Delft University of Technology

Optimisation for system and Control
SC42056

Assignment 2: NLP Assignment

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

October 30, 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

Firstly, we clarify the variable we use and their corresponding name in the document.

$$
\begin{aligned}
y(k+1) =& g(x(k), u(k)) \\
x(k+1) =& [s(k+1), z(k+1), m(k+1), o(k+1)]^T \\
=& [x_1(k+1), x_2(k+1), \cdots x_{26}(k+1)] = f(x(k), u(k)) \\
\text{where } s(k) =& [s_1(k), s_2(k), \cdots, s_{10}(k)] \\
=& \big[n_{u,d}(k), n_{o_1,d}(k), \alpha^{arrive}_{u,d}(k)\alpha^{arrive}_{o1,d}(k), q_{u,d,o_1}(k), ... \\
& q_{u,d,o_2}(k), q_{u,d,o_3}(k), q_{o_1,d,u}(k), q_{o_1,d,o_2}(k), q_{o_1,d,o_3}(k)\big] \\
z(k) =& [z_1(k), z_2(k), \cdots, z_6(k)] \\
=& \big[\alpha^{enter}_{u,d}(k), \alpha^{enter}_{o1,d}(k), C_{d,u}(k), C_{d,o_1}(k), C_{d,o_2}(k), C_{d,o_3}(k)\big] \\
m(k) =& [m_1(k), m_2(k), m_3(k), m_4(k)] \\
=& \big[\alpha^{leave}_{u,d}(k), \alpha^{leave}_{o_1,d}(k), q_{u,d}(k), q_{o_1,d}(k)\big] \\
o(k) =& [o_1(k), o_2(k), \cdots o_6(k)] \\
=& \big[\alpha^{leave}_{u,d,o_1}(k), \alpha^{leave}_{u,d,o_2}(k), \alpha^{leave}_{u,d,o_3}(k), \alpha^{leave}_{o_1,d,u}(k), \alpha^{leave}_{o_1,d,o_2}(k), \alpha^{leave}_{o_1,d,o_3}(k)\big] \\
f(x(k), u(k)) =& [f_1(x(k), u(k)), f_2(x(k), u(k)) \cdots, f_{26}(x(k), u(k))]
\end{aligned}
\tag{1}
$$

1.2 State Space Model

The state space relation are as follows. For equations like f_{17}, f_{18}, we described it as $f_{17}(x(k+1), u(k+1))$, but obviously, the element in the right side can actually be subscribed by $f_i(x(k), u(k))$. For connivance, $f_1(x(k+1), u(k+1))$ is represented as f_1 and so on. Equation of variables s:

$$
\begin{aligned}
f_1 &: s_1(k+1) = s_1(k+1) + c(z_1(k+1) + m_1(k+1)) \\
f_2 &: s_2(k+1) = s_2(k+1) + c(z_2(k+1) + m_2(k+1)) \\
f_3 &: s_3(k+1) = \frac{c-\gamma_1(k+1)}{c} z_1(k+1-\tau_1(k+1)) + \frac{\gamma_1(k+1)}{c} z_1(k-\tau_1(k+1)) \\
f_4 &: s_4(k+1) = \frac{c-\gamma_2(k+1)}{c} z_2(k+1-\tau_2(k+1)) + \frac{\gamma_2(k+1)}{c} z_2(k-\tau_2(k+1)) \\
f_5 &: s_5(k+1) = s_5(k+1) + c(\beta_{udo_1} x_3(k+1) - o_1(k+1)) \\
f_6 &: s_6(k+1) = s_6(k+1) + c(\beta_{udo_2} x_3(k+1) - o_2(k+1)) \\
f_7 &: s_7(k+1) = s_7(k+1) + c(\beta_{udo_3} x_3(k+1) - o_3(k+1)) \\
f_8 &: s_8(k+1) = s_8(k+1) + c(\beta_{o_1 du} x_3(k+1) - o_4(k+1)) \\
f_9 &: s_9(k+1) = s_9(k+1) + c(\beta_{o_1 do_2} x_3(k+1) - o_5(k+1)) \\
f_{10} &: s_{10}(k+1) = s_{10}(k+1) + c(\beta_{o_1 do_3} x_3(k+1) - o_6(k+1))
\end{aligned} \tag{2}
$$

The expression of $\gamma_1(k)$, $\gamma_2(k)$, $\tau_1(k)$, $\tau_2(k)$ are the same as assignment implied.
Equation of variables z:

$$
\begin{aligned}
f_{11} &: z_1(k+1) = \begin{cases} 1800 + 10E1, -2 < & k \le 19, \\ 2100 + 10E2, 19 < & k \le 39, \\ 2300 + 10E3, & k > 39 \end{cases} \\
f_{12} &: z_2(k+1) = 2000 + 10E1 \\
f_{13} &: z_3(k+1) = \begin{cases} 40 - E3, -2 < & k \le 29 \\ 40 + E3, & k > 29 \end{cases} \\
f_{14} &: z_4(k+1) = \begin{cases} 40 + E1, & -2 < & k \le 19 \\ 40 + E1 - 2(k-20), & 19 < & k \le 34 \\ 10 + E1, & 35 < & k \le 44 \\ 10 + E1 + 2(k-45), & & k > 44 \end{cases} \\
f_{15} &: z_5(k+1) = z_4(k+1) - E2 \\
f_{16} &: z_6(k+1) = \begin{cases} 30 - E3, -2 < & k \le 29 \\ 30 + E3, & k > 29 \end{cases}
\end{aligned} \tag{3}
$$

Equation of variables m:

$$
\begin{aligned}
f_{17} &: m_1(k+1) = o_1(k+1) + o_2(k+1) + o_3(k+1) \\
f_{18} &: m_2(k+1) = o_4(k+1) + o_5(k+1) + o_6(k+1) \\
f_{19} &: m_3(k+1) = x_5(k+1) + x_6(k+1) + x_7(k+1) \\
f_{20} &: m_4(k+1) = x_8(k+1) + x_9(k+1) + x_{10}(k+1)
\end{aligned} \tag{4}
$$

Equation of variables o:

$$
\begin{aligned}
f_{21} : o_1(k+1) &= \min(\frac{\mu_{udo1}u(k+1)}{c}, \frac{x_5(k+1)}{c} + \beta_{udo1}x_3(k+1), \frac{z_4(k+1)}{c}) \\
f_{22} : o_2(k+1) &= \min(\frac{\mu_{udo2}u(k+1)}{c}, \frac{x_6(k+1)}{c} + \beta_{udo2}x_3(k+1), \frac{z_5(k+1)}{c}) \\
f_{23} : o_3(k+1) &= \min(\frac{\mu_{udo3}u(k+1)}{c}, \frac{x_7(k+1)}{c} + \beta_{udo3}x_3(k+1), \frac{z_6(k+1)}{c}) \\
f_{24} : o_4(k+1) &= \min(\frac{\mu_{o1du}u(k+1)}{c}, \frac{x_8(k+1)}{c} + \beta_{udo1}x_4(k+1), \frac{z_3(k+1)}{c}) \\
f_{25} : o_5(k+1) &= \min(\frac{\mu_{o1do2}u(k+1)}{c}, \frac{x_9(k+1)}{c} + \beta_{udo1}x_4(k+1), \frac{z_5(k+1)}{c}) \\
f_{26} : o_6(k+1) &= \min(\frac{\mu_{o1do3}u(k+1)}{c}, \frac{x_{10}(k+1)}{c} + \beta_{udo1}x_4(k+1), \frac{z_6(k+1)}{c})
\end{aligned}
\tag{5}
$$

2 Task 2

2.1 Model and Analysis

The objective function is formalized as follows, when the optimization goal is to find suitable green light time length of each interval for the following hour, which leads to a minimized Total Time Spent(TTS) by the drivers on the link (u, d) and (o_1, d).

$$
\begin{aligned}
\min_{u_k} \quad \sum_{k=1}^{61} y = \quad & \sum_{k=1}^{61} y_1(k) + y_2(k) \\
= \quad & \sum_{k=1}^{61} c(n_{u,d}(k) + n_{o_1,d}(k)) = \sum_{k=1}^{61} h(x(k)) \\
s.t. \quad & x(k+1) = f(x(k), u(k)) \\
& y(k+1) = g(x(k), u(k))
\end{aligned}
\tag{6}
$$

Where $n_{u,d}(k)$ represent the number of cars in $link(u, d)$ at k moment, $n_{o_1,d}(k)$ represent the number of cars in $link(o_1, d)$ at k moment

First of all, it should be clarified that for the system during time $[0, 60c]$, if $u(k)$ for each interval and the initial states of the system are given, the value of each state parameter and objective functions can be specified. That means functions in Task 1 can be transformed to function about $u(k)$ and $x(0)$.

It is obvious that this objective function is a linear combination of $n_{u,d}(k)$ and $n_{o_1,d}(k)$. But because minimum function and reminder function exist inside constraints, the problem is a nonlinear and non-convex problem, and the gradient is hard to obtain because of the existence of minimum function and reminder function.

Therefore, we can only choose the algorithms suitable for for nonlinear, non-convex problem, and it is better that the algorithm has less dependent on gradient.

The next question is how to to simplify the constraints of this problem. As mentioned above, for this system during time $[0, 60c]$, if $u(k)$ for each interval and the initial states of this

system are given, the value of system sate at each time and the value of objective function can be specified (by using state space model, the probable constraints of this problem can be calculated recurrently). Although the constraints seems complex, we can include them in the objective function, although we cannot write it out directly (so we use function h to represent it). This means that the output is represented by the constraints and input, detailedly by $x(0)$ and $u(k)$.

$$\begin{aligned} \min_{u_k} \quad \sum_{k=1}^{61} y = & \quad \sum_{k=1}^{61} y_1(k) + y_2(k) \\ = & \quad \sum_{k=1}^{61} c(n_{u,d}(k) + n_{o_1,d}(k)) \\ = & \quad \sum_{k=1}^{61} h(x(k)) = h(x(0), u(k)) \\ s.t. \quad & 15 \leq u(k) \leq 45 \end{aligned} \tag{7}$$

Although h can not be written directly, when using MATLAB, the recursive relationship can be expressed by a `.m` file contains a function as a `function handle`. This is because, although this process is complicated, it still conforms the definition of function **"a function is a binary relation between two sets that associates every element of the first set to exactly one element of the second set."** when the input element and an implementation of $u(k)$ are given, exactly one y is mapped.

As explained above, in the objective function, most constraints can be substituted by h. That means that this problem can be interpreted as a problem like model 6 or a problem only with variables $u(k)$ with lower/upper limit like model 7 or problem like model 6. We choose the second way.

2.2 Choosing algorithm

From Task2, we firstly considered SQP optimization methods and Interior Points methods, and then we used Quasi-Newton Methods which regards this problem as an unconstrained problem and then we checked the answer whether it meets bounds of input. These three methods can be directly executed by MATLAB functions. The existing functions of MATLAB do not provide a direct optimization method of direction search+step size selection. we then tried to use perpendicular search method combined with golden section method to select step size. Finally we compare the results of these methods in Task 4.

Here, we provide the description of our search methods and the parameters that we choose when using MATLAB optimization tools.

2.2.1 SQP,Interior Points & Quasi-Newton

SQP and Interior Points Methods are good ways to solve nonlinear and non-convex problems (with constraints). We use MATLAB tools `fmincon` to implement it.
Quasi-Newton method is a good method to unconstrained methods, but from analysis, we only preserved the lower/upper bound of input as constraints, so we use MATLAB tool `fminunc` to

solve it at first and check whether the optimal solution meets the bounds.

2.2.2 Simulated Annealing Method

When this problem degenerates into a non-constrained problem, we can randomly search for the optimization from starting point. We can select the initial temperature and corresponding cooling rate. While temperature is going down, we can select a random solution around the previous one and compare them. We can update this solution depending on a probability controlled by temperature. We use MATLAB tool `simulannealbnd` to implement it.

2.2.3 Perpendicular Searching

Instead of randomly selecting the next solution, we can search in a predefined direction or in a perpendicular way. We can only change one of its components at a time by moving forward step by step.

While perpendicular searching is used, we use golden section method to select step size. Because we don't know whether this curve is uni-modal or not, some better optimal points will be missed. To improve this drawback, Golden Section is necessary.
The implementation of Perpendicular Search Method is described as follows:

Algorithm 1 Perpendicular Search Method

```
1: function PERPENDICULAR(u_initial, iter)
2:     count ← 1
3:     TS_previous ← calculate TTS given u_initial
4:     calculate TTS1 given input u1;
5:     while count < iter do
6:         for i = 1 → 61 do
7:             [u_temp, TS_temp] ← GOLDEN SECTION(i, u_initial)
8:             if u_temp > TS_previous then
9:             else
10:                u_initial ← u_temp
11:                TTS_initial ← TS_temp
12:            end if
13:        end for
14:        if TTS doesn't change after finishing a For loop then
15:            count ← count + 1
16:        end if
17:        return TS_temp, u_temp
18:    end while
19: end function
```

2.2.4 Golden Section steps

A local minimum can be found by applying Golden Section, this local minimum should be included when considering this task in a greedy way.

The implementation of Golden Section steps is described as follows:

Algorithm 2 Golden Section Method

Input: Time length of green light: $u\_initial$, number of iteration: $iter$
Output: Total Time Spent: TTS, optimized input:u_final

```
function GOLDEN SECTION(u_initial, k)
    choose a random value step ∈ [−1, 1];
    while step makes initial input violate its bounds do
        generate new step;
    end while
    Initialize max iteration number, origin bound, origin section points and other parameters
    calculate corresponding TTS values for section points, labelled as {f1, f2, flb, fub}
    for i = 1 → nIter do
        update section points with {f1, f2, flb, fub}
    end for
    return TTS, u
end function
```

3 Task 3

All detailed results with figures are shown in Task 4. In this task we only mention some of them and analyze the result. The optimal result of different methods is shown below:

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: Optimal Result

3.1 SQP, Interior points and Quasi-Newton

From starting points $g_d(k) = 45s, \forall k$ SQP methods, Interior points methods and quasi-newton methods obtain better solutions than that from starting points $g_d(k) = 15s, \forall k$. The result of SQP and quasi-newton are almost the best solution in the Table, and the difference of them between result from different starting points of them is not substantial different. All of them do not guarantee global optimal result, the difference between different starting points may result from: compared with no-control situation, the most important flow pressure appears in late k, which requires the increase of $g_d(k)$. For the early stage, because the flow pressure is not so great, the change is not so important. Therefore, the starting point of $g_d(k) = 45s, \forall k$ may get slightly better result.

3.2 Simulated Annealing

The parameters of MATLAB function of simulated annealing method is set to:

parameter	intial_temperature	coolin_speed	delta_tolerance	max_interation_times
value	100	0.95	1	100

Table 2: Parameters of simulated annealing method

Although simulated annealing methods always cannot get a real optimal solution, but the result is that it has one of the best results in the table above. And from different starting points the result keeps the same. That may be due to the parameter chosen: fasting cooling and multiple rotations. The first thing ensures that starting from a starting point, a worse next point is unlikely to be accepted after a short period, and the second thing helps to jump out of a local optimum. But compared to other methods, it has the largest number of rotations.

3.3 Perpendicular and Golden Section

The combination of perpendicular direction and golden section steps results in an optimal solution that is very close to the simulated annealing method. Results are slightly different from different starting points. When solving this task in perpendicular way, we do have a certain searching direction, for each direction, we choose a random range (not large) and do golden section in this range, the result may vary and may cost different running time. However, the random step size can avoid falling into local optimal solution to some extent.

3.4 Result Judgement

None of these models can guarantee global optimum, but some of them get the same optimum results those are best in the table. We assert that result 7.5194 is very close to the global optimal answer.

3.5 Improvement

For Perpendicular and Golden section search methods, there do has some drawbacks:

1. Firstly, When searching in perpendicular way, we choose the local minimum that is realized by using Golden Section. However, we do not know what does the function looks like following the direction. Some better point may be dismissed.

2. Every time we choose an optimal next point from a single direction, this approach is a bit greedy. Choice of single direction in the early stage may have great influence on the choice in the later stage. That means choosing a local minimum point may lead to a large deviation from the global minimum point.

Enlightened by Simulated Annealing, we may maintain a probability of rejecting a local optimal point, this probability will decrease when searching is deeper.

4 Task 4

4.1 Analysis

It is obvious from task 2 that with the parameters in task 2, when the initial state and the input are given, the states and output of the system at time k ($k >= 0$) are designated. 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.

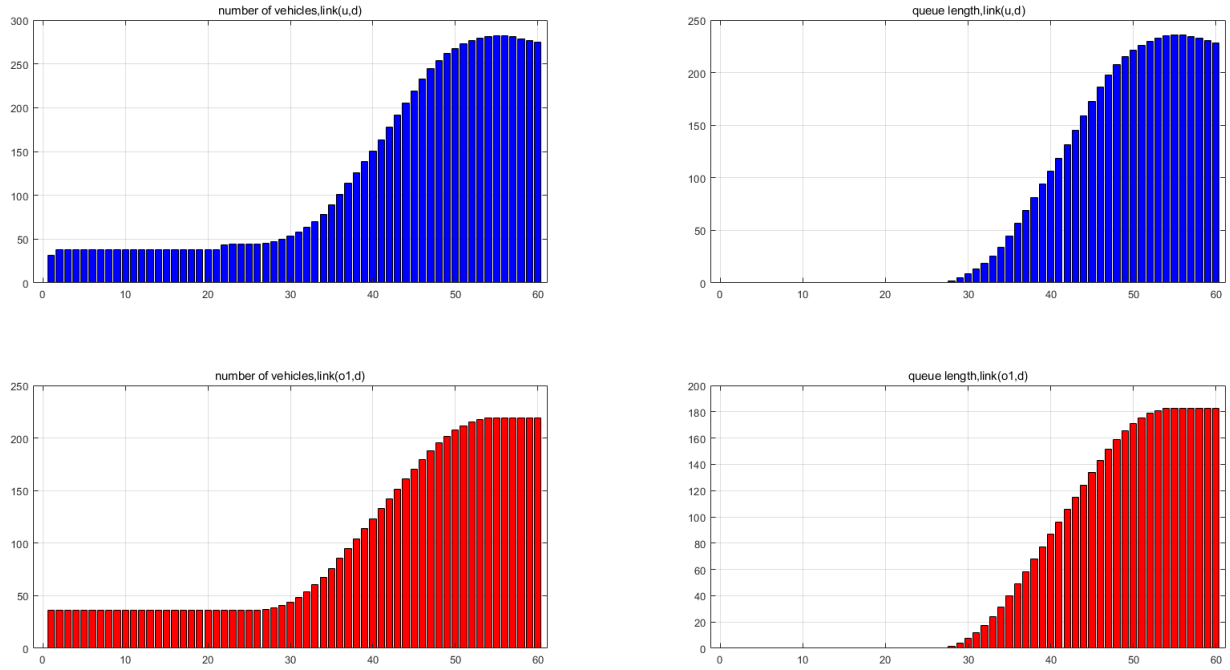


Figure 1: traffic situation no control

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 3: Table: Optimal Result

Figure 2 shows 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 $g(k) = 45s, \forall k$.

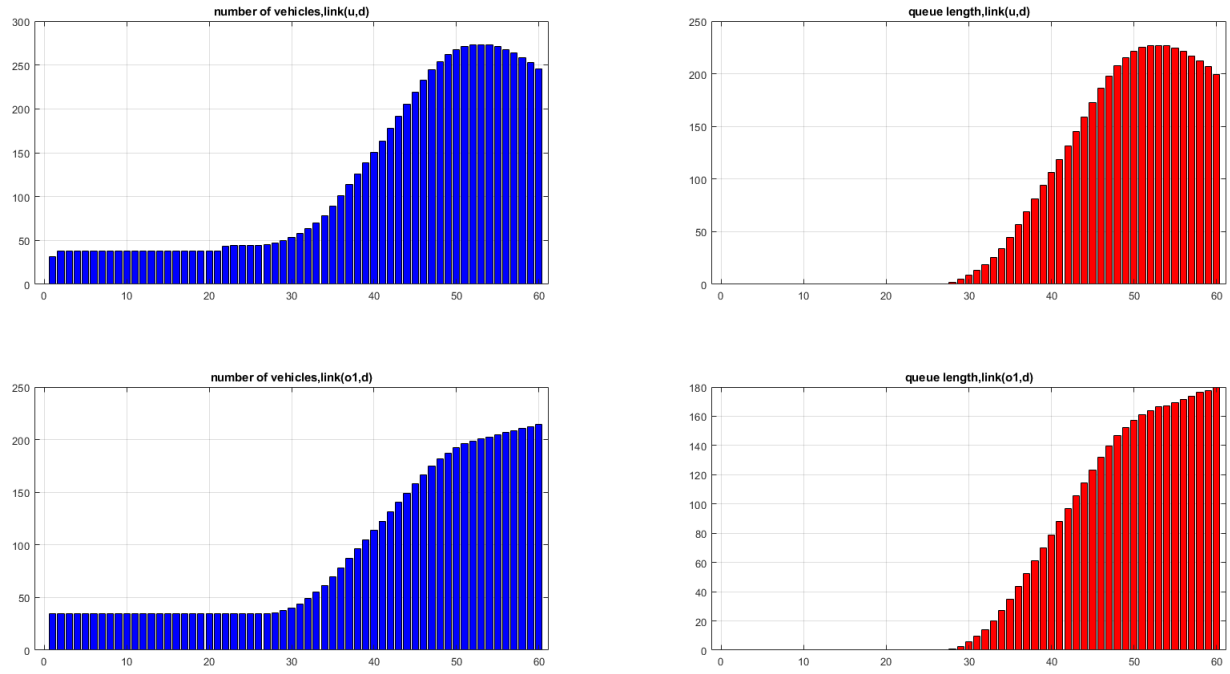


Figure 2: traffic situation SQP 15s

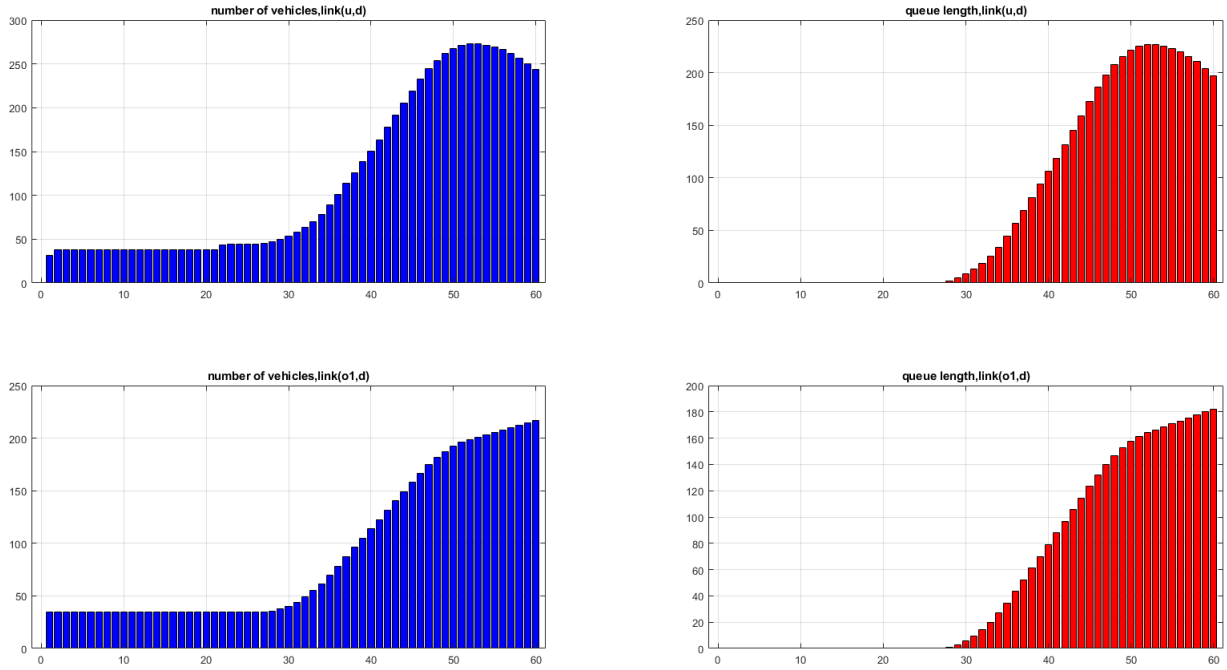


Figure 3: traffic situation SQP 45s

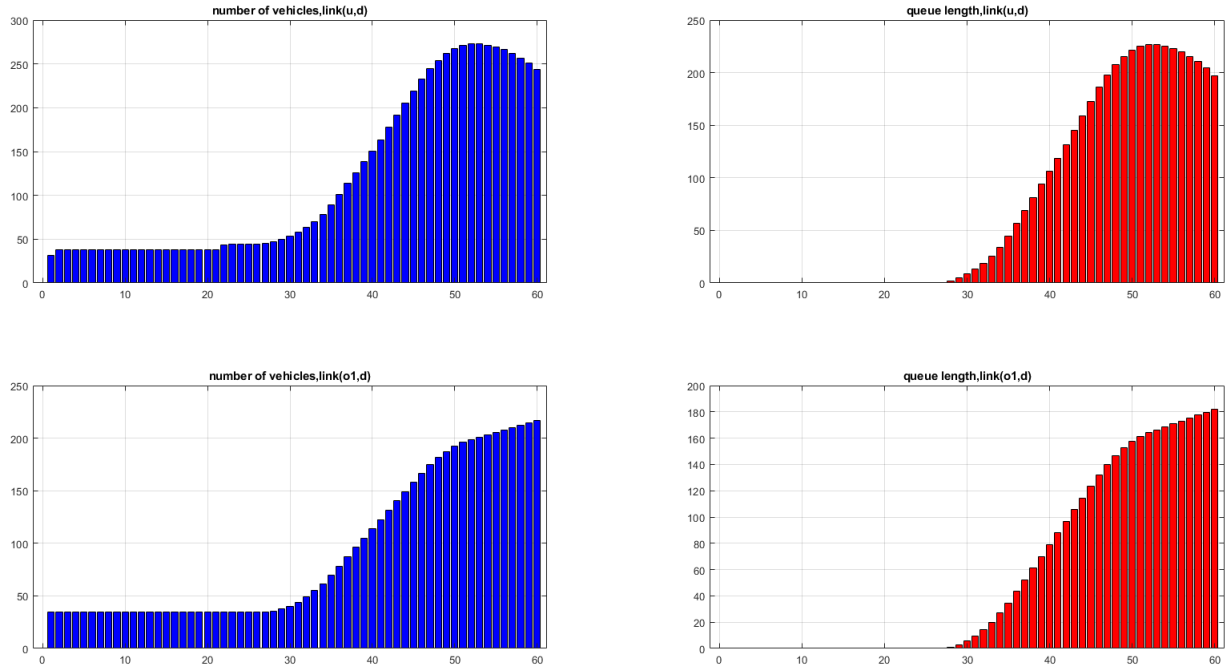


Figure 4: traffic situation simulannealing 15s

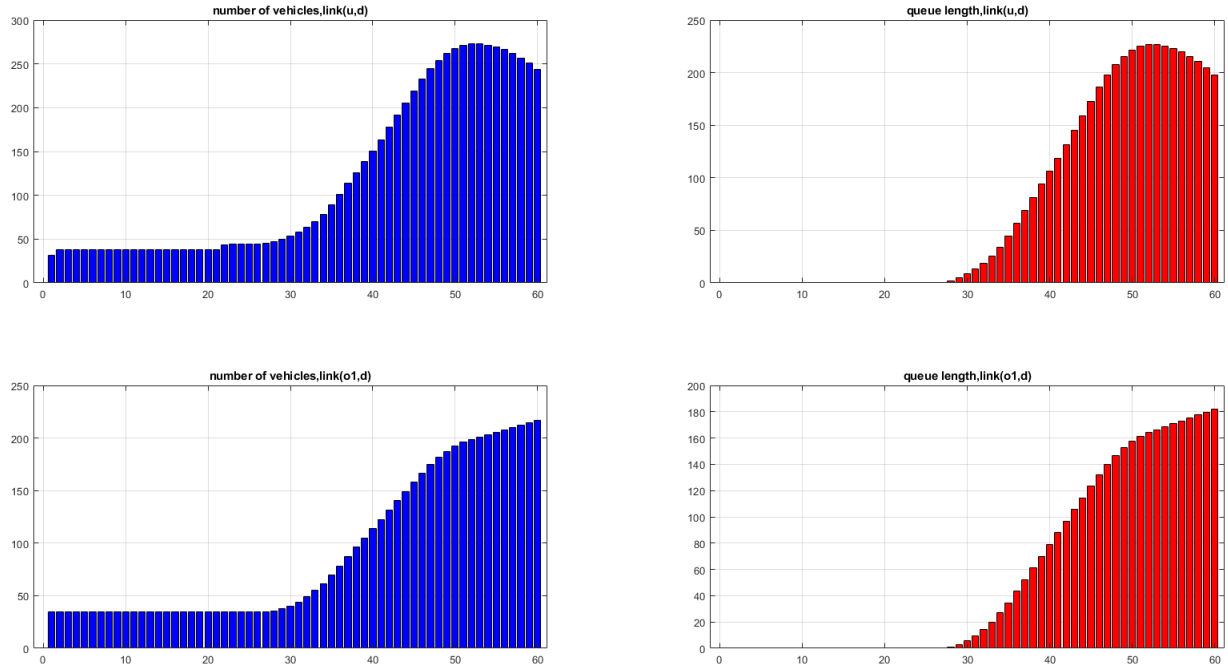


Figure 5: traffic situation simulannealing 45s

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 $g_d(k) = 45s, \forall k$

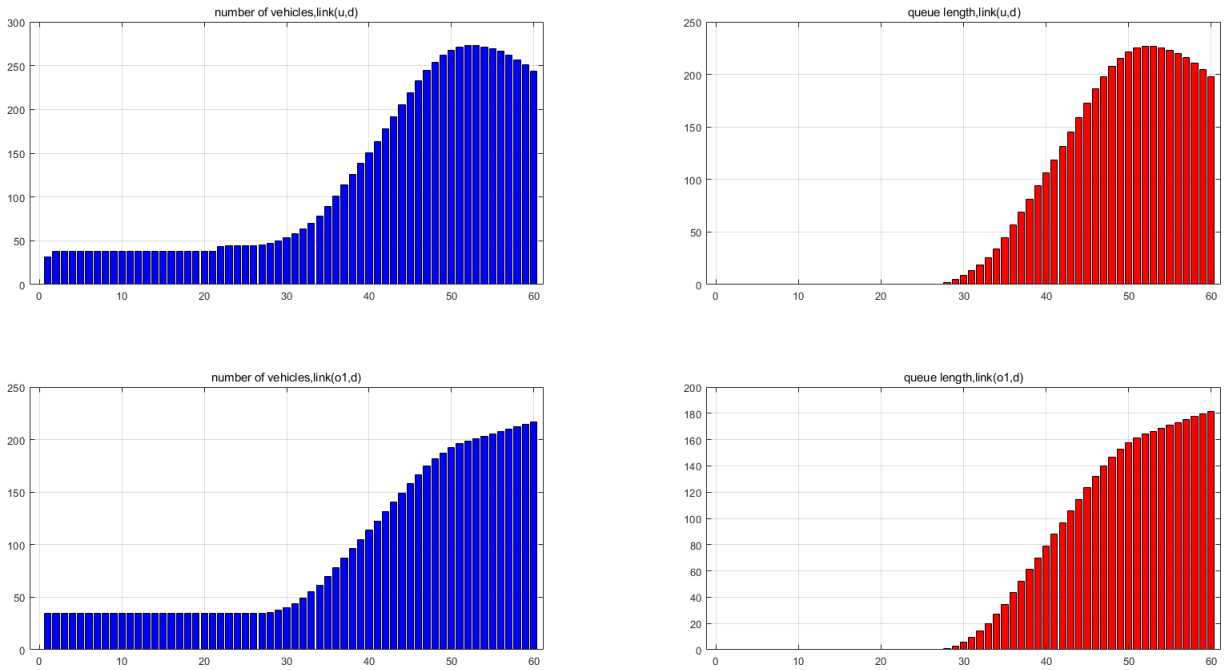


Figure 6: traffic situation perpendicular + golden section 15s

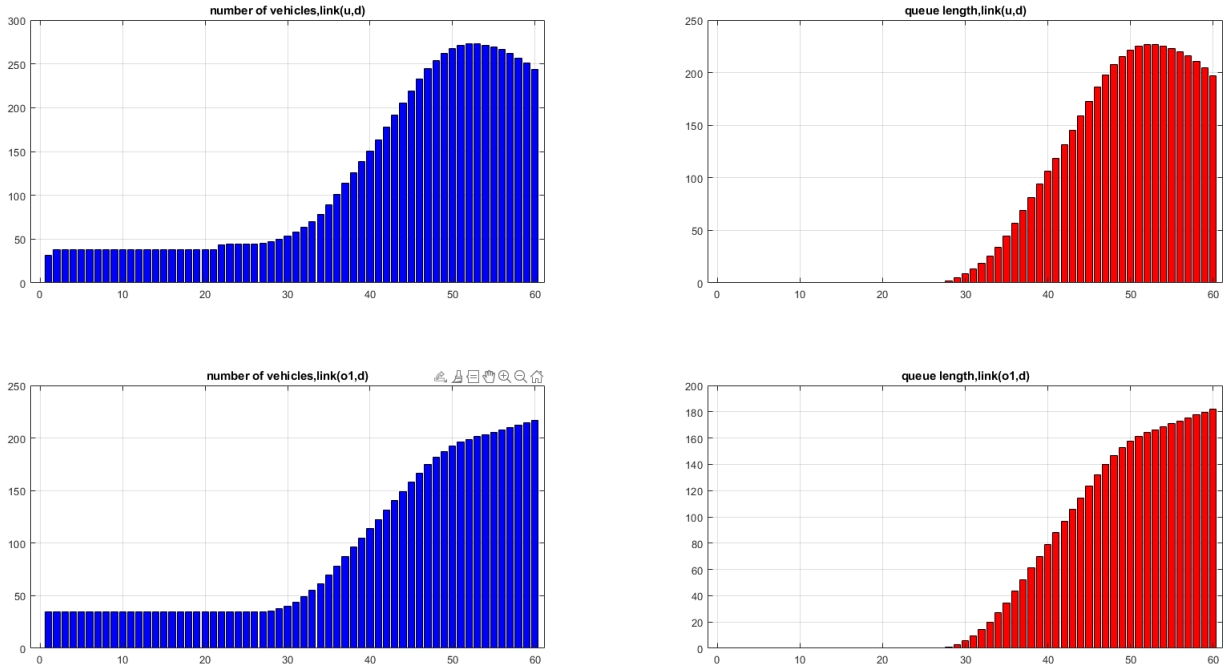


Figure 7: traffic situation perpendicular + golden section 45s

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.

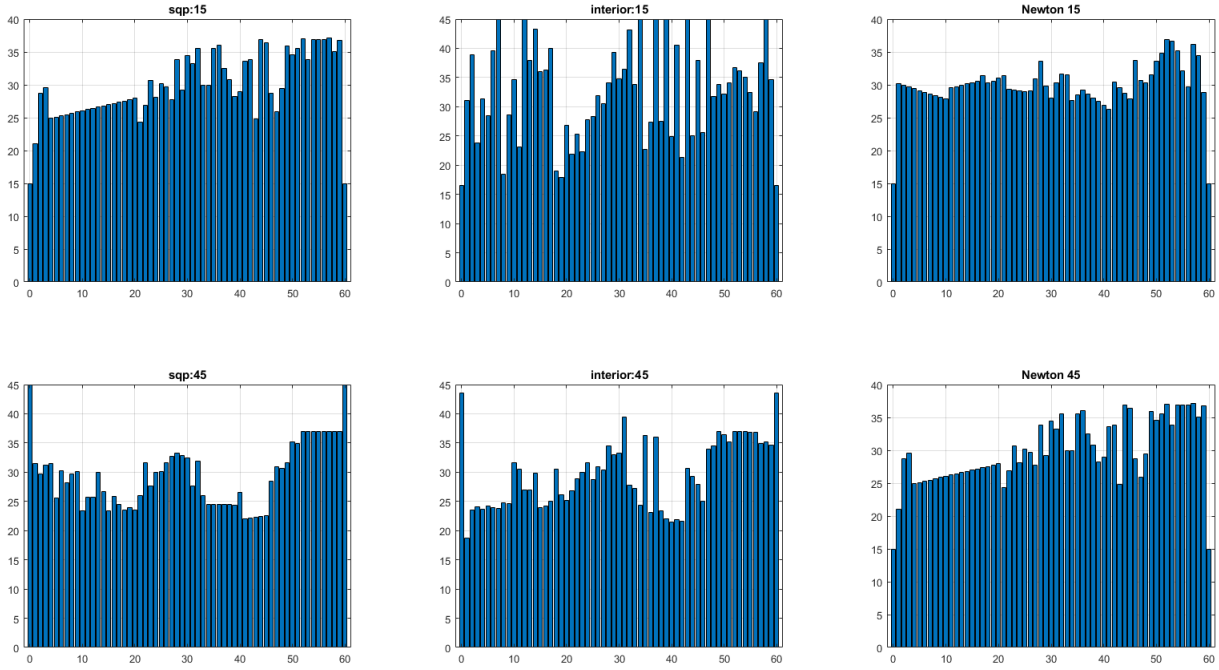


Figure 8: Input of SQP,Interior-Point and Quasi-Newton Method 45s

The two optimal inputs shown above decrease $g_d(k)$ when k is small, and increase $g_d(k)$ when k is big. This is reasonable because when k is small, link(o_1, d) has more flow than link(u, d) and the pressure of link(u, d) increasing as k goes on. And the changes in inputs make the value and duration of peak of link(u, d) decrease. Although the side effect is the increasing of peak of link(o_1, d), TTS is 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. This 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 .

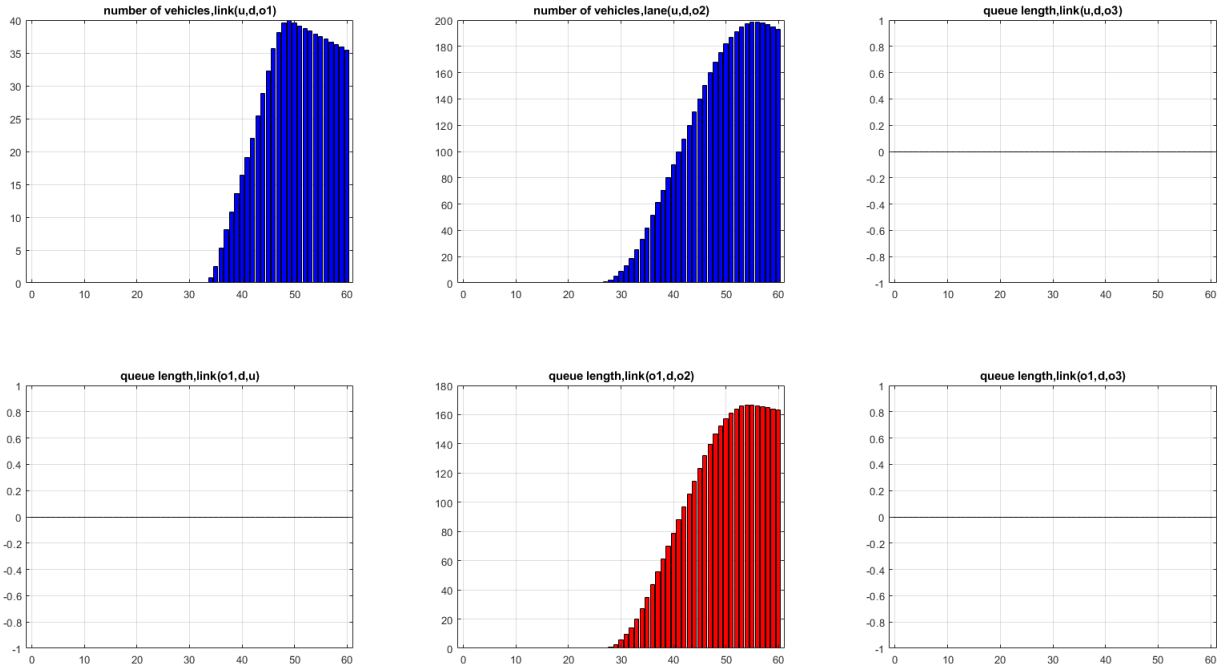


Figure 9: Queue each lane no control

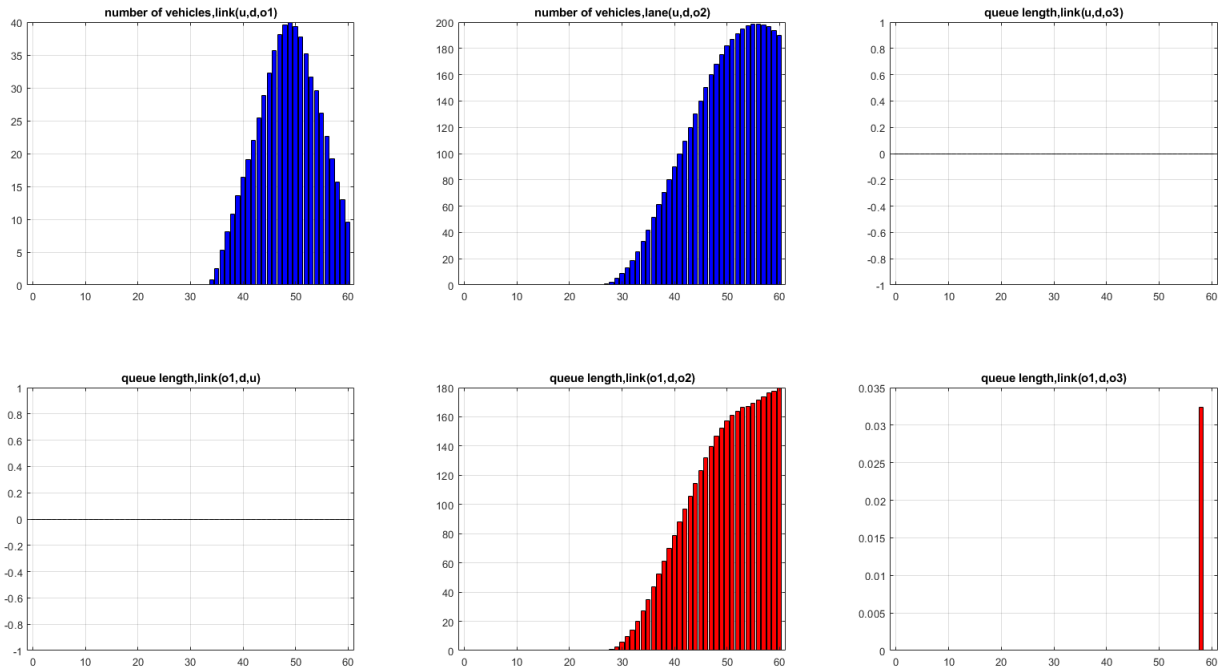
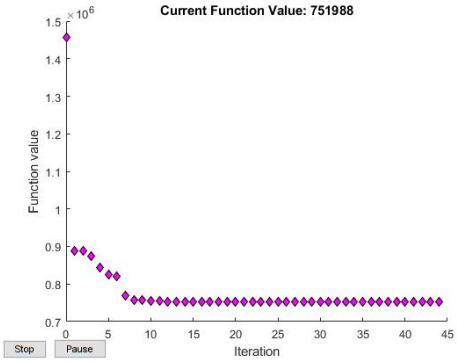


Figure 10: Queue each lane SQP 45s

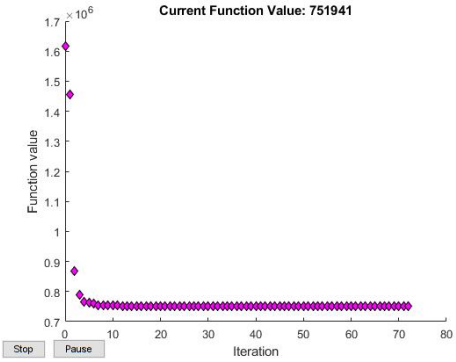
The optimal input makes the peak of link(u,d) drop faster while a little more pressure rejects into link(o_1, d). After time step 40, more flow enter link(u,d), the optimal input does have positive effect.

2. Process Compare

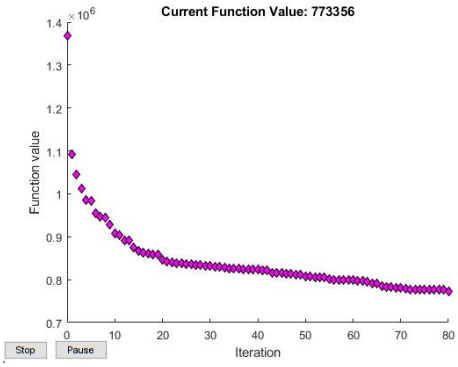
   Figure 11 shows the optimization process of Existing Matlab tools algorithm.



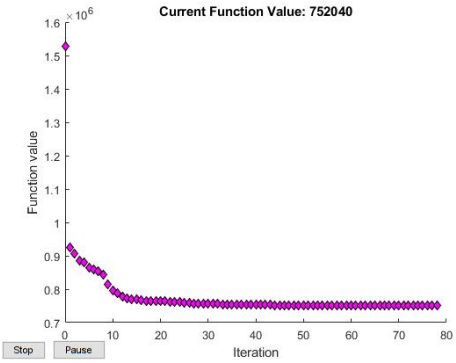
(a) SQP:15



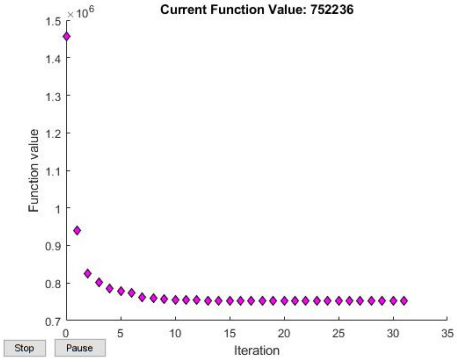
(b) SQP:45



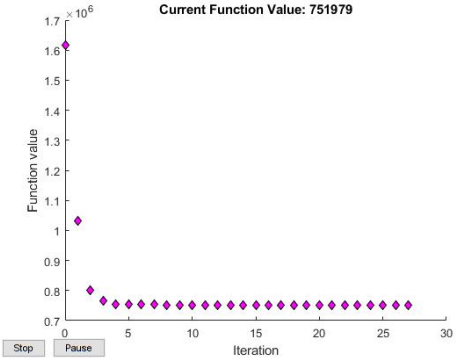
(c) Interior:15



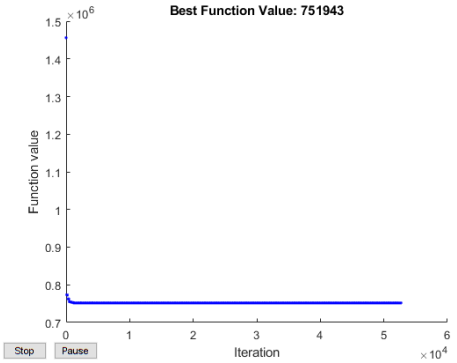
(d) Interior:15



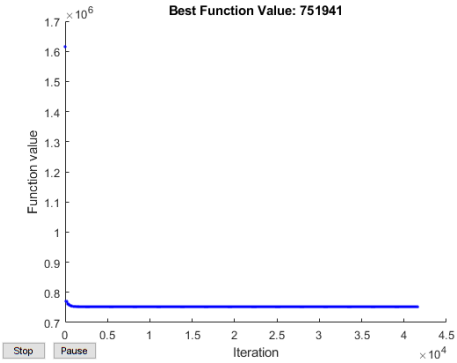
(e) Newton:15



(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 $g_d(k) = 30s, \forall k$

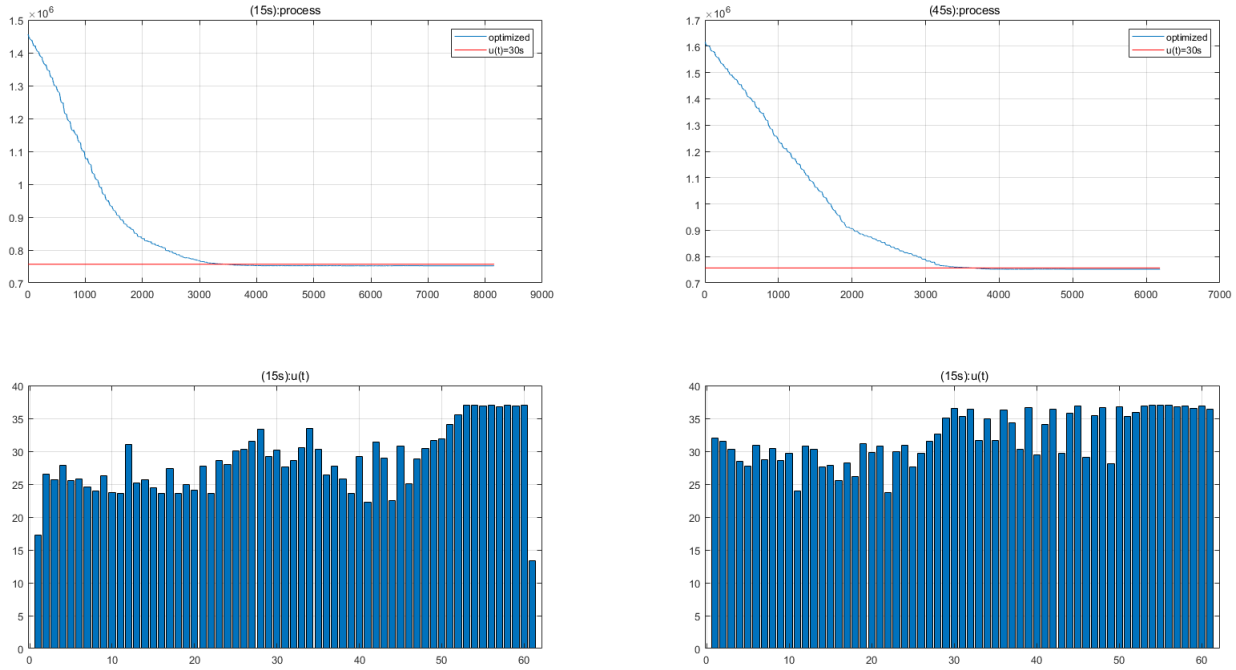


Figure 12: Perpendicular + Golden Section + Annealing

Under the parameters we set, it is obviously that the SQP optimization method has the higher speed to get to its optimal solution than Interior Methods. Although simulated annealing method has the most iteration cycles, the result obtained from two different starting points are the same. 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 is 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 are selected 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 3 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)

```
for i = 1; i ≤ N_g; i++ do
    for k = 1;k ≤ N_m; k++ do
        Calculate the TTS.
    end for
    Select Parents → the larger TTS, the less probability;
    create off-spring: 1-point cross-over;
    mutation flip bits with probability P_m;
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:

$$\begin{aligned} &gene = \{g_0 \quad g_1 \quad g_2 \quad \cdots, g_{60}\} \\ &\text{where } g_i \in [1,7] \text{ and } g_i \in \mathbb{Z} \end{aligned} \tag{8}$$

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 4: Genetic Optimization Algorithm Parameters

In a problem with minimize objective function, in order to make sure individuals 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 4 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

```
choose possible parents (the individual with the N_p smallest TTS)
for i = 1;i ≤ length(TTS);i++ do
    if individual i is able to become a parent then
        weight(i) ← TTS(i)/sum(TTS of possible parents);
    else
        weight(i)=0
    end if
    normalize weight;
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 5: 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.

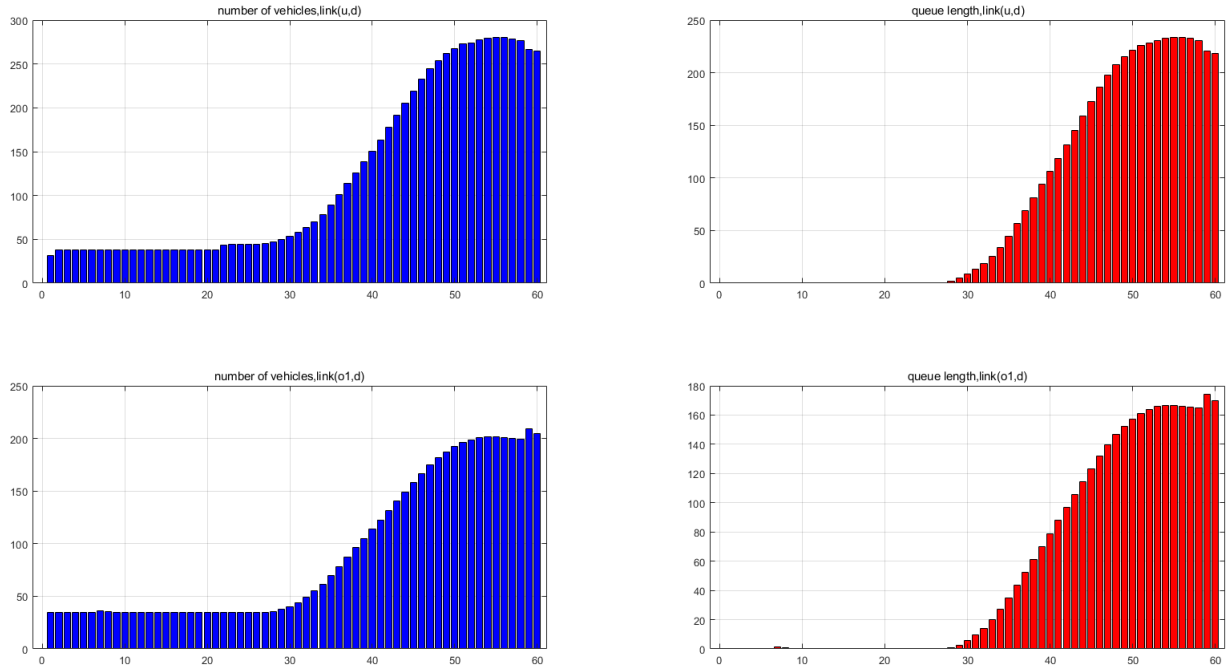


Figure 13: traffic situation genetic algorithm

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 5 Perpendicular Search Method(with integer step size).

Input: stopping criteria: *ther*
Output: Initialization: randomly generate start point ;

while diffrence $<$ *ther* **do**
  for $i = 1; i \leq 61; i++$ **do**
    direction $\leftarrow$ zeros(1,61);
    direction(i) $\leftarrow$ 1;
    for each s $\in [15, 20, 25, 30, 35, 40, 45]$ **do** choose best step
    end for
  end for
  calculate difference;
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.

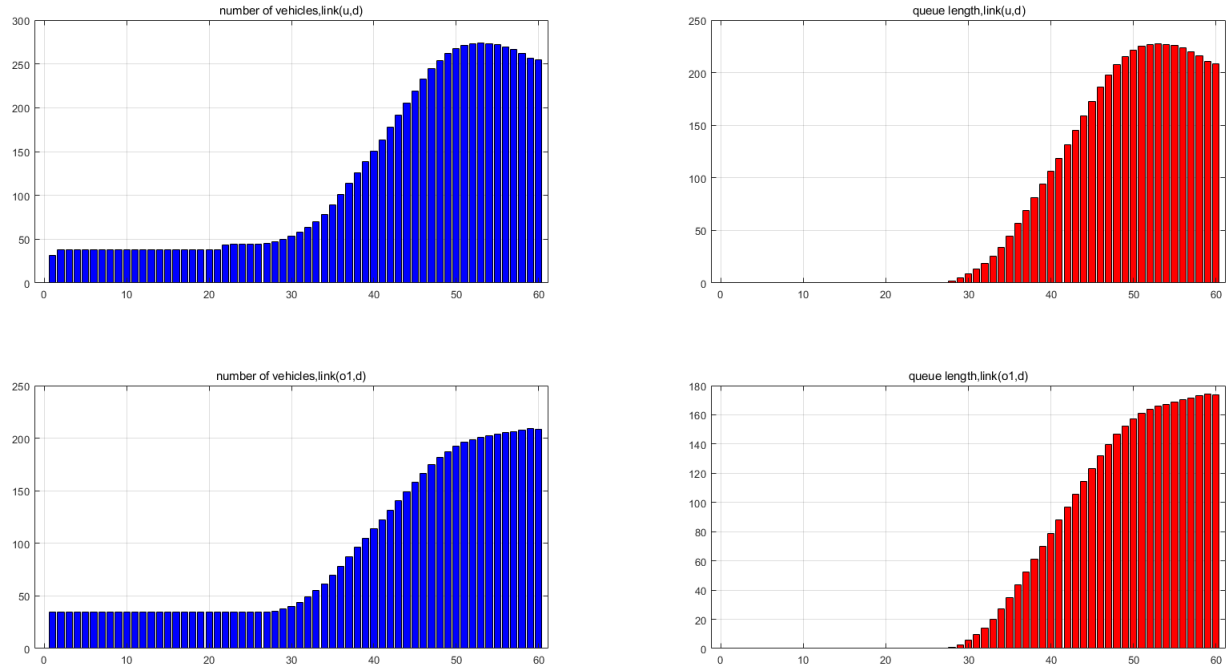


Figure 14: traffic situation perpendicular search algorithm

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

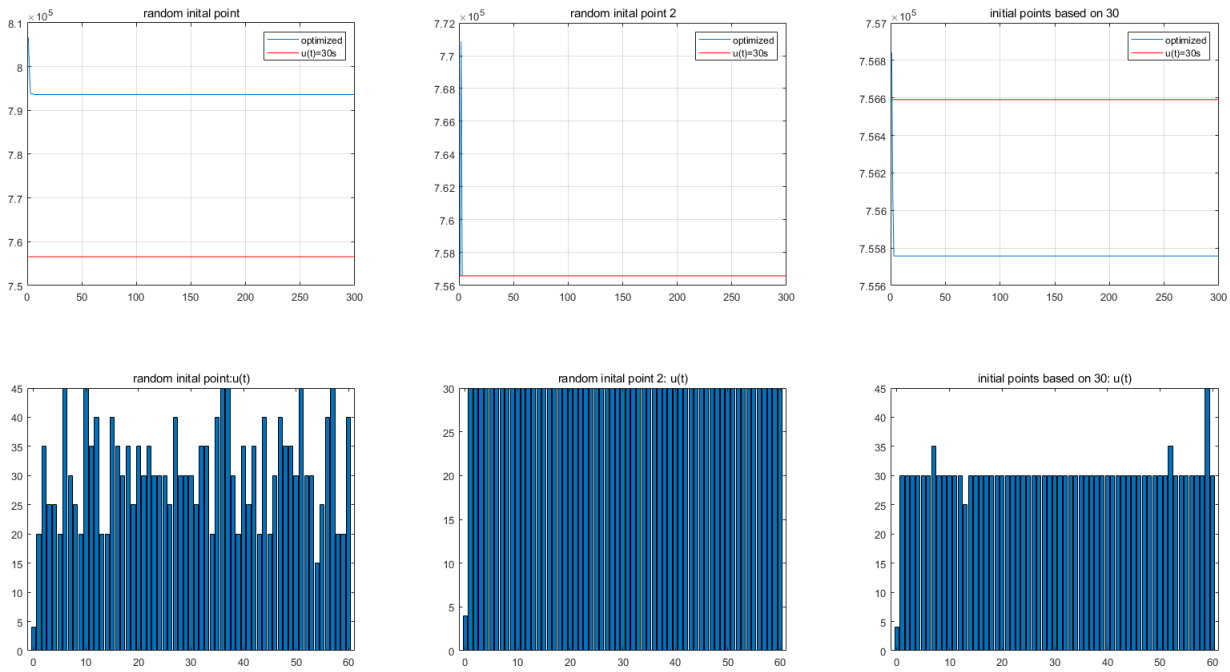


Figure 15: Genetic Algorithm results

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

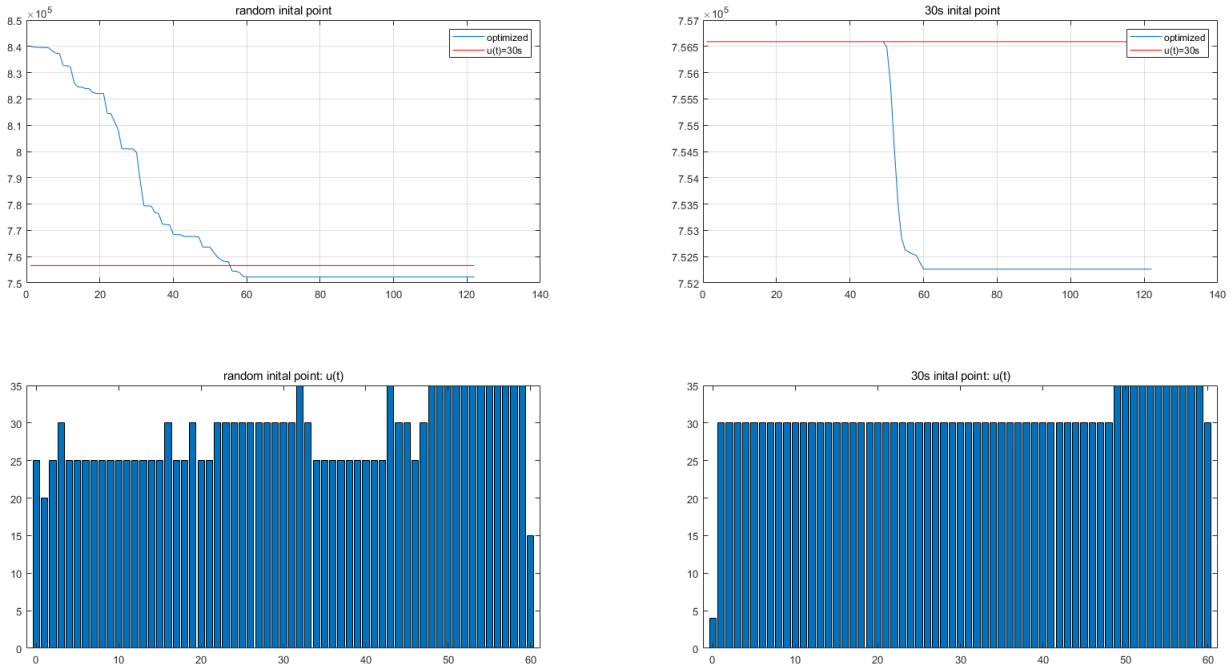


Figure 16: Perpendicular Search Algorithm results

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 in 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 from the perspective of global optimization, it is still possible to produce better results in later k.

A potential improvement way is to use some tricks of simulated 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 close to the moment with high flow pressure, we can adjust the threshold to increase the possibility to accept such changes.