leave_u_d_o1(k)+alpha_leave_u_d_o2(k)+
alpha_leave_u_d_o3(k)+alpha_leave_o1_d_u(k)+alpha_leave_o1_d_o2(k)+
alpha_leave_o1_d_o3(k));
    func(k+1) = (n_u_d(k+1)+n_o1_d(k+1))*c;
 end
```

```
    f = sum(func);
end
```

B. Problem formulation from Section II B.

```
clear all
close all

%% parameters
% parameters for link (u,d)
c = 60/3600;
N_u_d = 3;
v_u_d = 60;
l_veh = 7/1000;
l_u_d = 4;
beta_u_d_o1 = 0.4;
beta_u_d_o2 = 0.3;
beta_u_d_o3 = 0.3;
mu_u_d = 1800;

% parameters for link (o1,d)
N_o1_d = 3;
v_o1_d = 50;
l_o1_d = 4;
beta_o1_d_u = 0.3;
beta_o1_d_o2 = 0.4;
beta_o1_d_o3 = 0.3;
mu_o1_d = 1700;

E_1 = 8+6;
E_2 = 7+0;
E_3 = 9+3;

C_u_d = (N_u_d*l_u_d)/l_veh;
C_o1_d = (N_o1_d*l_o1_d)/l_veh;

%------------- alpha enter (u,d) -------------%
alpha_enter_u_d(1:21) = 2300 + 10*E_1;
alpha_enter_u_d(22:41) = 1800 + 10*E_2;
alpha_enter_u_d(42:61) = 2100 + 10*E_3;

alpha_enter_o1_d(1:61) = 1800+10*E_1;

%------------- Cd,u-------------%
C_d_u(1:41) = 20-E_3/2;
C_d_u(42:61) = 20+E_3/2;

%------------- Cd,o2-------------%
C_d_o2(1:21) = 10+E_2/2;
C_d_o2(22:36) = 10+E_2/2-2*(21:35)/(E_1+1);
C_d_o2(37:46) = 20+E_2/2;
C_d_o2(47:61) = 20+E_3/2+2*(46:60)/(E_1+1);

%------------- Cd,o1-------------%
C_d_o1(1:61)=C_d_o2(1:61)-2*(0:60)/(E_1+1);
```

```
%------------- Cd,o3-------------%
C_d_o3(1:21) = 10-E_3/2;
C_d_o3(22:61) = 10+E_3/2;

for k = 1:61
    if C_d_o3(k) < 0
        C_d_o3(k) = 0;
    end

    if C_d_o2(k) < 0
        C_d_o2(k) = 0;
    end

    if C_d_o1(k) < 0
        C_d_o1(k) = 0;
    end

    if C_d_u(k) < 0
        C_d_u(k) = 0;
    end
end

%% question 3
lb = 15/3600;
ub = 45/3600;

rng default % For reproducibility
options = optimoptions('fmincon','Algorithm','sqp','MaxFunctionEvaluations',8e4,'
   TolFun',5e-2);

for k = 1:60

    n_u_d_(1) = 0;
    n_o1_d_(1) = 0;

    q_u_d_o1(1) = 0;
    q_u_d_o2(1) = 0;
    q_u_d_o3(1) = 0;

    q_o1_d_u(1) = 0;
    q_o1_d_o2(1) = 0;
    q_o1_d_o3(1) = 0;

    iteration(1) = 0;
    alpha_enter_u_d(k==1) = 0;
    alpha_enter_o1_d(k==1) = 0;
    value(1) = 0;
    x1 = 15/3600;
    green_light(1) = x1;

    tau_u_d(k) = floor((C_u_d-q_u_d_o1(k)-q_u_d_o2(k)-q_u_d_o3(k))*l_veh/(N_u_d*v_u_d
   *c));
    tau_o1_d(k) = floor((C_o1_d-q_o1_d_u(k)-q_o1_d_o2(k)-q_o1_d_o3(k))*l_veh/(N_o1_d*
   v_o1_d*c));
```

```
 gamma_u_d(k) = rem((C_u_d-q_u_d_o1(k)-q_u_d_o2(k)-q_u_d_o3(k))*l_veh, N_u_d*v_u_d
*c)/(N_u_d*v_u_d*c);
 gamma_o1_d(k) = rem((C_o1_d-q_o1_d_u(k)-q_o1_d_o2(k)-q_o1_d_o3(k))*l_veh, N_o1_d*
v_o1_d*c)/(N_o1_d*v_o1_d*c);

 if k-tau_u_d(k)<=0
     alpha_arrive_u_d(k) = 0;
 else
     if k-tau_u_d(k)-1<=1
        alpha_arrive_u_d(k) = 0;
     elseif  k-tau_u_d(k)==2
        alpha_arrive_u_d(k) = (1-gamma_u_d(k))*alpha_enter_u_d(2);
     else
        alpha_arrive_u_d(k) = (1-gamma_u_d(k))*alpha_enter_u_d(k-tau_u_d(k))+
gamma_u_d(k)*alpha_enter_u_d(k-tau_u_d(k)-1);
     end
 end

 if k-tau_o1_d(k)<=0
     alpha_arrive_o1_d(k) = 0;
 else
     if k-tau_o1_d(k)-1<1
        alpha_arrive_o1_d(k) = 0;
     elseif  k-tau_o1_d(k)==2
        alpha_arrive_o1_d(k) = (1-gamma_o1_d(k))*alpha_enter_o1_d(2);
     else
        alpha_arrive_o1_d(k) = (1-gamma_o1_d(k))*alpha_enter_o1_d(k-tau_o1_d(k))+
gamma_o1_d(k)*alpha_enter_o1_d(k-tau_o1_d(k)-1);
     end
 end

 alpha_leave_u_d_o1 = @(x1)min([beta_u_d_o1*mu_u_d*x1/c, q_u_d_o1(k)/c +
beta_u_d_o1*alpha_arrive_u_d(k), C_d_o1(k)/c]);
 alpha_leave_u_d_o2 =@(x1)min([beta_u_d_o2*mu_u_d*x1/c, q_u_d_o2(k)/c +
beta_u_d_o2*alpha_arrive_u_d(k), C_d_o2(k)/c]);
 alpha_leave_u_d_o3 = min([beta_u_d_o3*mu_u_d, q_u_d_o3(k)/c + beta_u_d_o3*
alpha_arrive_u_d(k), C_d_o3(k)/c]);

 alpha_leave_o1_d_u = min([beta_o1_d_u*mu_o1_d, q_o1_d_u(k)/c + beta_o1_d_u*
alpha_arrive_o1_d(k), C_d_u(k)/c]);
 alpha_leave_o1_d_o2 = @(x1)min([beta_o1_d_o2*mu_o1_d*(c-x1)/c, q_o1_d_o2(k)/c +
beta_o1_d_o2*alpha_arrive_o1_d(k), C_d_o2(k)/c]);
 alpha_leave_o1_d_o3 = @(x1)min([beta_o1_d_o3*mu_o1_d*(c-x1)/c, q_o1_d_o3(k)/c +
beta_o1_d_o3*alpha_arrive_o1_d(k), C_d_o3(k)/c]);

 n_u_d=@(x1)(n_u_d_(k)+(alpha_enter_u_d(k)-(alpha_leave_u_d_o1(x1)+
alpha_leave_u_d_o2(x1)+alpha_leave_u_d_o3))*c);
 n_o1_d=@(x1)(n_o1_d_(k)+(alpha_enter_o1_d(k)-(alpha_leave_o1_d_u+
alpha_leave_o1_d_o2(x1)+alpha_leave_o1_d_o3(x1)))*c);

 func = @(x1)((n_u_d(x1)+n_o1_d(x1))*c);

 [x1,fval1,exitflag1,output1] = fmincon(func,15/3600,[],[],[],[],lb,ub,[],options)
;

 n_u_d_(k+1)=n_u_d(x1);
```

```
    n_o1_d_(k+1)=n_o1_d(x1);

    iteration(k+1)=(n_u_d_(k+1)+n_o1_d_(k+1))*c;

    q_u_d_o1(k+1) = q_u_d_o1(k)+(beta_u_d_o1*alpha_arrive_u_d(k)-alpha_leave_u_d_o1(
   x1))*c;
    q_u_d_o2(k+1) = q_u_d_o2(k)+(beta_u_d_o2*alpha_arrive_u_d(k)-alpha_leave_u_d_o2(
   x1))*c;
    q_u_d_o3(k+1) = q_u_d_o3(k)+(beta_u_d_o3*alpha_arrive_u_d(k)-alpha_leave_u_d_o3)*
   c;

    q_o1_d_u(k+1) = q_o1_d_u(k)+(beta_o1_d_u*alpha_arrive_o1_d(k)-alpha_leave_o1_d_u)
   *c;
    q_o1_d_o2(k+1) = q_o1_d_o2(k)+(beta_o1_d_o2*alpha_arrive_o1_d(k)-
   alpha_leave_o1_d_o2(x1))*c;
    q_o1_d_o3(k+1) = q_o1_d_o3(k)+(beta_o1_d_o3*alpha_arrive_o1_d(k)-
   alpha_leave_o1_d_o3(x1))*c;

    value(k+1) = fval1;
    green_light(k+1) = x1;
end

f = sum(value)

figure()
hold on
grid on
plot(green_light(1:61)*3600);
plot(green_light(1:61)*3600);
plot(ones(1,61)*c*3600);
plot(ones(1,61)*c*3600);
plot(c*3600-green_light(1:61)*3600);
plot(c*3600-green_light(1:61)*3600);
xlim([1 61]);
ylim([0 65]);
legend('$g_{u,d,o1}$','$g_{u,d,o2}$', '$g_{u,d,o3}$', '$g_{o1,d,u}$','$g_{o1,d,o2}$',
   '$g_{o1,d,o3}$','interpreter','latex','location',"best")
ylabel('Length of the green time (sec)');
xlabel('simulation cycle');
title('Starting point with $g_{u,d,i(k)}=15s \forall k$, for $i\in O_{u,d}$','
   interpreter','latex')
hold off;

figure()
plot(iteration(1:61),"Color",'m')
xlim([1 61]);
ylabel('$y(k)$','interpreter','latex')
xlabel('simulation cycle');
title('Starting point with $g_{u,d,i(k)}=15s \forall k$, for $i\in O_{u,d}$','
   interpreter','latex')

figure();
hold on
grid on
plot(n_u_d_(1:61)*c);
plot(n_o1_d_(1:61)*c);
```

```
xlim([1 61]);
legend('$n_{u,d}$', '$n_{o1,d}$','interpreter','latex','location',"best")
ylabel('Number of vehicles on the links');
xlabel('simulation cycle');
title('Starting point with $g_{u,d,i(k)}=15s \forall k$, for $i\in O_{u,d}$','
   interpreter','latex')
hold off;

figure();
hold on
grid on
plot(q_u_d_o1(1:61));
plot(q_u_d_o2(1:61));
plot(q_u_d_o3(1:61));
plot(q_o1_d_u(1:61));
plot(q_o1_d_o2(1:61));
plot(q_o1_d_o3(1:61));
xlim([1 61]);
legend('$q_{u,d,o1}$', '$q_{u,d,o2}$', '$q_{u,d,o3}$', '$q_{o1,d,u}$', '$q_{o1,d,o2}$
   ', '$q_{o1,d,o3}$','interpreter','latex','location',"best")
ylabel('Queue length of each lane (veh)');
xlabel('simulation cycle');
title('Starting point with $g_{u,d,i(k)}=15s \forall k$, for $i\in O_{u,d}$','
   interpreter','latex')
hold off;
```