Optimization Techniques for Multi Cooperative Systems MCTR 1021
Mechatronics Engineering
Faculty of Engineering and Materials Science
German University in Cairo

Optimal Control for Connected and Autonomous Vehicles at Signal-Free Intersections

By

Team 3

Michel Yakoub
Paula Fahmy
Amr Abdrabo
Youssef Abdelhady
Bishoy Essam

Course Team

Omar M. Shehata
Catherine Malak
Shaimaa El Baklish

November 12, 2021

This is to certify that:

(i) the report comprises only my original work toward the course project,

(ii) due acknowledgment has been made in the text to all other material used

Team 3
November 12, 2021

Contents

Chapter 1

Literature Review

In paper [1], they presented a vision-based infrastructure of a decentralized approach for the Intelligent Traffic System (ITS). They have modeled an intersection through the conflict matrix. The proposed optimized distributed algorithm is based on this matrix and it will provide a set of queues without conflict of collisions and the time needed to empty each queue. They also took into account the balance between the queue with heavy traffic and low traffic in order to avoid the problem of an empty queue with a green light.

In paper [2], they proposed a cooperative method for traffic signal optimization and vehicle speed control to improve the transportation efficiency and vehicle fuel economy both in macro traffic level and micro vehicle level. The simulation results showed that the cooperative method can effectively adjust the traffic signal timing according to the real time traffic condition and simultaneously produce optimal vehicle trajectory/speed profiles, which results in the improved transportation efficiency and vehicle fuel economy. In addition, the cycle length and communication range have significant effects on the performance of the algorithm.

In this paper [3], the authors were aiming to create a simulator to optimize traffic time management, so that the timers on each track have the intelligence to predict the right time, so that congestion at the intersection can be reduced by adding up to 15 seconds of green light from the previous time in the path of many vehicles, by using Fuzzy Mamdani logic They optimized traffic light control at intersections. The duration or green light period for each row can be optimized based on several real-time parameters, such as queue length in each row, number of vehicles going to the queue, vehicle speed, and width of each lane.

The goal of this research [4] is to offer a method for employing an optimization tool to reduce traffic congestion and crossing times on a road network. Agent-based modeling is a modeling style that manages to capture individuals and how they interact. The application of this method has been done in this paper using the AnyLogic simulation tool.here the used two traffic light configuration method Intersection's Lane connectors and Intersection's stop lines. We will find a decrease by 32.215 % in the mean time necessary to a car to cross the road network based on Intersection's Lane connector and 31.22 % based on Intersection's stop lines.

The authors in [5] developed Distributed Optimization and Coordination Algorithms (DOCA) for dynamic speed optimization of connected and autonomous vehicles in smart city networks to tackle the problem of traffic. After testing this technology in networks with eight, twenty, and forty intersections, The results estimated that DOCA-DSO provided solutions with at most 2.7% optimality gap. The developed methodology reduced the travel time by up to 14.8% and speed variation by 9.7–13.4% over different demand patterns compared to a no-speed harmonization strategy. Moreover, the average speed and the total number of completed trips were increased by up to 28.0% and 9.5% respectively in a network with 20 intersections and 500 study periods.

In this paper [6], they used a Distributed-Coordinated methodology for signal timing optimization in connected urban street networks. The algorithm controlled queue length and maximized intersection throughput (between 1% and 5% increase compared to the actuated coordinated signals optimized in VISTRO) and reduced travel time (between 17% and 48% decrease compared to actuated coordinated signals) in all cases. This study provide benchmark solution which have developed a linear formulation for DSO in urban street networks. The approach finds the optimal solution to the problem in medium size urban street network, however, its run-time increases with the size of the network.

This paper [7] addresses the problem of coordination between connected autonomous vehicles (CAVs) at signal-free intersections by a centralized methodology. A weighted sum of the aggregate energy consumption and traveling time of all CAVs is optimized. The OCP is solved using GPOPS-II which is a general-purpose MATLAB software for solving continuous optimal control problems. The paper provides numerical examples at the end.

This paper [8] provides a similar study to [7]. The difference is that [8] introduces multi-lane roads and allows the vehicles to change their direction inside the intersection region. This paper solves the OCP problem using Particle Swarm Optimization (PSO).

This study [9] formulated coordinated signal timing and speed optimization problem in urban street networks. The proposed algorithm in a network with twenty intersections. Compared to the Benders decomposition algorithm (benchmark), DOCA found solutions with at most 5.4% optimality gap. It was shown that the CSSO is more effective when the network is congested. In comparison to the signal timing optimization, DOCA-CSSO reduced the travel time by 0.5%, 1.1%, and 2.7% in the under-saturated, saturated, and over-saturated demand conditions, respectively. CSSO reduced the travel time, average delay, average number of stops, and average delay at stops.

This paper [10] builds coordinated signal control system based on the cooperative vehicle infrastructure technology. The ABC-SFL algorithm can effectively balance the global search and depth search opportunities of the algorithm based on the frog's three-jump operation. Along with improving the search coverage of the algorithm, this algorithm can also develop some areas of excellent frogs and reduce the blindness of the frogs' learning direction.

Chapter 2

Problem Formulation

There are four double-lane roads meeting at an intersection as shown in Figure 2.1. The area around the intersection is referred to as Control Zone (CZ) and extends L distance from the intersection boundary, while the center of the intersection is called Merging Zone (MZ). The length of the MZ is denoted by S and assumed to be $S < L$. For simplicity it is assumed that vehicles do not turn so that the only directions of movement allowed are $\mathcal{D}$ = {North-South (NS), South-North (SN), East-West (EW), West-East (WE)}. Each Connected Autonomous Vehicle (CAV), i, that is entering the CZ has an arrival time to the CZ, t_i^0, initial velocity, v_i^0 and direction of movement $d_i \in \mathcal{D}.t_i^m$ and t_i^f denote the required travel time of CAV i to arrive and exit the MZ respectively. It is considered that when $t_i^0 < t_j^0$, it is also true that $i < j$. N is denoted as the total number of arrivals, i.e, $i \in \mathcal{N} = \{1, \ldots, N\}$.

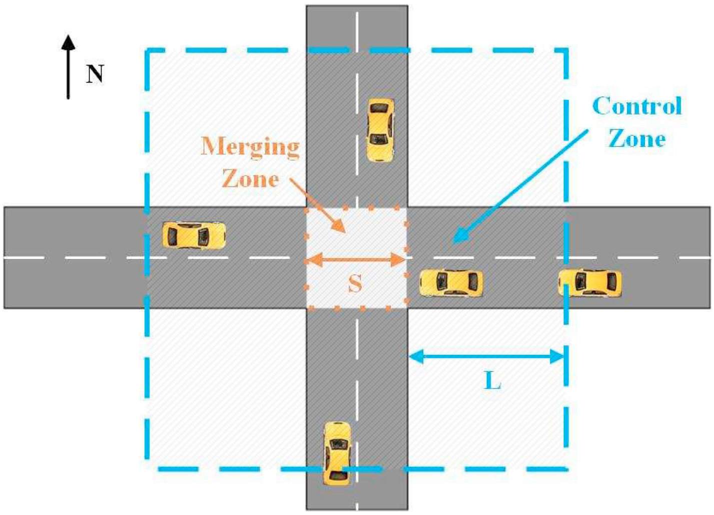

Figure 2.1: Scheme of autonomous intersection crossing with connected and autonomous vehicles

2.1 Decision variables

- Acceleration profile, $u(t)$ of each vehicle, i, from t_i^0 to $t_i^0 + t_i^m$

2.2 Objective functions

$$\text{minimize} \quad \sum_{i=1}^{N} t_i^m \tag{2.1}$$

$$\text{minimize} \quad \int_{t_i^0}^{t_i^0+t_i^m} u_i^2(t)\,\mathrm{d}t \tag{2.2}$$

where the first objective function minimizes the travel time and the second objective function minimizes the fuel consumption of all the vehicles. Both of the objective functions are subject to (2.3), (2.4), (2.5), (2.6) and (2.7).

2.3 Constrains

- CAVs follow second order dynamics such that:

$$\begin{aligned} \dot{x}_i(t) &= v_i(t), \quad x_i(t_i^0) = 0 \\ \dot{v}_i(t) &= u_i(t), \quad v_i(t_i^0) = v_i^0 \end{aligned} \tag{2.3}$$

where $x_i(t)$, $\dot{v}_i(t)$ and $u_i(t)$ are the position, velocity and acceleration of CAV i at time t with $u_i(t)$ being the control input

- To ensure that the velocity and acceleration are within physical limits, the following constraints are imposed:

$$\begin{aligned} 0 \leq v_i(t) &\leq v_i^{max} \quad \forall t \in [t_i^0, t_i^0 + t_i^m] \\ u_i^{min} \leq u_i(t) &\leq u^{max} \quad \forall t \in [t_i^0, t_i^0 + t_i^m] \end{aligned} \tag{2.4}$$

- Each CAV $i \in \mathcal{N}$ arrives at the MZ at a predefined velocity $v_i^m = v^m$ and moves with constant velocity inside the MZ:

$$v_i(t) = v^m, \quad \forall t \in [t_i^0 + t_i^m, t_i^0 + t_i^f] \tag{2.5}$$

- To ensure rear-end collision avoidance, all vehicles traveling in the same direction of movement are enforced to maintain a safe distance δ at all times such that

$$x_k(t) - x_i(t) \geq \delta. \quad \forall t \in [t_i^0, t_i^0 + t_i^f] \tag{2.6}$$

  where k denotes the index of the vehicle in front of CAV i in the same direction of movement.

- In order to avoid lateral collisions inside the MZ, the merging time of CAVs i and j, $i < j$ travelling in perpendicular directions (i.e. NS and EW) must adhere to the constraint

$$t_i^0 + t_i^m + \frac{S}{v^m} \leq t_j^0 + t_j^m \tag{2.7}$$

  which ensures that vehicle j enters the MZ only after vehicle i has exited the MZ.

Chapter 3

Simulated Annealing Algorithm

3.1 Python implementation

We created our Simulated annealing algorithm using Python. To do that we first created two classes to formulate our problem:

- **CAV(direction, To, Vi)**

  This class represents each CAV in our problem. the direction here can be ”NS” or ”EW”. To is the time when the CAV arrives CZ and Vi is the velocity at this time. The class has also three other attributes x (position), v (velocity), and u (acceleration). Each one of these attributes is an array of arrays. The smaller array has two elements: the time and the x or v or u at this time. So, for example the value of u[0] = [t,u].

- **problem(Vmax, Vmin, Umax, Umin, Vc, L, S, safe_distance, delta_T)**

  This class represents our problem with the variables needed as shown. The class has also another attribute which is an array of the CAVs in our problem. The class has a lot of functions:

  - **setCAVs():** This function set our CAVs in the array CAVs.
  - **setRandomSol(CAV):** This function takes a CAV as an input and generate random values for the array u as the acceleration here is our decision variable. Then based on the values of u we update the other two arrays x and v using equations of motion under constant acceleration.
  - **costFunc():** This functions returns the objective function of time added with the objective function of fuel. In our problem we want to minimize this cost function.
  - **isFeasible(CAVs):** This functions returns True or False if the solution generated is

feasible or not based on the constraints illustrated in Chapter 1.

- **SimulatedAnnealing(problem, max_ter, init_T, final_T, iter_Per_T, cooling_Schedule, beta, alpha)**

  This class represents the SA algorithm and we used our problem classes and functions to be able to generate an optimal solution for our problem. The class has Four functions:

  - **coolDownTemp(current_Iteration):** This functions reduces the temperature after each iteration.
  - **arraySols(CAVs):** This function returns an array of CAVs after updating x, v, and u and checking the feasibility of the solution.
  - **performAlgorithm():** In this function the SA algorithm is done.
  - **visualize():** In this function we generate a graph where the iteration number in x-axis vs the cost function value in y-axis.

3.2 Results and Tests

The algorithm was run and tested extensively several times. The algorithm was run for L=300 m, $V_{max} = 25m/s$, $V_{min} = 5m/s$, $u_{max} = 5m/s^2$, $u_{min} = -5m/s^2$, $V_m = 15m/s^2$, $\delta = 20m$ and for a number of four cars. The simulated annealing algorithm was run for different number of iterations as follows:

3.2.1 Total number of iterations = 1000

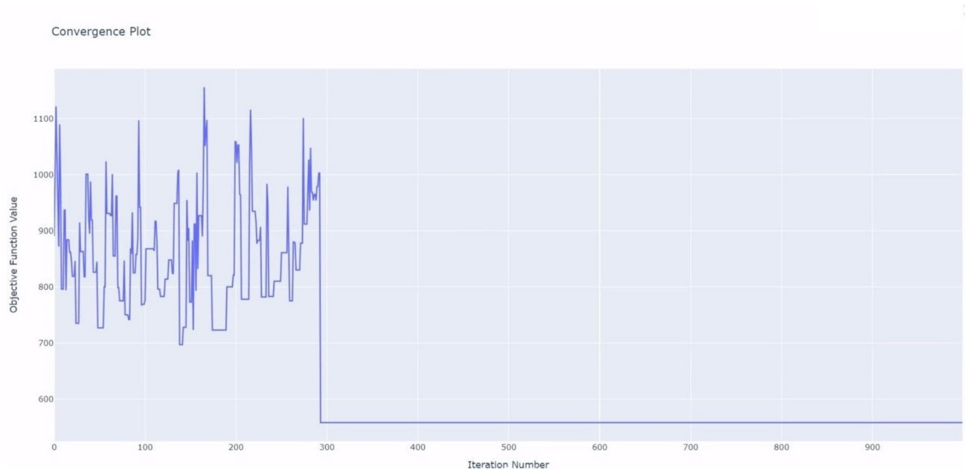

Figure 3.1: Convergence at a total of 1000 iterations

As shown in Figure 3.1, the cost function converges to a minimum value quickly after approximately 300 iterations. Prior to that, when the temperature was high, the exploration feature was dominant after which the exploitation feature becomes the more dominant to drive the algorithm to converge to a minimum cost value. As an example, the results for the acceleration, velocity and position for the first car of the best solution are illustrated in Figures 3.2, 3.3 and 3.4 respectively. In these figure, it is obvious that all the constraints are satisfied which were explained in detail in Chapter 1.

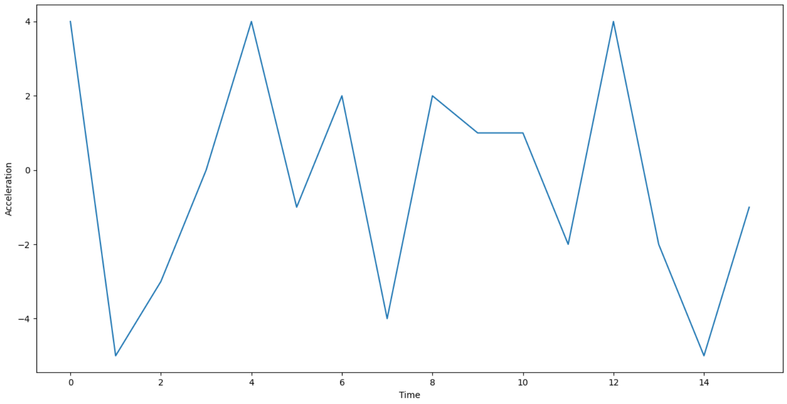

Figure 3.2: Acceleration profile of the first car

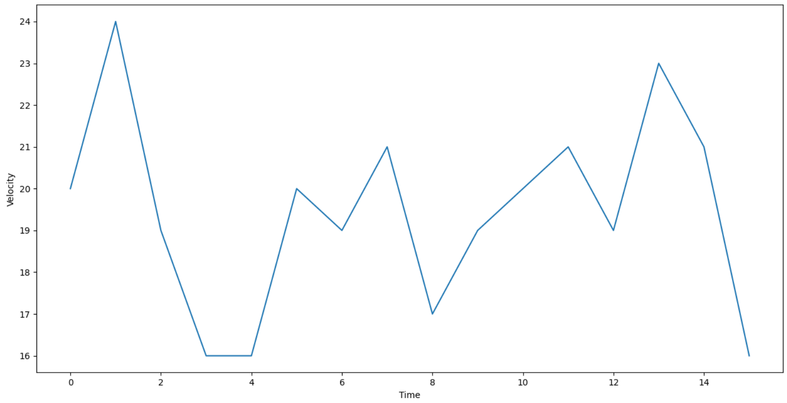

Figure 3.3: Velocity profile of the first car

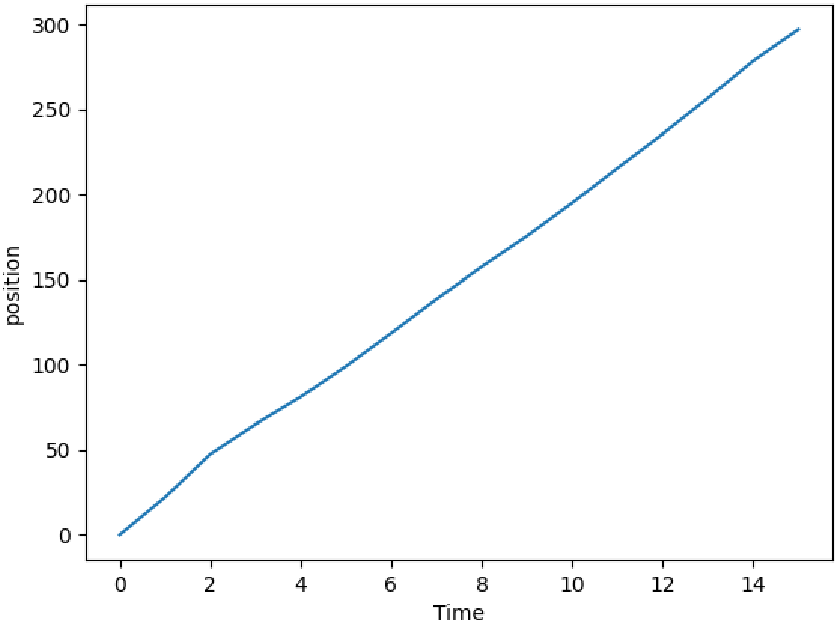

Figure 3.4: Position profile of the first car

3.2.2 Total number of iterations = 3000

The convergence at a 3000 total number of iterations is shown in Figure 3.6.

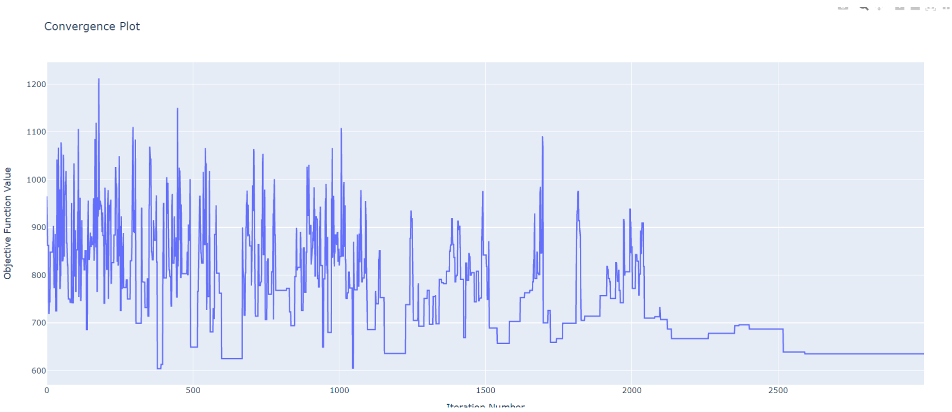

Figure 3.5: Convergence at a total of 3000 iterations

3.2.3 Total number of iterations = 5000

The convergence at a 5000 total number of iterations is shown in Figure 3.6.

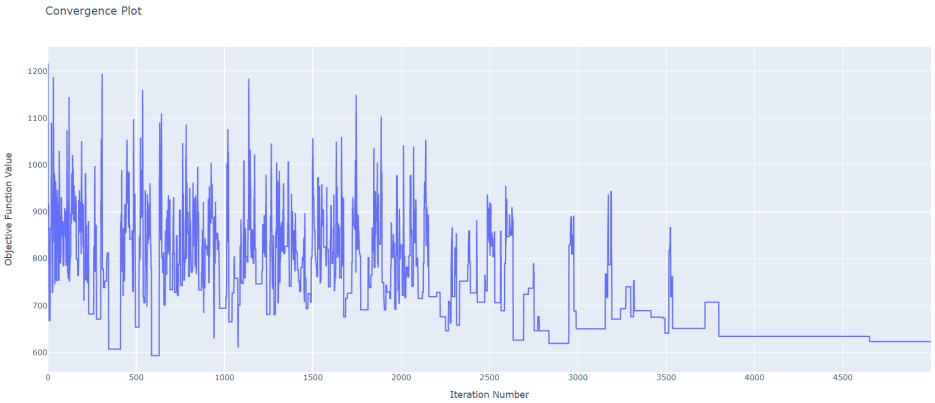

Figure 3.6: Convergence at a total of 5000 iterations

It is clear from the previous figures that when the number of iterations increases, it allows more time for exploration which can lead to finding a more optimal solution that was undiscovered before,

Bibliography

[1] Willy Carlos Tchuitcheu, Christophe Bobda, and Md Jubaer Hossain Pantho. Internet of smart-cameras for traffic lights optimization in smart cities. *Internet of Things*, 11:100207, 2020.

[2] Biao Xu, Xuegang Jeff Ban, Yougang Bian, Wan Li, Jianqiang Wang, Shengbo Eben Li, and Keqiang Li. Cooperative method of traffic signal optimization and speed control of connected vehicles at isolated intersections. *IEEE Transactions on Intelligent Transportation Systems*, 20(4):1390–1403, 2018.

[3] Dian Hartanti, R Nur Aziza, and P Catur Siswipraptini. Optimization of smart traffic lights to prevent traffic congestion using fuzzy logic. *TELKOMNIKA Telecommunication Computing Electronics and Control*, 17(1):320–327, 2019.

[4] Mădălin-Dorin Pop. Traffic lights management using optimization tool. *Procedia-social and behavioral sciences*, 238:323–330, 2018.

[5] Mehrdad Tajalli and Ali Hajbabaie. Distributed optimization and coordination algorithms for dynamic speed optimization of connected and autonomous vehicles in urban street networks. *Transportation research part C: emerging technologies*, 95:497–515, 2018.

[6] SMA Bin Al Islam and Ali Hajbabaie. Distributed coordinated signal timing optimization in connected transportation networks. *Transportation Research Part C: Emerging Technologies*, 80:272–285, 2017.

[7] Boli Chen, Xiao Pan, Simos A. Evangelou, and Stelios Timotheou. Optimal control for connected and autonomous vehicles at signal-free intersections. 53(2):15306–15311, 2020.

[8] Mehmet Ali Guney and Ioannis A. Raptis. Scheduling-based optimization for motion coordination of autonomous vehicles at multilane intersections. 2020:1–22, March 2020.

[9] Mehrdad Tajalli, Mehrzad Mehrabipour, and Ali Hajbabaie. Network-level coordinated speed optimization and traffic light control for connected and automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–12, 2020.

[10] Changxi Ma, Wei Hao, Aobo Wang, and Hongxing Zhao. Developing a coordinated signal control system for urban ring road under the vehicle-infrastructure connected environment. *IEEE Access*, 6:52471–52478, 2018.