Delft University of Technology

SC42056 - Optimization for Systems and Control

Nonlinear Programming Assignment

Fères Hassan [*]

Emiel Legrand [†]

October 28, 2020

[*]4362152 - f.p.hassan@student.tudelft.nl
[†]4446100 - e.b.legrand@student.tudelft.nl

Contents

Table 1: Group-specific parameters used in the model.

E1	E2	E3
2	5	2

1 State-space model

The model that predicts the number of vehicles on link (u, d) and link $(o1, d)$ for the next time step $k + 1$ will be defined as a discrete-time state-space model of the form:

$$\begin{cases} x(k+1) = f(x(k), u(k)) \\ y(k) = g(x(k), u(k)) \end{cases} \tag{1}$$

1.1 Definition of the states

The overall system can be viewed as a parallel system interconnection of the systems of the two individual links with a common output and input. However, since the traffic flow coming from link (u, d) and link (o_1, d) take turns (due to the traffic signal) going into their shared links (d, o_2) and (d, o_3), we can assume links (u, d) and (o_1, d) are two separate systems whose states don't interact. As such, the generalized model for one link will be derived first, after which the two links are simply combined into the overall model.

The state vector for a single link contains the number of cars in every queue (i.e. cars waiting at the junction to leave for in a certain direction), with the specific order defined in eq. (2).

$$x \text{ (general structure per link)} = \begin{bmatrix} \text{Queue for cars turning left} \\ \text{Queue for cars going straight} \\ \text{Queue for cars turning right} \\ \text{Total number of cars in the link} \end{bmatrix} \tag{2}$$

Hence, this results in the state definition

$$x_{u,d} = \begin{bmatrix} q_{u,d,o_1} \\ q_{u,d,o_2} \\ q_{u,d,o_3} \\ n_{u,d} \end{bmatrix} \quad \text{and} \quad x_{o_1,d} = \begin{bmatrix} q_{o_1,d,o_2} \\ q_{o_2,d,o_3} \\ q_{o_1,d,u} \\ n_{o_1,d} \end{bmatrix} \quad \text{combined as} \quad x = \begin{bmatrix} x_{u,d} \\ x_{o_1,d} \end{bmatrix} \tag{3}$$

1.2 Deriving the state-space model for link (u, d)

The model itself is quite straightforward as it consists in summing the net inflow (that is, the actual inflow subtracted by the outflow) of cars and multiplying it by the time step to get the difference between two subsequent states. This, colloquially called 'reservoir-based', approach is formalized as follows (both the three queues and the total number of cars in the link are so-called reservoirs):

$$x_{u,d}(k+1) = x_{u,d}(k) + c \begin{bmatrix} \alpha^{\text{arrive}}_{u,d,o_1}(k) - \alpha^{\text{leave}}_{u,d,o_1}(k) \\ \alpha^{\text{arrive}}_{u,d,o_2}(k) - \alpha^{\text{leave}}_{u,d,o_2}(k) \\ \alpha^{\text{arrive}}_{u,d,o_3}(k) - \alpha^{\text{leave}}_{u,d,o_3}(k) \\ \alpha^{\text{enter}}_{u,d}(k) - \alpha^{\text{leave}}_{u,d}(k) \end{bmatrix} \tag{4}$$

The value for the number of cars that enter link (u,d), $\alpha^{\text{enter}}_{u,d}(k)$ is considered as a (disturbance) input. However, because the behavior of the disturbance is known a priori, it can be incorporated completely into the model instead of being a disturbance in the classic sense. The total number of cars that leave the link is modeled as a minimum operation over several possibilities, each representing a certain scenario:

$$\alpha^{\text{leave}}_{u,d}(k) = \sum_{o \in O_{u,d}} \alpha^{\text{leave}}_{u,d,o} = \sum_{o \in O_{u,d}} \min \left\{ \frac{\mu_{u,d,o} g_d(k)}{c}, \frac{q_{u,d,o}(k)}{c} + \alpha^{\text{arrive}}_{u,d,o}(k), \frac{C_{d,o}(k)}{c} \right\} \tag{5}$$

These scenarios are, in that order:

1. The outflow is limited by the saturation rate of the link, i.e. the maximal number of cars is leaving over the duration of the green light.
2. The outflow is limited by the number of cars that where in or just arrived at the queue.
3. The downstream queue is saturated and limits the number of cars going in that direction.

Clearly, the $\min\{\cdot\}$ operator selects whichever scenario is the most limiting.

The cars that enter the link (governed by $\alpha^{\text{enter}}_{u,d}(k)$) spend some time on the road before they will eventually end up in one of the queues to continue their journey in one of the three directions of the downstream junction. The time it takes for them to reach the end of the link (or rather, the end of the queue, since this is taken into account) is modeled as follows:

$$\alpha^{\text{arrive}}_{u,d}(k) = \frac{c - \gamma(k)}{c} \alpha^{\text{enter}}_{u,d}(k - \tau(k)) + \frac{\gamma(k)}{c} \alpha^{\text{enter}}_{u,d}(k - \tau(k) - 1) \tag{6}$$

with

$$\tau(k) = \text{floor} \left\{ \frac{\left(C(k) - \left[\begin{smallmatrix} 1 & 1 & 1 & 0 \end{smallmatrix}\right] x(k)\right) l_{\text{veh}}}{N^{\text{lane}}_{u,d} v^{\text{free}}_{u,d} c} \right\}$$

and

$$\gamma(k) = \text{rem} \left\{ \left(C(k) - \left[\begin{smallmatrix} 1 & 1 & 1 & 0 \end{smallmatrix}\right] x(k)\right) l_{\text{veh}} N^{\text{lane}}_{u,d} v^{\text{free}}_{u,d},\ c \right\}$$

where $\alpha^{\text{arrive}}_{u,d}(k)$ is the flow of cars arriving at the queues at time step k. These expressions are an inevitable result of the inherent nature of a discrete-time system — it can only handle delays of an integer number of time steps. To cope with this issue, a weighted sum distributes the cars that would 'fall in between' two time steps between those subsequent time delays.

The use of the remainder rem$\{\cdot\}$ allows for a more refined distribution, allocating relatively more cars to the integer time delay that is the closest approximation to the actual one.

Given the total number of cars arriving at the queue $\alpha_{u,d}^{\text{arrive}}(k)$, a certain fraction $\beta_{u,d,o}$ goes in each of the three queues. With some abuse of notation, we will let $\alpha_{u,d}^{\text{arrive}}$ from now represent the vector of arriving cars per lane, which can be achieved by multiplying the total number with the β vector. As such, the expression for the arriving cars at each queue can be written in vector format:

$$\alpha_{u,d}^{\text{arrive}}(k) = \begin{bmatrix} \alpha_{u,d,o_1}^{\text{arrive}} \\ \alpha_{u,d,o_2}^{\text{arrive}} \\ \alpha_{u,d,o_3}^{\text{arrive}} \end{bmatrix} = \begin{bmatrix} \beta_{u,d,o_1} \\ \beta_{u,d,o_2} \\ \beta_{u,d,o_3} \end{bmatrix} \begin{bmatrix} \frac{c-\gamma(k)}{c} & \frac{\gamma(k)}{c} \end{bmatrix} \begin{bmatrix} \alpha_{u,d}^{\text{enter}}(k-\tau(k)) \\ \alpha_{u,d}^{\text{enter}}(k-\tau(k)-1) \end{bmatrix} \tag{7}$$

Combining eqs. (4), (5) and (7) the state transition equation for link (u,d) can be constructed. The input $u(k)$ is the time length of green light at every cycle and also the only input of the system. All other variables are already known a priori. Note that the the lanes turning right are not influenced by the traffic lights; so e.g., $g_{u,d,o_3}(k) = u(k) = c$.

$$\begin{aligned} x_{u,d}(k+1) = & f_{u,d}(x_{u,d}(k), u(k)) = \\ & x_{u,d}(k) + c \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \alpha_{u,d}^{\text{arrive}}(k) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ c \end{bmatrix} \alpha_{u,d}^{\text{enter}}(k) \\ & - c \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \min \left\{ \frac{1}{c} \begin{bmatrix} \mu_{u,d,o_1} & 0 & 0 \\ 0 & \mu_{u,d,o_2} & 0 \\ 0 & 0 & \mu_{u,d,o_3} \end{bmatrix} \begin{bmatrix} u(k) \\ u(k) \\ c \end{bmatrix}, \right. \\ & \left. \frac{1}{c} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x(k) + \alpha_{u,d}^{\text{arrive}}(k), \ \frac{1}{c} \begin{bmatrix} C_{d,o_1}(k) \\ C_{d,o_2}(k) \\ C_{d,o_3}(k) \end{bmatrix} \right\} \end{aligned} \tag{8}$$

where the min $\{\cdot\}$ operator acts in horizontal direction (i.e. the operation results in a column vector of length three).

1.3 Model for link (o_1, d)

The model for link (o_1, d) is completely analogous to eqs. (7) and (8) and is listed below for completeness.

$$\alpha_{o_1,d}^{\text{arrive}}(k) = \begin{bmatrix} \alpha_{o_1,d,o_2}^{\text{arrive}} \\ \alpha_{o_1,d,o_3}^{\text{arrive}} \\ \alpha_{o_1,d,u}^{\text{arrive}} \end{bmatrix} = \begin{bmatrix} \beta_{o_1,d,o_2} \\ \beta_{o_1,d,o_3} \\ \beta_{o_1,d,u} \end{bmatrix} \begin{bmatrix} \frac{c-\gamma(k)}{c} & \frac{\gamma(k)}{c} \end{bmatrix} \begin{bmatrix} \alpha_{o_1,d}^{\text{enter}}(k-\tau(k)) \\ \alpha_{o_1,d}^{\text{enter}}(k-\tau(k)-1) \end{bmatrix} \tag{9}$$

$$
\begin{aligned}
x_{o_1,d}(k+1) = & f_{o_1,d}(x_{o_1,d}(k),u(k)) = \\
& x_{o_1,d}(k) + c\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\\0&0&0\end{bmatrix}\alpha^{\text{arrive}}_{o_1,d}(k) + \begin{bmatrix}0\\0\\0\\c\end{bmatrix}\alpha^{\text{enter}}_{o_1,d}(k) \\
& - c\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\\1&1&1\end{bmatrix}\min\left\{\frac{1}{c}\begin{bmatrix}\mu_{o_1,d,o_2}&0&0\\0&\mu_{o_1,d,o_3}&0\\0&0&\mu_{o_1,d,u}\end{bmatrix}\begin{bmatrix}c-u(k)\\c-u(k)\\c\end{bmatrix},\right. \\
& \left.\frac{1}{c}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\end{bmatrix}x(k)+\alpha^{\text{arrive}}_{o_1,d}(k),\ \frac{1}{c}\begin{bmatrix}C_{d,o_2}(k)\\C_{d,o_3}(k)\\C_{d,u}(k)\end{bmatrix}\right\}
\end{aligned}
\tag{10}
$$

Note that the green light time of link $(o1,d)$ is directly related to the green light time of link (u,d) as $g_{o1,d}(k) = c - g_{u,d}(k)$. The model for $\alpha^{\text{arrive}}_{o_1,d}(k)$ is identical to eq. (7) with the proper parameters for link (o_1,d) (as is the calculation for $\gamma(k)$ and $\tau(k)$).

1.4 Combined model

The combined state-space model is then

$$
x(k+1) = \begin{bmatrix}x_{u,d}(k+1)\\x_{o_1,d}(k+1)\end{bmatrix} = \begin{bmatrix}f_{u,d}(x_{u,d}(k),u(k))\\f_{o_1,d}(x_{o_1,d}(k),u(k))\end{bmatrix} \tag{11}
$$

The output (TTS) of the state-space model is expressed as the sum of the number of vehicles on link (u,d) and link $(o1,d)$ multiplied by the time step, which is achieved by

$$
y(k) = c\begin{bmatrix}0&0&0&1&0&0&0&1\end{bmatrix}x(k) \tag{12}
$$

2 Formulating the optimization problem

To find the green light time at node d for every cycle that minimizes the Total Time Spent (TTS) by drivers in the link (u,d) and (o_1,d), we formulate the following optimization problem. We consider the green time $g_d(k)$ as continuous variables that are constrained in $[15\,\mathrm{s},\ 45\,\mathrm{s}]$ and we assume the system is empty at $k=0$.

$$\begin{aligned} \underset{g_{d,1},\cdots,g_{d,N}}{\text{minimize}} \quad & \sum_{k=1}^{N} c\,\left(n_{u,d}(k)+n_{o_1,d}(k)\right)=\sum_{k=1}^{N} y(k) \\ \text{subject to} \quad & 15s \leq g_{d,k} \leq 45s \qquad\qquad k=1,\dots,N \end{aligned} \tag{13}$$

Furthermore, fig. 1 shows the time-varying parameters that are incorporated in the state-space model; as mentioned in section 1 they are the capacity of the downstream links and the flow of traffic introduced in each link over time..

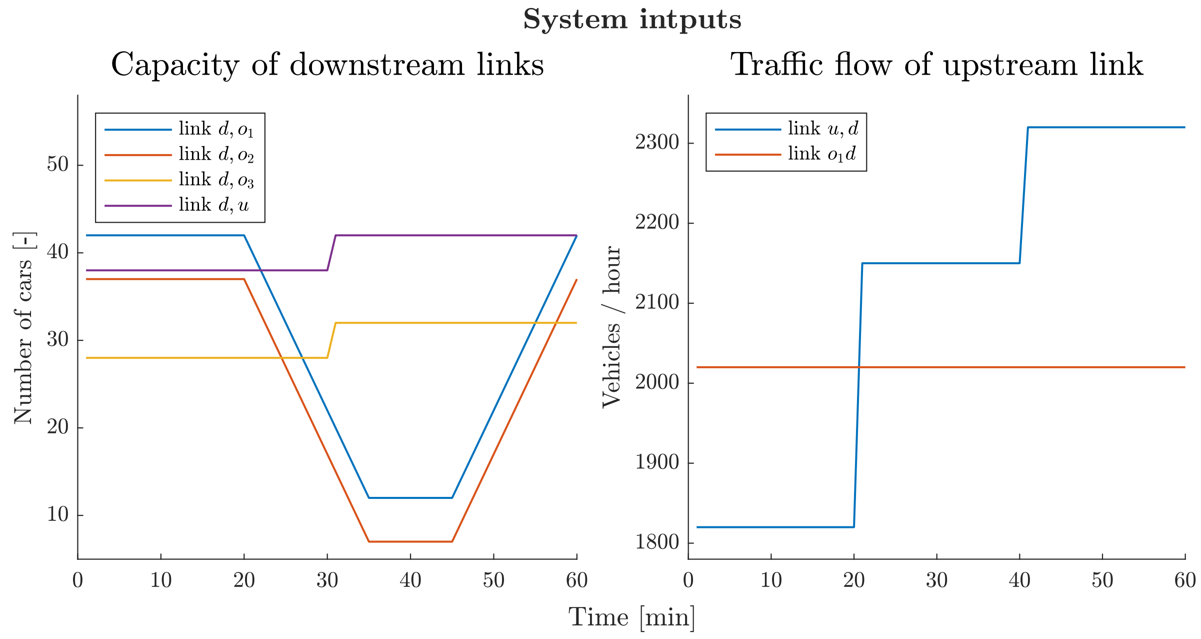

Figure 1: Overview of the time-varying inputs for the model that are known a priori: the capacity of the downstream link and the incoming trafic flow for each link.

3 Algorithm selection

In order to select a suitable optimization method one must first understand the type of design space that is being used. Generally, it is difficult to know the characteristics of a design space without first exploring it. First some formal problem characteristics can be determined to narrow the scope of our search. Subsequently, a comparison between optimization algorithms will be made to determine the best suitable algorithm.

3.1 Problem characteristics

Number of variables

Since a different green light time can be chosen at every time step, the optimization variable u is a 60-dimensional vector. As already mentioned, any continuous value for u can be chosen as long as it satisfies the constraints.

Linearity

From inspection of the state-transition it is clear that it is nonlinear; at least due to the presence of the nonlinear operators $\min\{\cdot\}$, $\text{floor}\{\cdot\}$ and $\text{rem}\{\cdot\}$.

On the other hand, the constraints are simply upper and lower bounds for u which can easily be written as a matrix inequality and therefore linear.

Convexity

Again, due to the presence of the non-convex $\min\{\cdot\}$, $\text{floor}\{\cdot\}$ and $\text{rem}\{\cdot\}$ operators the resulting cost function will also not be convex. Please note that although the $\text{floor}\{\cdot\}$ and $\text{rem}\{\cdot\}$ do not act on the input directly in the state transition, they do so indirectly due to the influence of previous states on the present one. Since the constraints are linear, they are necessarily also convex.

Cost of evaluation

On average, one evaluation of the cost function took about 6.33×10^{-3} s. As such, approximating the gradient or Hessian will probably pay off in terms of overall computation time.

3.2 Comparison of optimization algorithms

Based on this information a suitable algorithm can be chosen. Evidently, linear, quadratic and convex programming are ruled out. To deal with the constraints, one has two options

- Use an optimization routine that can handle constraints
- Transform the constrained problem to an unconstrained problem using penalty or barrier functions.

The second option has inherent disadvantages: the problem can become ill-conditioned during the optimization process and the selection of the 'steepness' of the penalty/barrier functions is often a trial-and-error procedure. Because the Matlab Optimization Toolbox has multiple options available to handle constraints, the first option was chosen over the second one.

Based on these choices, a few options remain. First, one could opt for Sequential Quadratic Programming or the gradient projection method. Because the problem is nonlinear non-convex (and the high dimensionality of the design variable) many local minima may exist. Therefore, these methods could be applied in a multi-start fashion. The advantage of choosing this method is that they use gradient and possibly Hessian information which could possibly result in faster convergence to (local) minima.

Secondly, one could choose a method that is specifically suited for global optimization of problems with many local minima, such as Simulated Annealing or a Genetic Algorithm, both of which are included in Matlab.

It was not immediately clear which of these methods was better suited. Therefore, a comparison was made between three methods (to the authors' knowledge gradient projection is not directly implemented in Matlab):

- `fmincon` using Sequential Quadratic Programming
- `fmincon` using the Interior-point method (because it is the default option in Matlab).
- `simulannealbnd` i.e. Simulated Annealing

The results of this comparison are documented in section 5.

4 Integer optimization

Another optimization problem has to be formulated if we consider the green light time to be limited to a discrete set of possibilities. Even if we could implement these integer constraints in the `fmincon` function, it would still search for a continuous space which is not efficient given the problem statement. In this case the problem has to be approached as a nonlinear optimization problem with integer constraints, which will be solved for an integer index which corresponds to a value in the set of possible solutions.

$$\begin{aligned} \underset{g_{d,1},\cdots,g_{d,N}}{\text{minimize}} \quad & \sum_{k=1}^{N} c\left(n_{u,d}(k) + n_{o_1,d}(k)\right) = \sum_{k=1}^{N} y(k) \\ \text{subject to} \quad & 15s \leq g_{d,k} \leq 45s \qquad k = 1,\ldots,N \end{aligned} \tag{14}$$

Where

$$g_{d,k} \in \left\{15 \quad 20 \quad 25 \quad 30 \quad 35 \quad 40 \quad 45\right\} \tag{15}$$

For this problem a heuristic search technique can be used to directly deal with the integer optimization variables. Since we already know the problem has a lot of local minima from results of the other algorithms, a global search algorithm will be a suitable optimization method. The genetic algorithm implementation in Matlab exhibits the characteristics which are favorable for this problem; it can escape from local minima since a whole population of possible solutions is considered. The random mutation guarantees to some extent that a wide range of solutions will be explored.

5 Results

The results of all the optimization routines are shown in fig. 2 and a measure of their 'performance' (in terms of the relative difference in the resulting Total Time Spent) is shown in fig. 3 and fig. 4. From fig. 2 it is clear that all the algorithms and their varying initialization yield different results; it is therefore safe to conclude that the problem has a lot of local minima. The best solution was found by the Interior-point algorithm initialized at 45 s, although there is no reason to believe that this is due to the inherent nature of the Interior-point algorithm and probably coincidental.

Table 2 gives an overview of the relative optimization results in terms of the Total Time Spent. The largest reduction compared to the no-control case is 3.02%. This is not a major change due to the fact that the number of cars entering the system in the first 30 minutes do not put a lot of pressure on the system. Even the no-control case manages to handle the queues efficiently. Bluntly said, there is not a lot to optimize during this period. Since the traffic flow does not have a dramatic effect on the system so the green light time has only a minor impact on the cost function. Hence, the latter will not drive the algorithms to the same input variables, explaining the disparity between the algorithms in fig. 2.

In the second half of the simulation period the traffic flow increases gradually which appears to result in converging outcomes in fig. 2. The final 10 minutes, when the traffic flow is the highest, the algorithms in both the initialization cases are very close. The reason the genetic algorithm does not converge to the same result eventually is because of the limitations of the discrete time set. In particular, for both links the outgoing queues going to o_2 experience the most dramatic growth. This is somehow to be expected since both the saturation rate of this queue is the lowest and the capacity of the downstream queue is lower as well. From the solution it is clear all optimization methods tend to give link (u, d) a slightly higher priority then the other link, particularly during the final part of the simulation. The reason for this is evident, since the number of entering cars is also larger in that time interval.

If these analyses and assumptions are correct, the algorithms would converge more to the same solution if the traffic situation was more critical, and the green light time has a more dramatic effect on the cost function. To check these assumptions a test run was performed with higher $\alpha_{u,d}^{\text{enter}}(k)$ values. From fig. 5 it becomes clear this is indeed the case. Also a significant improvement in algorithm performance has been made. In the most optimal case a 25.33% lower Total Time Spent than in the no-control case is obtained.

A possible improvement could be made by using a multi-start method, preferably using one of the `fmincon` algorithms (they show comparable results to the global methods at a fraction of the run time). They could be initiated from many randomized initial guesses, after which the best overall solution is picked. Then still however, it cannot be proven to be the global optimum (to prove this would actually be dramatically more complex then the optimization itself).

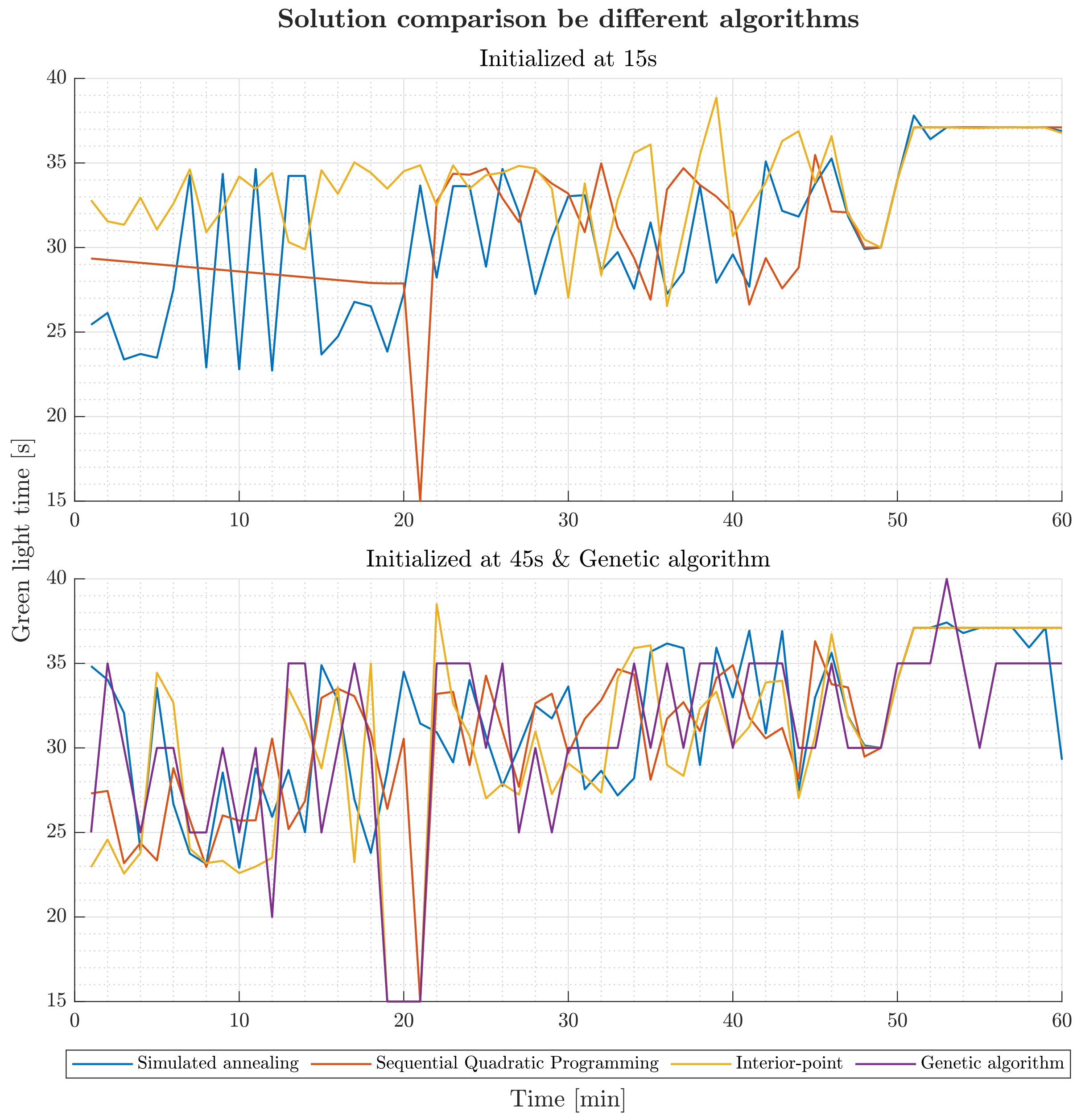

Figure 2: Overview of all the solutions resulting from various algorithms. The plot on top shows the results from the Simulated Annealing, SQP and Interior-point methods initialized at $u(k) = 15\,\mathrm{s}\ \forall k$. The lower plot shows the same results, but initialized at $u = 45\,\mathrm{s}\ \forall k$ together with the result from the integer programming problem using the Genetic algorithm.

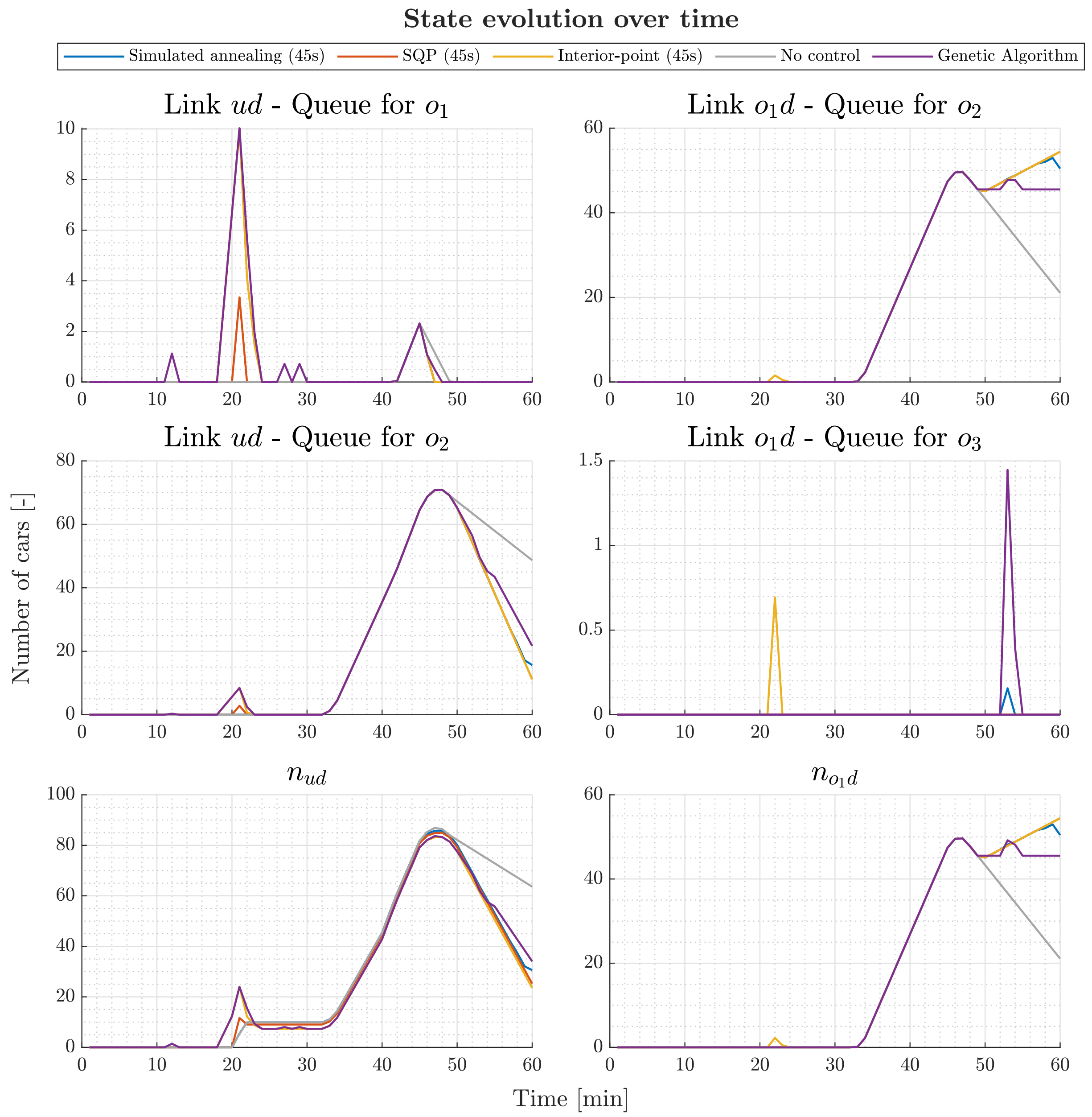

Figure 3: Evolution of the states over time, subject to the inputs corresponding to the lower plot of fig. 2 (i.e. all the solutions initialized at 45 s and the result from the integer optimization). From top to bottom are the left-turning queue, the queue going straight and the total number of cars. The queue for cars turning right was always empty for all simulations and therefore not very informative.

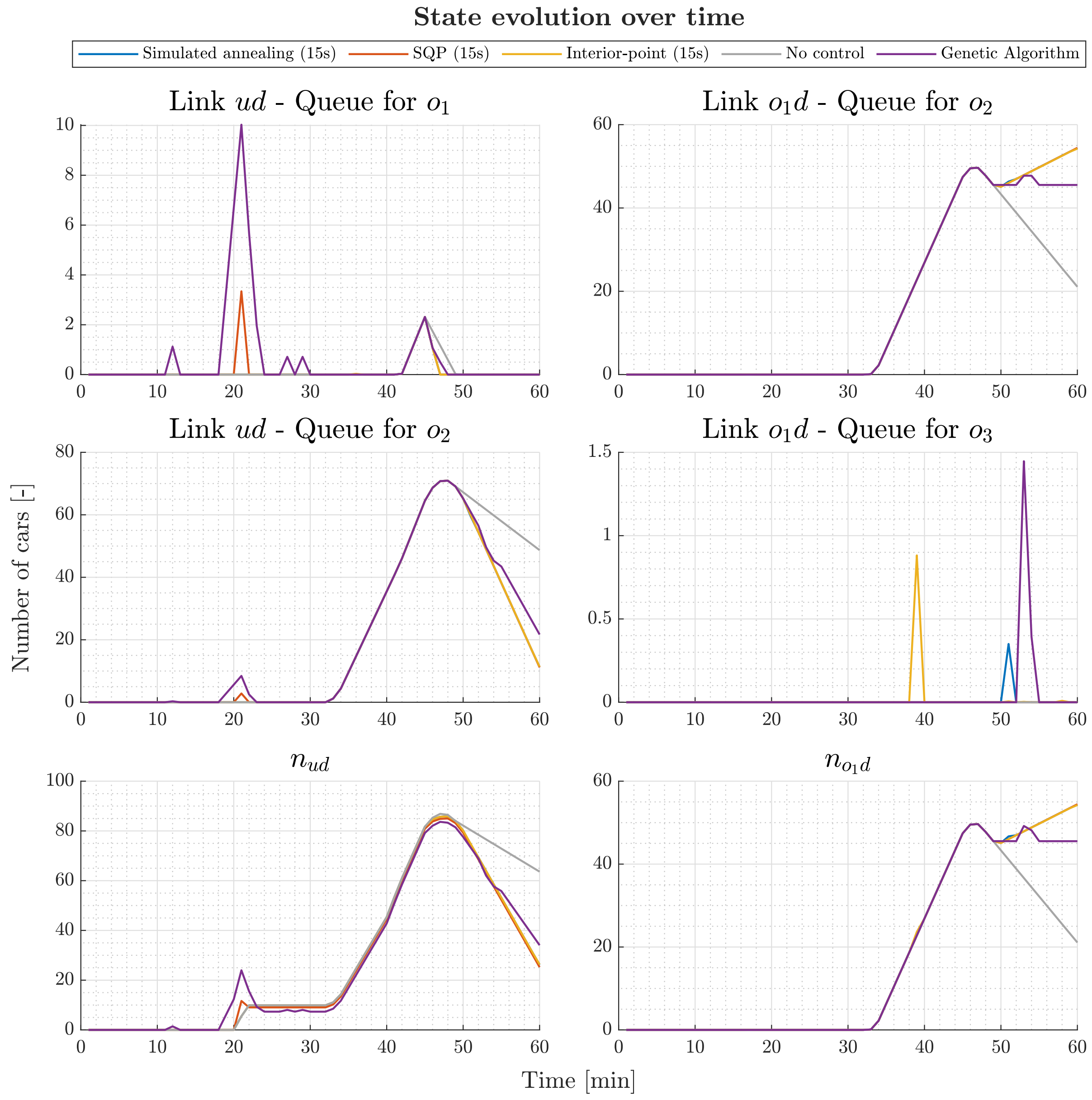

Figure 4: Similar to fig. 3, this figure shows the evolution of the states over time, subject to the inputs corresponding to the upper plot of fig. 2 (i.e. all the solutions initialized at 15 s. Again, the right-turning queue were left out because they were always empty.

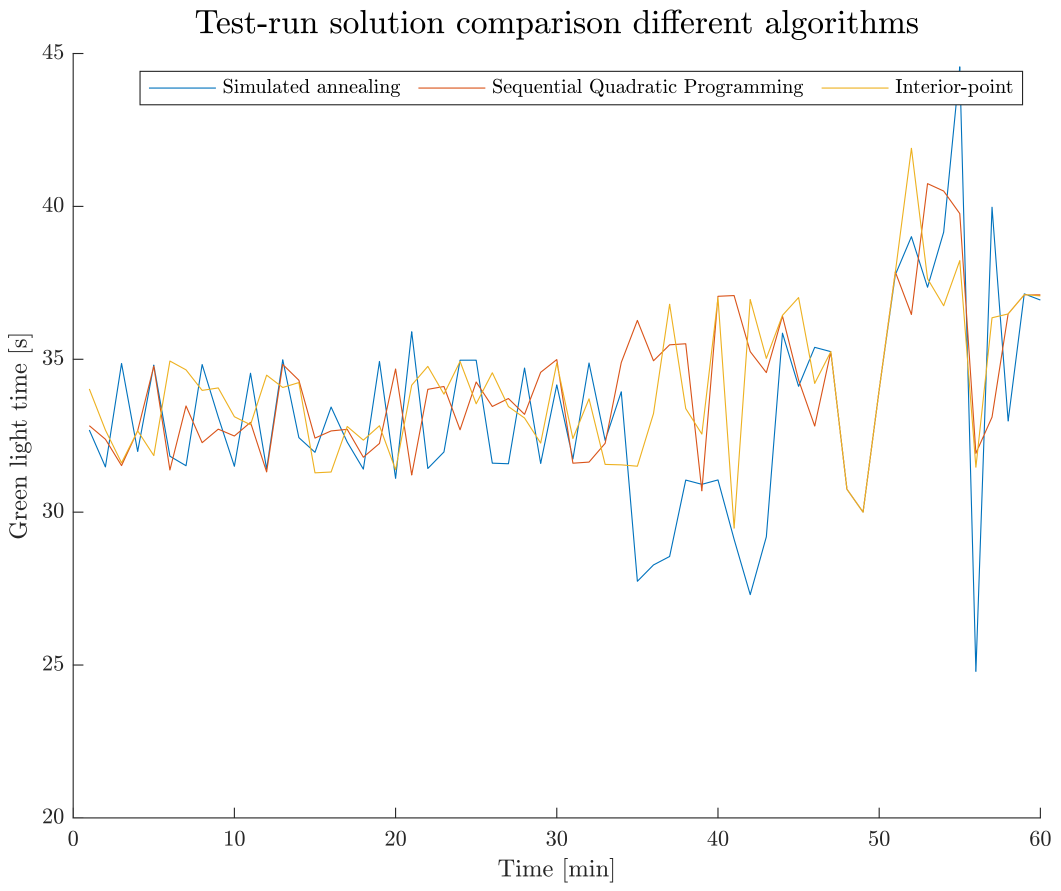

Figure 5: Overview of the test-run solutions of the various algorithms methods initialized at $u(k) = 15\,\mathrm{s}\ \forall k$, with $\alpha_{u,d}^{\mathrm{enter}}(k)$ set to 2500 veh/h for the whole period.

Table 2: Total Time Spent for the solution of three optimization algorithms, each both initialized at a green light time of 15 and 45 seconds for every time step. The best solution is shown for the interior-point algorithm initialized at 45 s, the other entries show difference in percentage with the no control case.

u_0	**SA**	**SQP**	**IP**	**GA**	**No control**
15 s	-0.99%	-2.02%	-0.96%	-2.38%	159 730 s
45 s	-0.97%	-2.02%	-3.02%		