Delft University of Technology

Optimisation for system and Control
SC42056

Assignment 2: NLP Assignment

Authors: Jiaxuan Zhang, Yiting Li
Group ID: (47)

October 27, 2020

Assumption

There are several global assumptions/declaration we made in all tasks.

1. For all links, we assume their capacity $C_{i,o}$ is shared by other links who has flow into this link in each cycle. For example, for $link(d, o_2)$ when calculating α^{leave}_{u,d,o_1} and calculating $\alpha^{leave}_{o_1,d,o_2}$, the third part of the equation (3) in the task document is specified by $C_{d,o_2}(k)/(2c)$.

2. Value of E_1, E_2, E_3 are: $E_1 = 9$, $E_2 = 13$, $E_3 = 5$

1 Task 1

1.1 Variables Definition

1.2 State Space Model

2 Task 2

3 Task 3

4 Task 4

4.1 Analysis

From Task 2, it is apparently that with the parameter in Task 2,when the initial state and the input is given, system's states and output at time k ($k >= 0$) is specified. So in Task 4, a recurrence function `TTS_calculate` is used as in the Task 3, to obtain the final objective value of the output.

4.2 Result

With MATLAB Code, it comes out that if the input is $g(k) = 30s, \forall k$, the TTS of this system is $7.56591 \times 10^5 (s)$.

The transform of states variant of number of vehicles on the link and the queue length of each lane under no-control situation is shown in Figure 1.

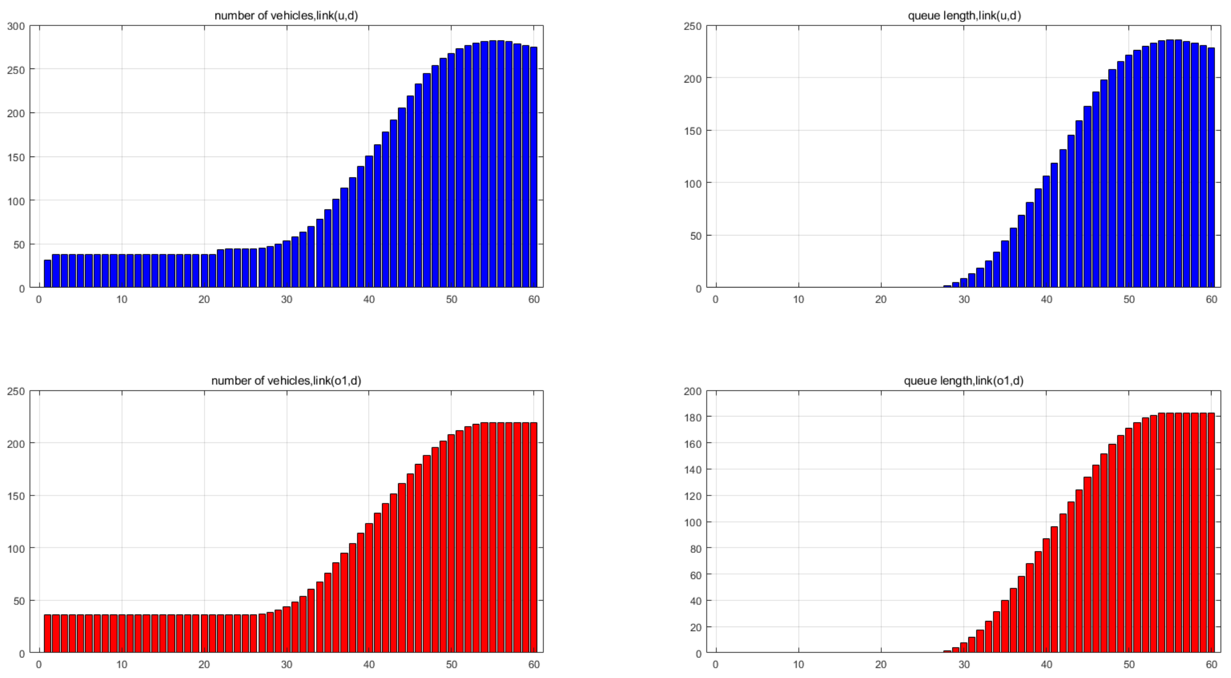


Figure 1: traffic situation no control

4.3 Compare

1. Optimal Result Compare

   The optimal result of objective function is shown in the following Table.

methods	SQP		Interior		Newton		Annealing		Perp+Golden	
starting	15	45	15	45	15	45	15	45	15	45
results	7.5199	7.5194	8.1550	7.5248	7.5224	7.5198	7.5194	7.5194	7.5194	7.5194
iter($\times$60)	44	72	80	78	31	27	654	790	100.2	68.2

Table 1: Table: Optimal Result

Figure 2 shows the the optimal traffic situations after using SQP optimization method form starting point $g(k) = 15s, \forall k$. Figure 3 shows the the optimal traffic situations after using SQP optimization method form starting point $g(k) = 45s, \forall k$. Figure 4 shows the optimal traffic situation after using simulate annealing algorithm from starting point $g(k) = 15s, \forall k$. Figure 5 shows the optimal traffic situation after using simulate annealing algorithm from starting point $g(k) = 45s, \forall k$.

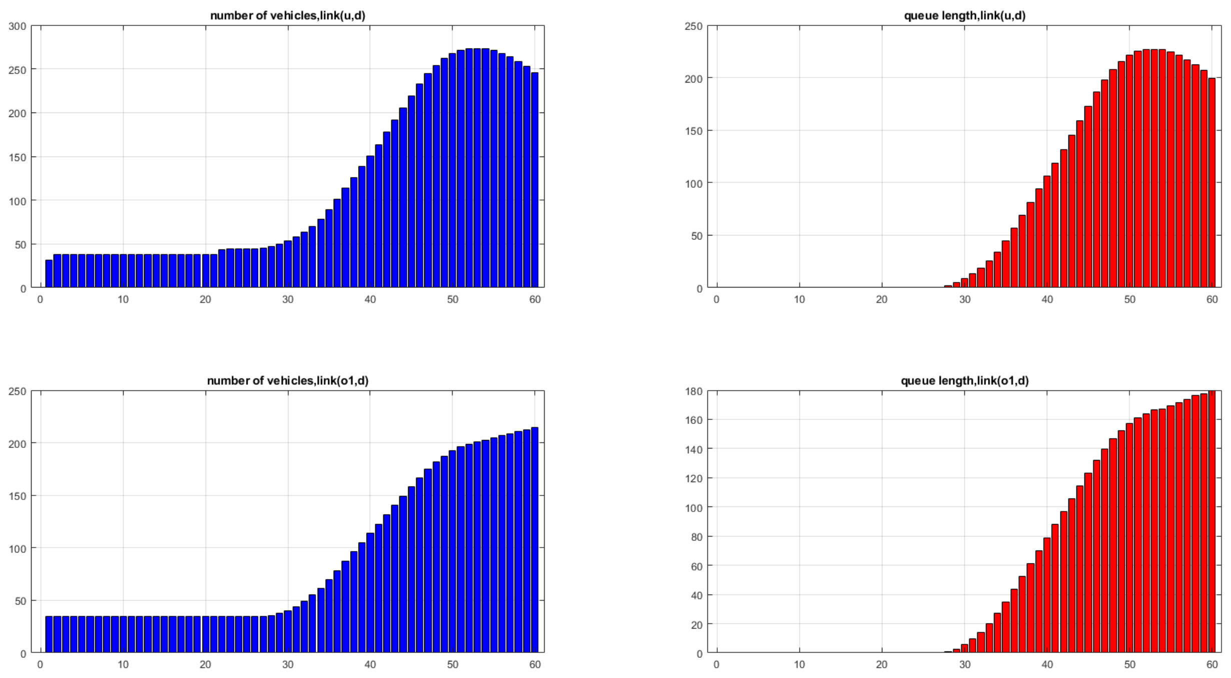


Figure 2: traffic situation SQP 15s

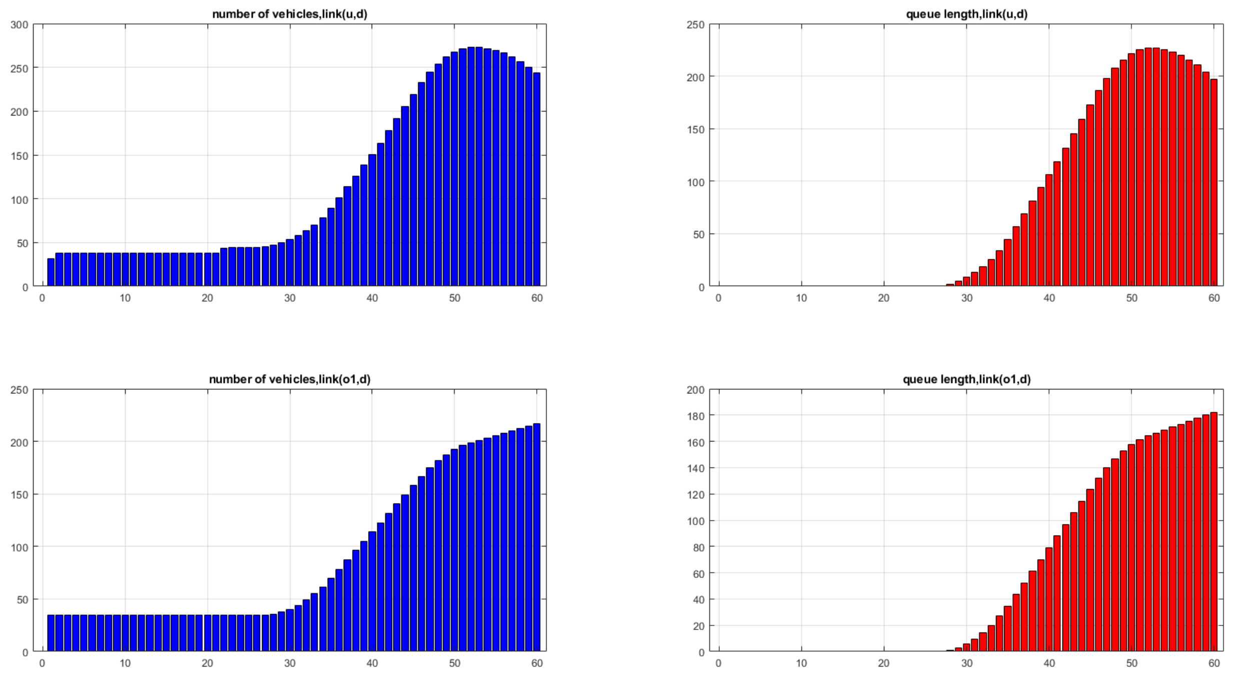


Figure 3: traffic situation SQP 45s

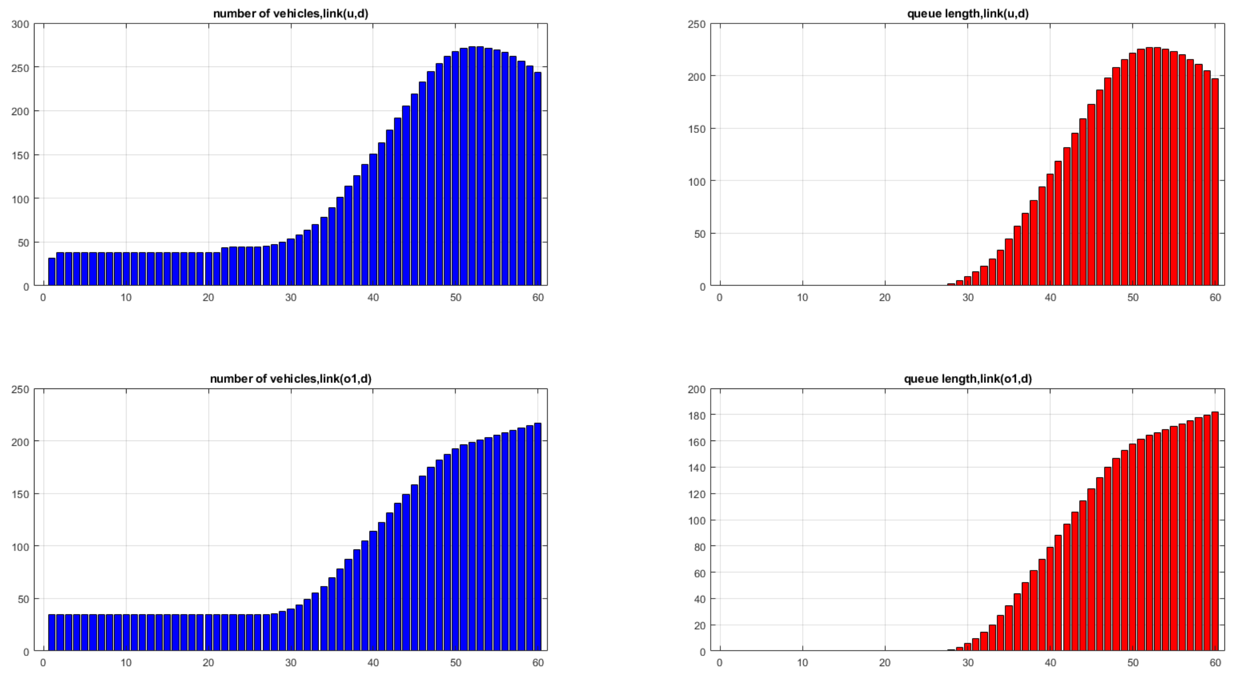


Figure 4: traffic situation simulannealing 15s

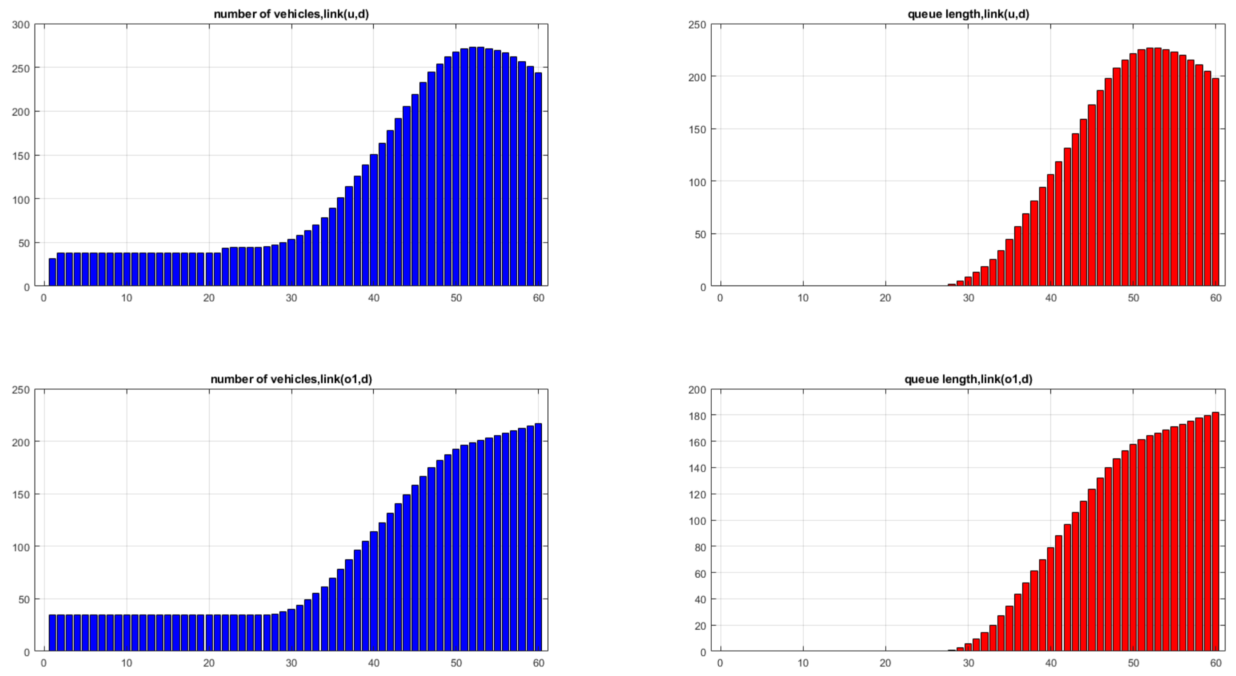


Figure 5: traffic situation simulannealing 45s

Figure 6 and Figure 7 shows the optimal traffic situation after using perpendicular research + golden section steps + annealing from starting points $g_d(k) = 15s, \forall k$ and $g_d(k) =$

$45s, \forall k$

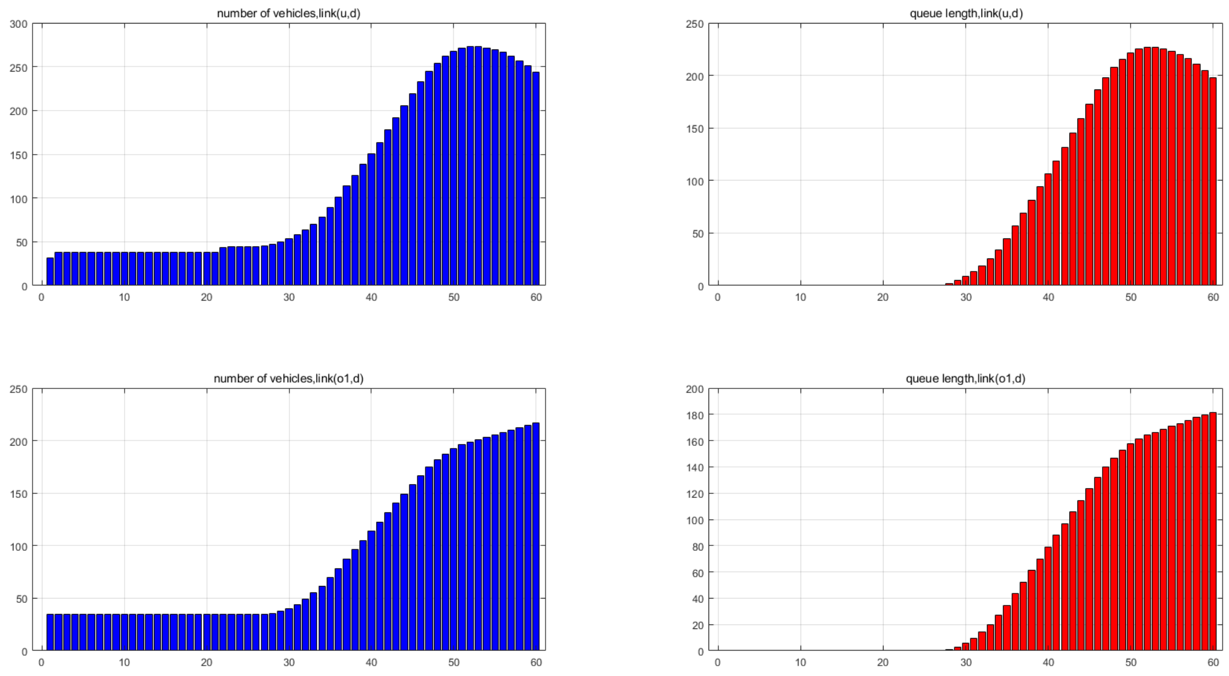


Figure 6: traffic situation perpendicular + golden section 15s

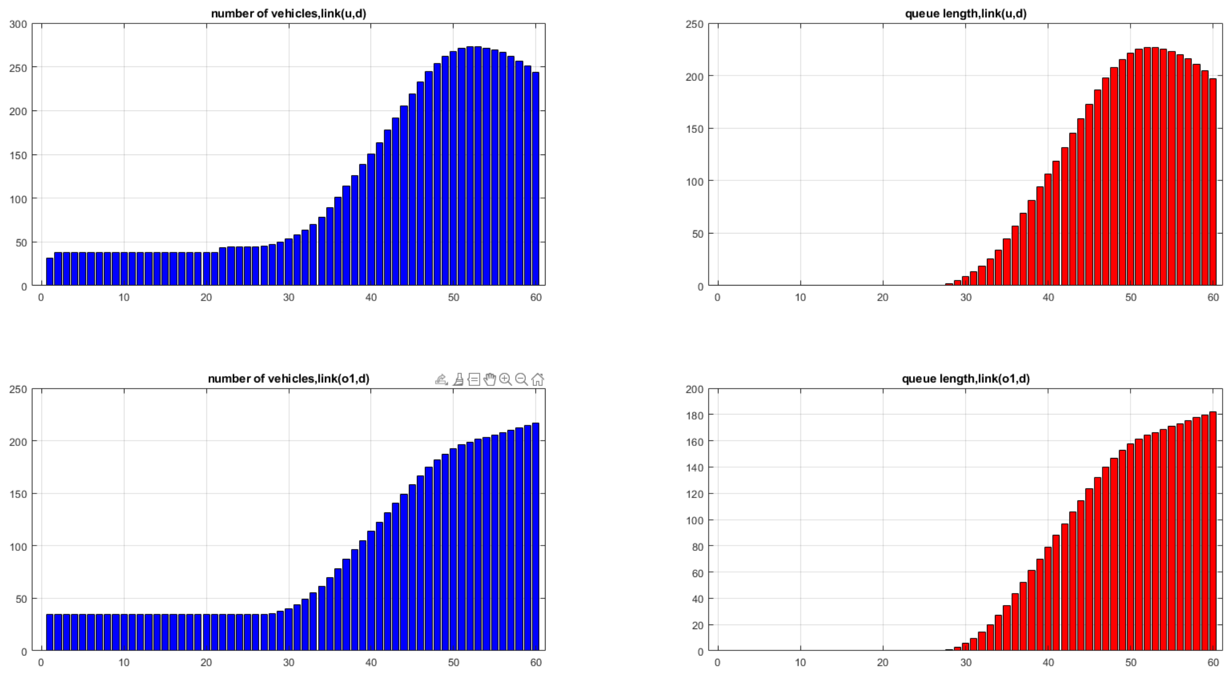


Figure 7: traffic situation perpendicular + golden section 45s

The corresponding inputs of SQP,Interior-Point and Quasi-Newton Method is shown in

Figure 8. The corresponding inputs of perpendicular search + Golden step method are shown in Figure 12.

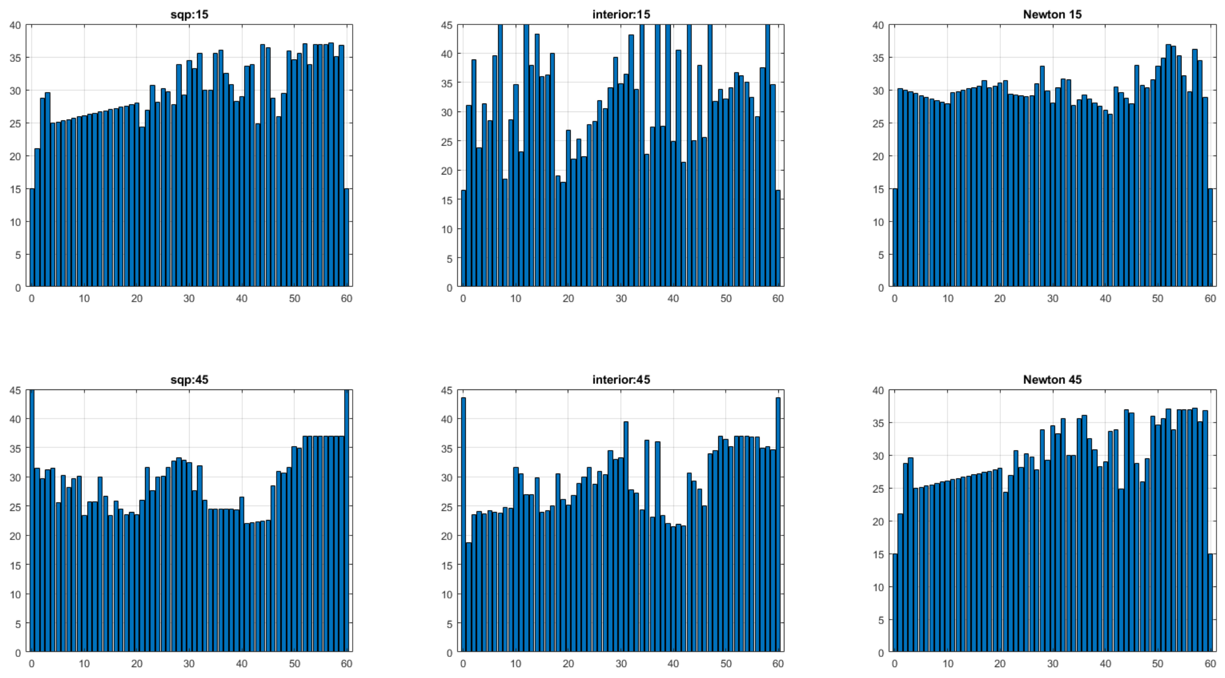


Figure 8: Input of SQP,Interior-Point and Quasi-Newton Method 45s

Both optimal inputs decrease $g_d(k)$ when k is small, and increase $g_d(k)$ when k is large. It is reasonable since when k is small, link(o_1, d) has more flow than link(u, d), the pressure of link(u, d) increasing as k goes on. And these changes in inputs make the value and duration of peak of link(u, d) reduced. Although the side-effect is the increasing of peak of link(o_1, d), TTS are decreased.

From the $u(t)$ diagram, in early k, the Interior-Point methods has more high value, while in late k, it has more low value. That is why it is worse than the other solutions.

The queue length of each lane no control is shown in Figure9, that based on the optimal input from SQP (45s starting points) is shown in Figure 10 .

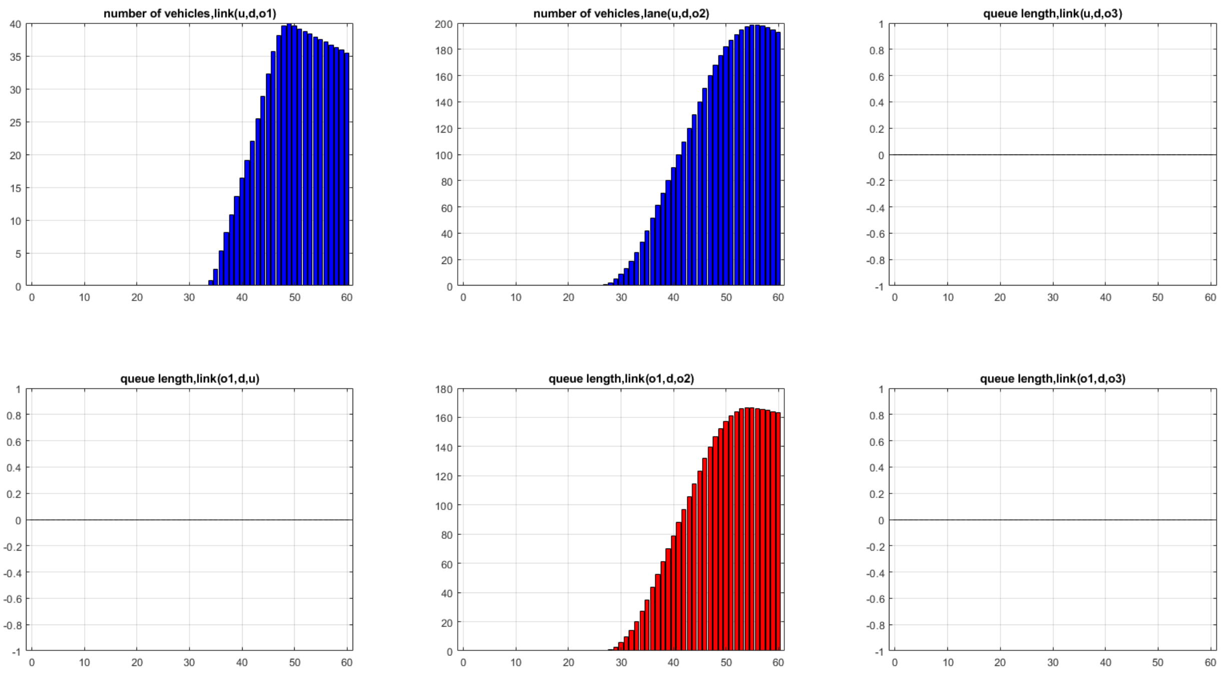


Figure 9: Queue each lane no control

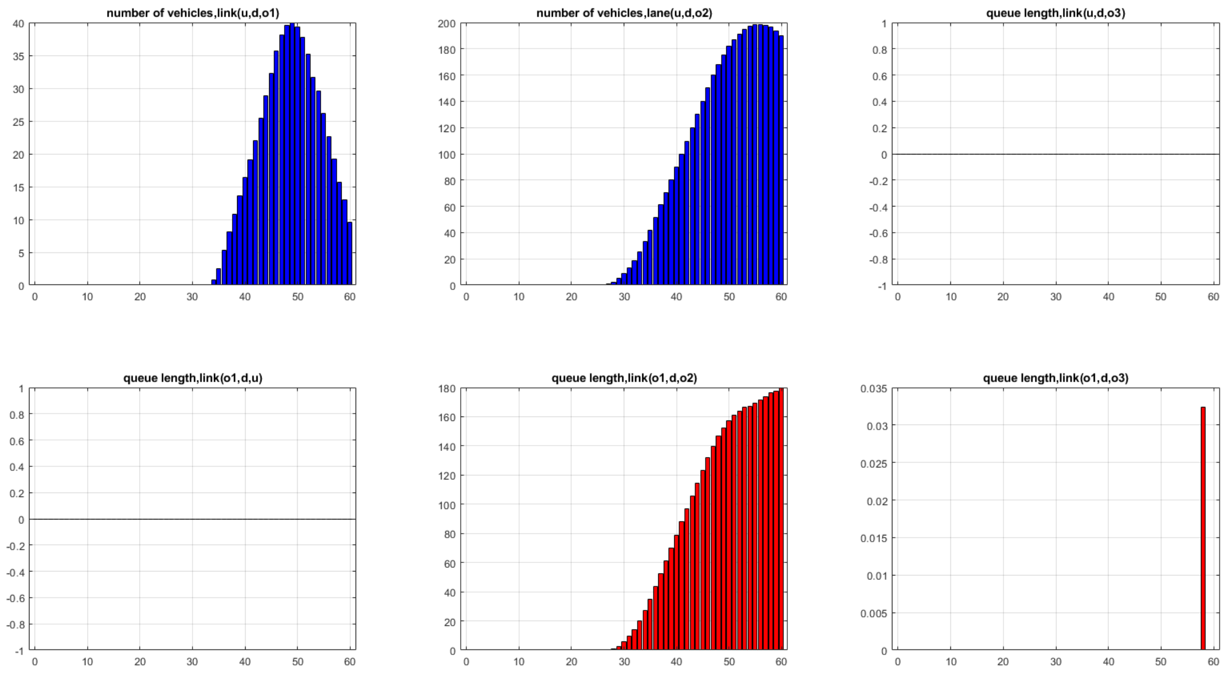
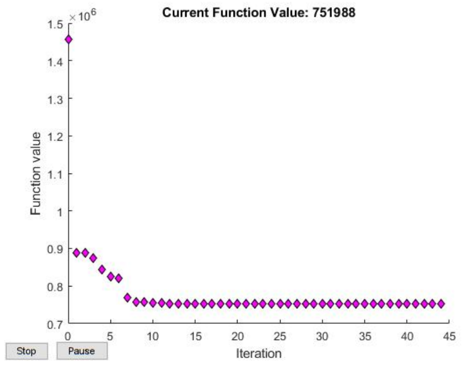


Figure 10: Queue each lane SQP 45s

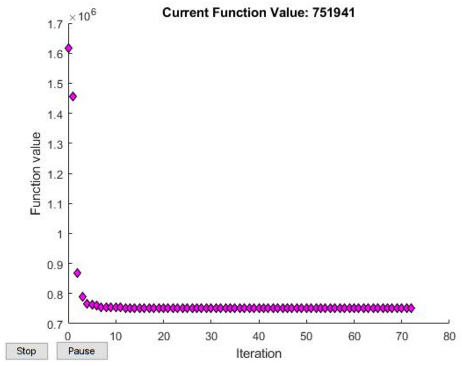
The optimal input makes the peak of link(u,d) drop faster while a little more pressure is taken into link(o_1, d). After time step 40, more flow enter link(u,d), the optimal input does have positive effect.

2. Process Compare

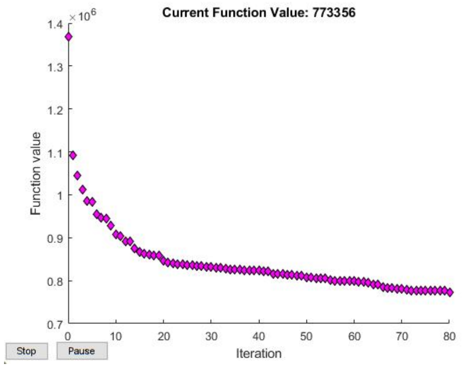
   Figure 11 shows the optimization process of Existing Matlab tools algorithm.



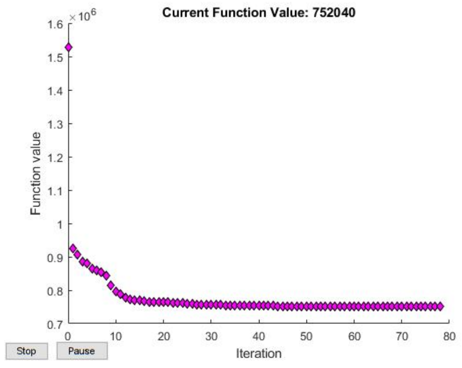
(a) SQP:15



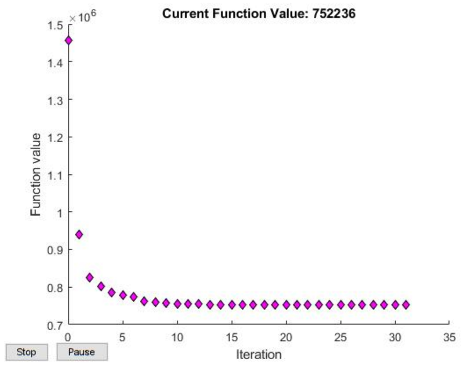
(b) SQP:45



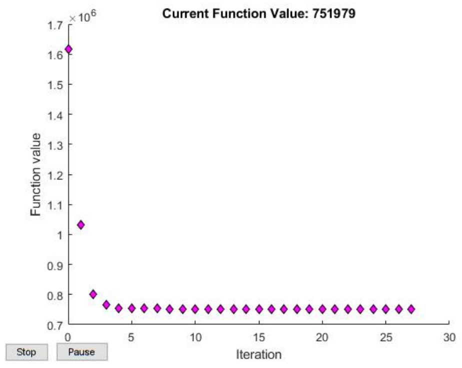
(c) Interior:15



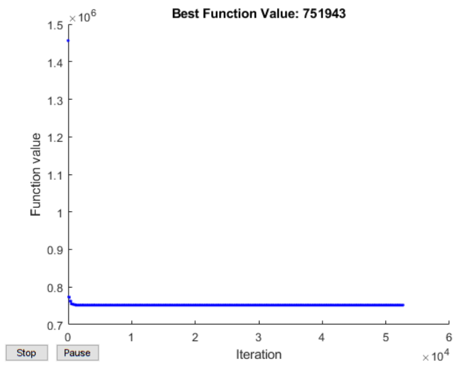
(d) Interior:15



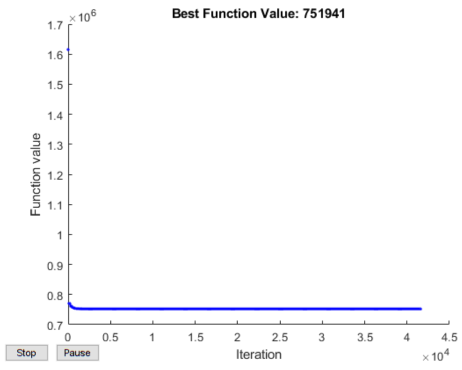
(e) Newton:15



(f) Newton:45



(g) Annealing:15



(h) Annealing:45

Figure 11: SQP,Quasi-Newton, Annealing,Interior-Point

Figure 12 shows a representative iterative process of one of our many experiments in Perpendicular + Golden Section Method for different initial points. The red line in figures is the TTS of $g_d(k) = 30s, \forall k$

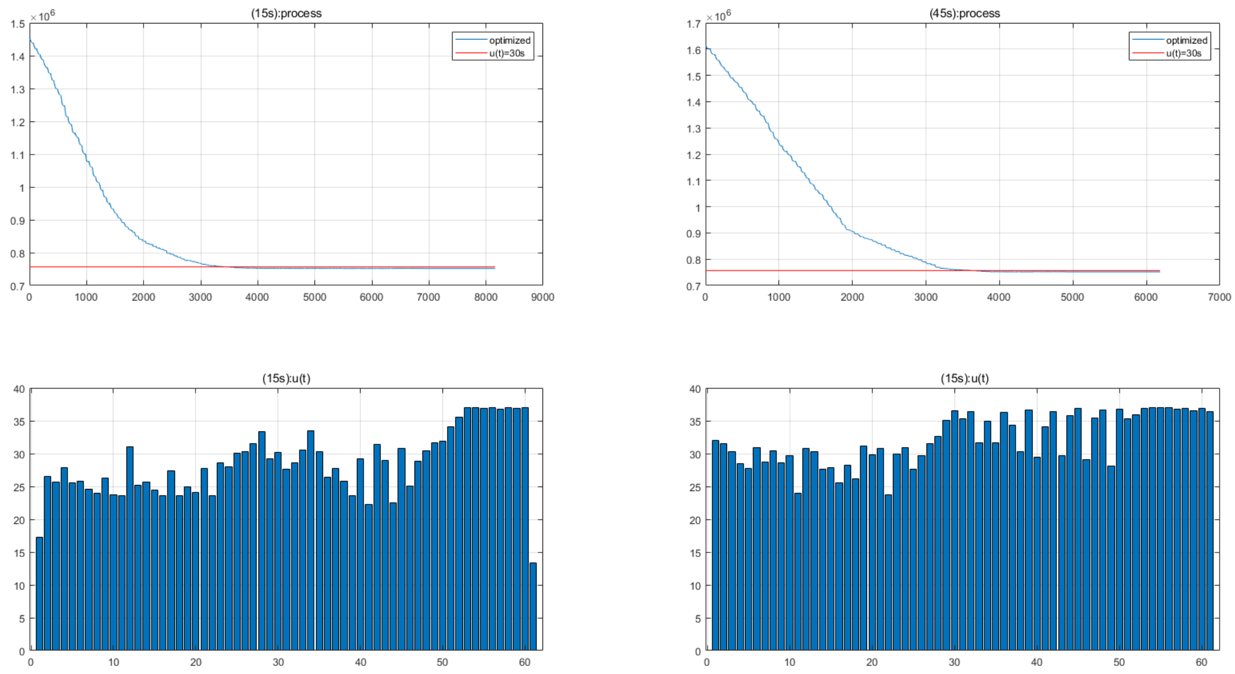


Figure 12: Perpendicular + Golden Section + Annealing

Under the parameters we set.It is obviously that the SQP optimization methods has the higher speed to get to its optimal solution than Interior Methods. Although simulated annealing methods has the most iteration cycles, but the result keep the same from two different starting points. It may because the internal process of MATLAB `|simulannealbnd|` has quick speed to cold down and more circle of iterations, it guarantee the efficiency of the algorithm to get close to local optimal solution. The perpendicular search + Golden Steps methods perform well and it mainly because our program doing golden steps research in a small steps each time. the convergence point speed of the simulated annealing algorithm the slowest than that of the Perpendicular+Golden Section method.

5 Task 5

5.1 Model and Analysis

Based on the model and analysis above, here two methods that is suitable for directly dealing with integer optimization is used to solve this task, they are Genetic Optimization Method and Perpendicular Search Method (with integer step size).

5.2 Solution 1: Genetic Method

The process of Genetic Optimization Method we designed is shown in

Algorithm 1 Genetic Optimization Method.

```
Input: Mutation Possibility for each gene point $P_m$; Total Number of Member in one generation
    $N_m$; Total Number of Generation $N_g$;
Output: Initialization:set up initial population (3 ways)
1: for $i = 1$; $i \leq N_g$; $i++$ do
2:     for $k = 1$;$k \leq N_m$; $k++$ do
3:         Calculate the TTS.
4:     end for
5:     Select Parents $\rightarrow$ the larger TTS, the less probability;
6:     create off-spring: 1-point cross-over;
7:     mutation flip bits with probability $P_m$;
8: end for
```

Typically, in order to use Genetic Optimization Method, binary serials is needed to deal with the "gene" of each "individual". Here for each genetic interval we will use Octal Code, the sufficient reason for Octal Code will be provided later. Gene for an individual is shown as follows:

$$\begin{aligned} &gene = \{g_0 \quad g_1 \quad g_2 \quad \cdots, g_{60}\} \\ &\text{where } g_i \in [1, 7] \text{ and } g_i \in \mathbb{Z} \end{aligned} \tag{1}$$

Here we discussed the sufficient for using the Octonary Number System for each gene bit. Binary Number can also be used, each number in the discrete time set can be numbered with binary serial of 3-bits. However, in order to keep the integral of the gene bit, when we exchange gene interval to create off-spring, the fragment length of exchanged fragment should be a multiple of 3 in order to not break off a gene.

The parameter of Genetic Optimization Algorithm is chosen as follow:

generation round	member number	exchange position	mutation probability	parents number
300	100	30	0.2	20

Table 2: Genetic Optimization Algorithm Parameters

In a problem with minimize objective function, in order to make sure individual with smaller TTS has the higher probability to become parents. And because the number of individual in one generation is quite large, then the followed algorithm is used.

Algorithm 2 Weight of Individual.

```
Input: The list of TTS of each individual: $TTS$; The number of individual that can become parents in a generation: $N_p$
Output: The possibility for each individual to become a parent: weight
1: choose possible parents (the individual with the $N_p$ smallest TTS)
2: for $i = 1;i \leq length(TTS);i++$ do
3:     if individual i is able to become a parent then
4:         weight(i) $\leftarrow$ TTS(i)/sum(TTS of possible parents);
5:     else
6:         weight(i)=0
7:     end if
8:     normalize weight;
9: end for
```

Firstly we will use totally random way for generating the first generation. And it comes that the result is always much worse than the no-control case $g_d(k) = 30s, \forall k$. Then two steps are executed for optimization.

1. add $g_d(k) = 15s, \forall k$, $g_d(k) = 20s, \forall k$, $g_d(k) = 25s, \forall k$, $g_d(k) = 30s, \forall k$, $g_d(k) = 35s, \forall k$, $g_d(k) = 40s, \forall k$ into the first generation and use the Genetic Optimization Method again. Through the iteration, it will appear that most individuals in a generation are composed of $g_d(k) = 30s, \forall k$ serial. This result shows a tendency that the more 30s, the better the solution.

2. From the result of step 1, the result always comes that the finally optimal input is still $g_d(k) = 30s, \forall k$, which means gene bit $g_d(k) = 30s$ is a quite good gene fragment. In this step, we want to figure out whether small change in the whole input serial will lead to better solution. So, we configured a initial generation based on $g_d(k) = 30s, \forall k$ serials. For each individual in the population, each bit of its gene has a possibility to change, and use Genetic Optimization Again.

After the second steps, the result finally becomes better, five result of simulation is chosen and shown in following Table.

index	1	2	3	4	5
result ((10^5))	7.5576	7.5575	7.5652	7.5657	7.5593

Table 3: Genetic Algorithm results Based on mutation of 30s

Some corresponding input is shown in Figure 15. It comes that from the starting points of mutation individual from $g_d(k) = 30s, \forall k$, most results have small improvements. From Figure 15, these improvements are from the increase of $g_d(k)$ in late k.

The transform of states variant of number of vehicles on the link and the queue length of each lane is in Figure 13.

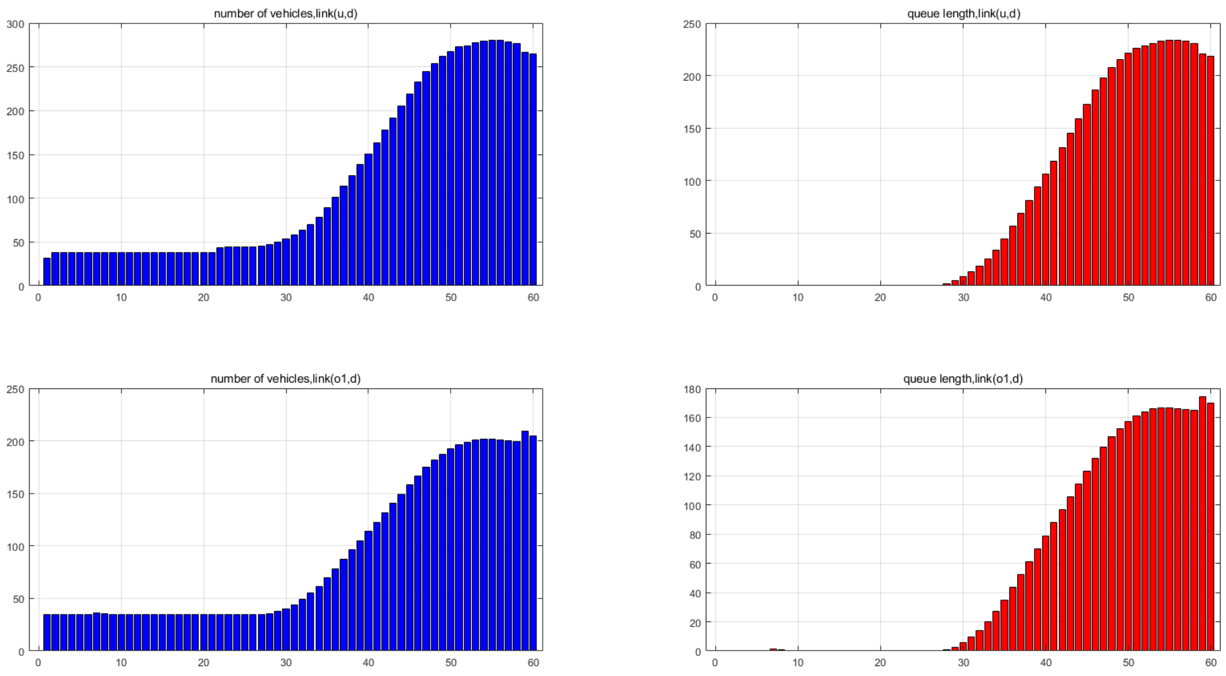


Figure 13: traffic situation genetic algorithm

5.3 Solution 2: Perpendicular Search Method (with integer step size)

Perpendicular Search Method can be modified to deal with integer optimization directly by changing the steps to integer. So we use the Perpendicular Search Method to solve Task 5 in the following way:

Algorithm 3 Perpendicular Search Method(with integer step size).

```
Input: stopping criteria: $ther$
Output: Initialization: randomly generate start point ;
1: while diffrence < $ther$ do
2:     for $i = 1; i \leq 61; i++$ do
3:         direction ← zeros(1,61);
4:         direction(i) ← 1;
5:         for each s ∈ $[15, 20, 25, 30, 35, 40, 45]$ do choose best step
6:         end for
7:     end for
8:     calculate difference;
9: end while
```

The stop criterion is set as when the absolute value of difference between two optimal result in two sequential cycle is less than 1. The iteration will stop after two cycle (contains 120 steps) and the optimal result is shown in the following table.

starting points	without optimization	random starting points	starting points based on 30s
result(10^5)	7.5659	7.5226	7.5226

The transform of states variant of number of vehicles on the link and the queue length of each lane is in Figure 14.

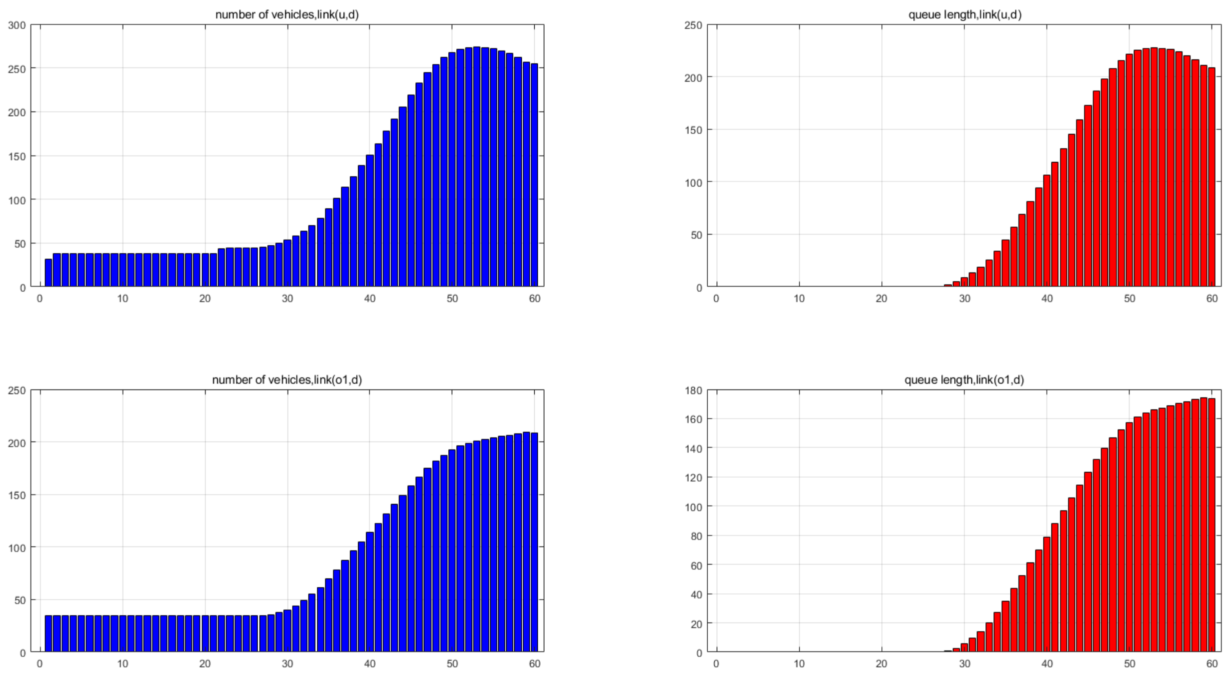


Figure 14: traffic situation perpendicular search algorithm

Neither of these two solutions guarantees a global optimal solutions, but the result comes to be reasonable. The result is easy to explain: at beginning, all links do not have pressure on the traffic flow, some small decrease of $g_d(k)$ has been made to postpone the appearance of queue on link(u,d). As time goes on, some small increase of $g_d(k)$ has been made to relive the pressure of link(u,d).

5.4 Compare

The change of output y following the iteration of two process is shown in this part.

5.4.1 Genetic

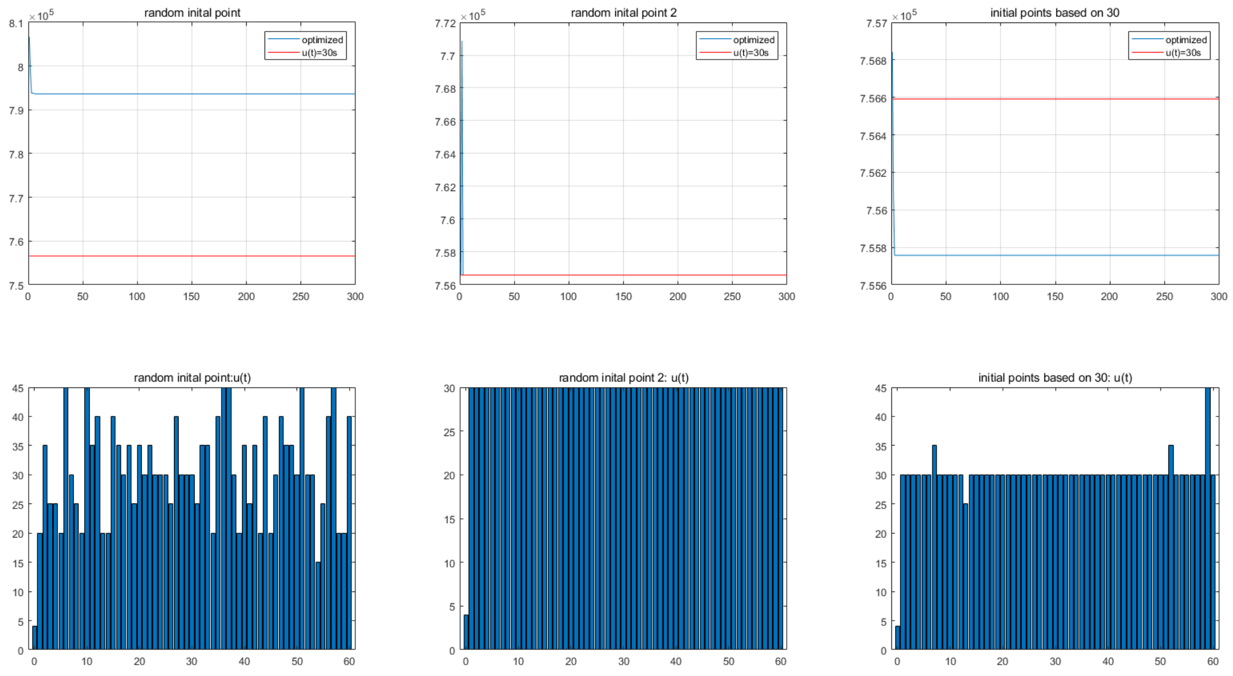


Figure 15: Genetic Algorithm results

1. Genetic Optimization with Random Initial Generation:

   By using Genetic Optimization with totally random initial generation, the results have high possibility worse than $g_d(k) = 30s.\forall k$.

2. Genetic Optimization with Initial Generation contain $g_d(k) = constant \quad \forall k$

   By using Genetic Optimization with Initial Generation contain $g_d(k) = constant \quad \forall k$, the results seems to close to the result of $g_d(k) = 30s.\forall k$, but most times, not better than it.

3. Genetic Optimization with Initial Generation based on the mutation of $g_d(k) = 30s \quad \forall k$:

   By using Genetic Optimization with Initial Generation based on the mutation of $g_d(k) = 30s \quad \forall k$, the optimization will be better than $g_d(k) = 30s.\forall k$.

The difference among result of these two kinds initial points is reasonable. In the third kind of initial point, some mutation may make the result better, and these high-quality mutations have a higher probability of being passed on to the next generation.

5.4.2 Perpendicular

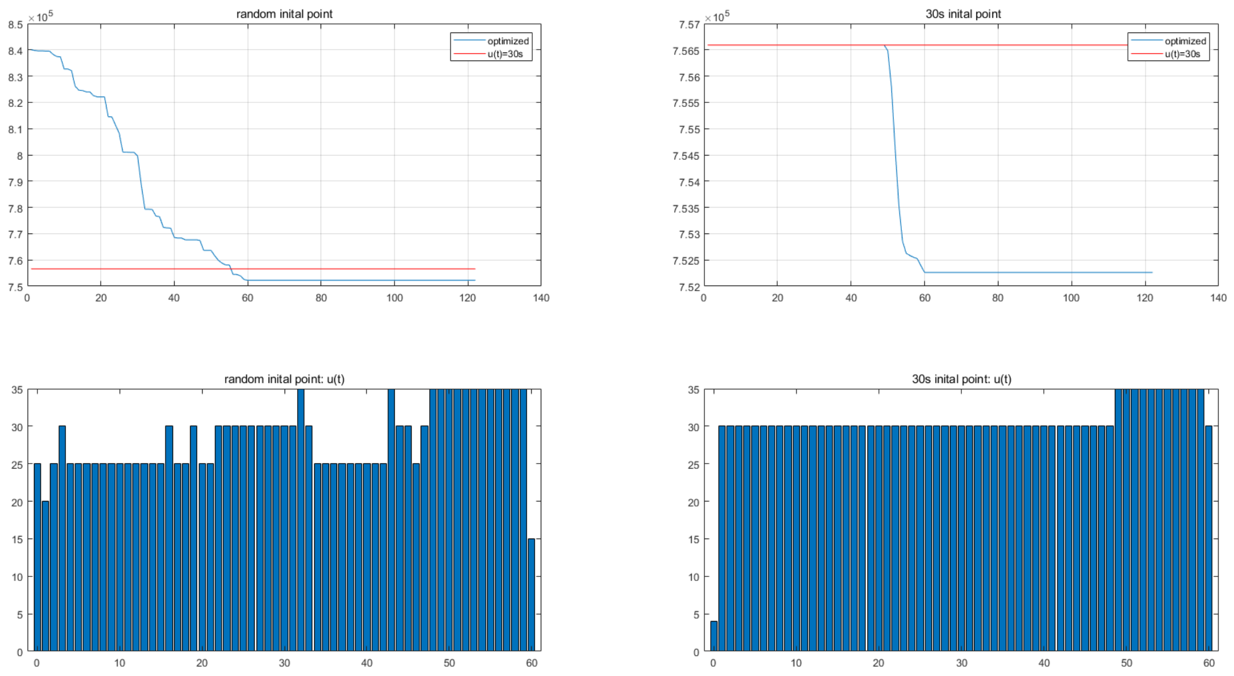


Figure 16: Perpendicular Search Algorithm results

1. Random Initial Points

   The final result is always worse than $g_d(k) = 30s.\forall k$;

2. Initial Points from $g_d(k) = 30s.\forall k$

   The final result is better than $g_d(k) = 30s.\forall k$;

The difference among results of these two kinds initial points is reasonable. Both of these results not guarantee global optimum. Random Initial Points will start from its initial points and gradually trapped into a local minimum points that is much worse than $g_d(k) = 30s.\forall k$. For Initial Points from $g_d(k) = 30s.\forall k$,the result comes that from this initial points some positive changes has been made, but for most k, the input still the same. The potential reason is that for some changes in the early k, changes may do not generate better results immediately, but it still has possibility to later k, from a global optimization views.

5.5 Potential Improvements Method

From subsection 5.4, the potential reason that perpendicular methods always stop at 60 or 120 steps is that for some changes in the early k, changes may do not generate better results immediately, but it still has possibility to later k, from a global optimization views. It is similar to the disadvantages of greedy algorithms, that is, the current step do not take later optimization into consideration.

A potential improvement way is taking the advantage of simulate annealing methods, a threshold can be introduced to help us accept some changes that may not generate better results immediately but has positive influence for later k. When k is lose to high traffic pressure moment, we can adjust the threshold to increase the possibility to accept such changes.