TollHelper: A Safe and Efficient Traffic Control Approach on Toll Plaza via Constrained Load Balancing

Mengbing Zhou[1], Bocong Zhao[1], Minxian Xu[1], Hao Dai[1], Yang Wang[1,2,3*]

[1] *Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China*
[2] *Shenzhen University of Advanced Technology, Shenzhen, China*
[3] *The SIAT-Suntang Big Data&AI Joint Innovation Laboratory, Shenzhen, China*
* *Corresponding: yang.wang1@siat.ac.cn*

***Abstract*—Traffic congestion at toll plazas is a critical issue in urban infrastructure, which is often exacerbated by surges in vehicle volume during peak hours. The congestion typically arises from imbalances in traffic demand and toll booth efficiency, often resulting in safety hazards and delays. Existing solutions, while addressing efficiency or safety aspects, often lack a comprehensive approach for efficient traffic management at toll plazas. To address this challenge, in this paper, we propose TollHelper, a framework designed to optimize vehicle scheduling and load balancing at toll plazas as well as improve safety. Our approach treats concurrently arriving vehicles as a single scheduling batch, guiding different vehicles from the same batch to different toll booths to enhance safety and reduce congestion. We address this scheduling constraint in both general and heterogeneous toll booth scenarios, introducing effective load balancing algorithms to minimize toll booth service loads and optimize user driving experiences. Based on empirical studies, we demonstrate that our methods achieve improvements in standard deviation compared to the baselines, ranging from 51.6% to 97.0% improvement in terms of load balancing effects.**

***Index Terms*—toll plaza, heterogeneous efficiency, load balancing, batch scheduling, safety**

1. Introduction

Traffic congestion is a major challenge in modern urban life, often occurring at key nodes in the road network. Toll plazas on expressways are typical hotspots where thousands of vehicles need to pass through quickly and safely every day in big city. As a result, these plazas often experience high congestion due to the imbalance between traffic demand and supply, especially during peak hours when the number of travelers surges [1], [2].

While the imbalance between demand and supply is the main cause of traffic congestion, the severity of the congestion usually stems from several other related factors that can be observed at any toll plaza. First, the vehicles entering the station lack necessary guidance, making it difficult to choose the optimal toll booth for passing through. Second, the efficiency of toll booths is not consistent, which hinders the formation of a smooth vehicle flow to reduce traffic. Finally, the vehicles arriving in parallel may have a criss-cross—different vehicles trying to enter the same queue of a toll booth, leading to potential scratches and collisions, which could further obstruct the traffic [3]. To address these issues and optimize traffic flow, numerous studies have been conducted in both academia and industry, each with its own goals and respective methods [4], [5], [6].

Although these existing solutions (including variable lane configurations, optimized lane allocations and traffic forecasting, etc.) can address certain aspects of the problem in some particular scenarios, they often fail to consider the third aspect mentioned above and consequently lack an integrated comprehensive solution for effectively managing traffic flow in a safe way at toll plazas [7], [8].

To fill this gap, we focus on the traffic control problem at toll plazas with an attempt to achieve safe and efficient traffic control. To this end, we carefully design a set of optimization algorithms to balance the traffic across toll booths, ensuring that vehicles arriving in parallel, as shown in Figure 1, are dispatched to different toll booths in a criss-cross free way. With this strategy, we can increase the distance and space between vehicles, thus reducing the likelihood of accidents and enhancing the safety [9], [10]. Furthermore, to integrate these algorithms for efficiently addressing the traffic control challenges while ensuring the safety at toll plazas, we also develop a framework, called *TollHelper*.

Our algorithms target two key scenarios. Firstly, in general scenarios, we explore the methods for guiding vehicles entering the toll plaza within the same small time slot to different booths. This step lays the foundation for load balancing, aiming to distribute traffic evenly across available toll booths and minimize the maximum service load, i.e., the cumulative service time at each booth. Secondly, we analyze the scenarios where toll booths operate at varying levels of efficiency due to collection method, physical location, and staff proficiency [5], [7], [11]. To accommodate these disparities, we investigate several specific allocation methods. By employing these load balancing scheduling algorithms, we are able to balance the service load between booths and

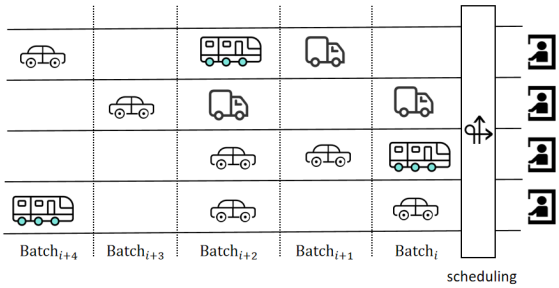


Figure 1: The scenario of vehicle batch scheduling.

minimize the maximum cumulative service time, thereby optimizing the user experience.

In summary, our key **contributions** are as follows:

1) We introduce TollHelper, an auxiliary framework to integrate the proposed scheduling algorithms with the objective to enhance the safety and balance the traffic loads across all the toll booths.
2) We propose a greedy algorithm for scheduling vehicles arriving simultaneously for general scenario of toll plazas, and a dynamic programming algorithm for further optimized scheduling.
3) We present two algorithms for the heterogeneous service capacities of toll booths: one achieves a local optimal solution for each allocation, and the other uses a heuristic to monitor deficiencies reaching a specified actual service time threshold.
4) We conduct an empirical study by using realistic highway vehicle datasets [12] to evaluate our methods and compare them with baselines to validate the effectiveness.

Overall, these contributions demonstrate the potentials of our approach to improve both the operational efficiency and safety of toll plaza traffic managements.

In the reminders of this paper, we first overview related work in Section 2, and formulate the problem and describe the TollHelper framework in Section 3. Then, we propose our scheduling algorithms in Section 4, and demonstrate their effectiveness on real traffic datasets in Section 5. We finally conclude the paper in the last section.

2. Related Work

Traffic management is essential in modern transportation systems. By optimizing traffic signals, monitoring traffic flow, and implementing intelligent transportation technologies, it reduces road congestion, improves transit efficiency, decreases accident rates, and enhances overall traffic safety and travel experience.

Recent research has extensively explored traffic management, particularly focusing on toll plazas, which are critical nodes in the transportation network [13], [14]. Yuan et al. [5] proposed an optimal control method to improve traffic efficiency using tollbooth lane configuration and variable speed limit (VSL) control. Zhou et al. [15] introduced a proactive traffic control method using dynamic lane configuration and VSL, coupled with a long short-term memory neural network model for short-term traffic demand forecasting at toll plazas. Furthermore, the adoption of emerging technologies such as machine learning and neural networks to predict traffic patterns and enhance flow is increasingly prevalent. Pan et al. [16] propose Ising-Traffic, a dual-model Ising-based traffic prediction framework that achieves higher accuracy and lower latency than state-of-the-art solutions, effectively addressing traffic congestion prediction under uncertainty. In contrast, Petrovic et al. [6] utilized several recurrent neural network architectures to predict the average intensity of vehicle arrivals whereby the number of lanes opened in each time period can be calculated for cost-effective toll service. Ramana et al. [17] designed and enforced a traffic prediction scheme using Vision Transformers and CNN to accurately forecast traffic flow on a city-wide scale, demonstrating superior precision, accuracy, and recall, particularly during anomalous traffic situations. Yang et al. [18] proposed a multi-graph learning-based model named TPP-GCN to predict traffic propagation flow in urban road networks by capturing both temporal and multi-spatial features using multi-layer convolution. These approaches focus primarily on efficiency gains, but do not include safety as a consideration.

However, toll plazas have a higher accident rate compared to regular road sections, often significantly impeding traffic [19]. Therefore, enhancing toll plaza safety is crucial for ensuring smooth road conditions. Xing et al. [20] used logistic regression and five non-parametric models to examine the relationship between influencing factors and vehicle collision risk, comparing the models' strengths and weaknesses for accurate risk assessment. In a further study, Xing et al. [21] introduced the concept of motion constraint degree to investigate traffic conflict risks in toll plaza diverging areas, developing a two-step method for risk regression and prediction using random parameters logit models and machine learning, demonstrating superior performance over conventional methods. In addition, Mo et al. [22] analyzed and predicted short-term crash risk at toll plazas using a random-effects logit regression model to identify crash precursors, coupled with an LSTM-CNN network for crash prediction, showcasing its effectiveness in enhancing traffic safety and real-time management at toll plazas. Xiang et al. [23] studied how to optimize guidance signage systems, demonstrating that a complete manual toll collection guidance plan can significantly improve driver performance and safety. These efforts contribute a lot to enhancing the safety of toll plazas, but additional measures can often be taken to further improve the service efficiency. Unfortunately, this aspect is frequently overlooked.

In short, studies on toll plazas usually do not give reasonable consideration to the safety hazards involved in the optimization strategies to improve efficiency, and operational efficiency is often sacrificed when safety is improved. By contrast, our study focuses on rationally combining the operational efficiency and safety of toll plazas, and considers enhancing vehicle safety as a prerequisite and an important

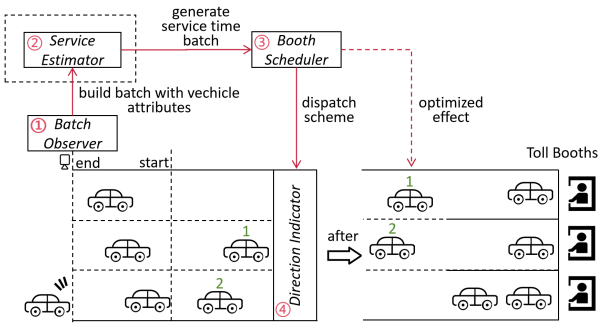


Figure 2: Vehicle batch scheduling framework for toll plaza.

means to ensure operational efficiency, based on which the efficiency is further enhanced by scheduling algorithms.

3. Problem Modeling and Framework Design

In front of toll booths, there are typically multiple parallel lanes, with the number of lanes generally fewer than the number of toll booths. Without loss of generality, our model assumes an equal number of lanes and toll booths. It is always possible to view the vehicles advancing in a small time slot as a batch (at most one vehicle per lane and no empty batches). Upon arrival, each vehicle is dispatched to a distinct toll booth to complete its service.

The service time of vehicles varies due to multiple factors such as vehicle type, size, weight, toll rates, and personnel attributes [24], [25]. Nonetheless, researchers have employed various methods such as mathematical modeling, historical service record analysis, and other technologies to predict vehicle service times, and the accuracy of such predictions has been improved over time [26]. Therefore, we can utilize these prediction methods to consider the service time of each vehicle as known.

The vehicle batch scheduling problem can be formulated as a model involving m toll booths, where the load (i.e., cumulative service time) of each booth is denoted by $\vec{L} \triangleq [L_1, L_2, \ldots, L_m] \in R^{1\times m}$. Let $\mathcal{B} \triangleq \{B_1, B_2, \ldots\}$ represent the set of all vehicle batches that need to be scheduled, with each batch $B_k \in \mathcal{B}, B_k \triangleq \{v_1, v_2, \ldots\}$ containing at least one and no more than m vehicle service times. The service times of vehicles within each batch B_k are allocated to distinct toll booths. For any $v, v' \in B_k$, if v is assigned to booth L_i and v' is assigned to booth L_j, with $i, j \in \{1, 2, \ldots, m\}$, then it must be that $i \neq j$. Let τ denote the moment when the vehicle batch appears.

Additionally, we will consider the heterogeneous toll booth scenario, let $\vec{P} \triangleq [p_1, p_2, \ldots, p_m] \in R^{1\times m}$ denote the service efficiency ratios of the toll booths relative to a standard toll booth.

Our objective is to achieve load balancing by minimizing the final maximum service load, i.e., the cumulative actual service time. Thus, the optimization objective can be formulated as $\min\{\max\{L_i\} \mid i \in \{1, 2, \ldots, m\}\}$.

In order to apply traffic control practically, we also propose a comprehensive framework called TollHelper, which aims to optimize vehicle flow and ensure safety through load balancing techniques, as shown in Figure 2. The framework consists of the following main modules:

1) **Batch Observer:** This module is responsible for detecting incoming vehicles at the toll plaza. It groups vehicles into batches based on their arrival within a short time window or when a second vehicle appears in the same lane. This data, along with each vehicle's characteristics, is then passed to the next module.
2) **Service Estimator:** Using existing vehicle service time prediction algorithms, this module calculates the service time for all vehicles in each batch. The results are then forwarded to the next module.
3) **Booth Scheduler:** This module uses the designed vehicle batch scheduling algorithm to allocate each vehicle in the batch to different toll booths. The scheduling results are then sent to the next module.
4) **Direction Indicator:** This module directs vehicles to their assigned toll booths. It provides clear directions and instructions to drivers, ensuring that vehicles reach the correct booths efficiently and safely.

The performance of the *Booth scheduler* module is crucial for the overall efficiency of the TollHelper framework. In the following sections, we discuss vehicle batch scheduling algorithms for both general and heterogeneous toll booth efficiency scenarios. By optimizing the *Booth Scheduler*, the algorithms can significantly enhance traffic flow and improve safety at the toll plaza.

4. Vehicle Batch Dispatch Algorithms

In this section, we consider the vehicle scheduling algorithms within the framework, addressing two scenarios: whether the toll stations are heterogeneous. For each scenario, we introduce effective load balancing algorithms accordingly.

4.1. General Scenario

For the general scenario, to speed up the decision time, we have developed a simple Load Greedy Algorithm (LGA). It schedules a batch at each moment, arranging the vehicles in descending order and then sequentially assigning the vehicle to the booth that is currently least loaded and has no other vehicles.

Theorem 1. *Algorithm 1 ensures that a local optimal solution is obtained and minimizes the maximum load difference between the booths at any moment τ.*

Proof. Without loss of generality, at moment τ, let $L_m \leq L_{m-1} \leq \cdots \leq L_1$ denote the loads of the toll booths, and let $v_1 \geq v_2 \geq \cdots \geq v_m$ denote the service times of the vehicles in the same batch. Both sequences can have trailing zeros. After Algorithm 1 completes the assignment, the load of

Algorithm 1 Load Greedy Algorithm (LGA)

Input:
m: Number of toll booths;
$\mathcal{B}$: A group of vehicle batches.
Output:
$\vec{L}$: Service load of toll booths.
1: $\vec{L} \leftarrow [0] * m$
2: **for all** B in $\mathcal{B}$ **do**
3: $\quad \vec{I} \leftarrow sort\_ascend(index(\vec{L}))$;
4: $\quad B \leftarrow sort\_descend(service\_time(B))$;
5: $\quad pos \leftarrow 0$;
6: $\quad$ **for all** v in B **do**
7: $\quad\quad \vec{L}[\vec{I}[pos]] \leftarrow \vec{L}[\vec{I}[pos]] + v$;
8: $\quad\quad pos \leftarrow pos + 1$;
9: $\quad$ **end for**
10: **end for**

each booth is $L_m + v_1, L_{m-1} + v_2, ..., L_1 + v_m$ respectively. Let the maximum value is $L_{m-i+1} + v_i, i \in \{1, 2, \ldots, m\}$.

Assuming that there is a combination method with a smaller maximum value, where the $v^{'} < v_i$ combine with L_{m-i+1}. Now consider the new position of v_i. Keep the order of $L_m \leq ... \leq L_{m-i+1} \leq ... \leq L_1$, if v_i is assigned to the right of L_{m-i+1}, then $L_x + v_i \geq L_{m-i+1} + v_i, x \in \{m-i, \ldots, 1\}$, this contradicts the hypothesis. If v_i is assigned to the left of L_{m-i+1}, it's obvious that at least one $v_j \geq v_i, j \in \{1, \ldots, i-1\}$ is assigned to the right of L_{m-i+1}, so $L_x + v_j \geq L_{m-i+1} + v_i$, this contradicts the hypothesis. So there is no other combination method with a smaller maximum value than Algorithm 1, i.e., it obtains a local optimal solution.

Similarly, it can be concluded that no other combination method produces a greater minimum value than Algorithm 1. Therefore, maximum load difference between booths is minimized. □

In Algorithm 1, the time complexity for the two sorts is $O(m \log m)$, and updating the load is $O(m)$, resulting in a total time complexity of $O(m \log m)$ for dispatching a batch.

We can enhance the load balancing performance by implementing additional measures. For instance, detecting incoming vehicles at a distance far from the toll booths and subsequently controlling their speed allows the *Booth Scheduler* module to dispatch multiple vehicle batches at once. By treating the current service load at the toll booth as an initial batch and combining it with subsequent vehicle batches, we can establish an optimal sequence for merging multiple batches, referred to as a multi-batch merging scheme. This approach enables more effective load balancing over a given distance, especially during congested hours.

To achieve this, we propose a Dynamic Programming Optimization Algorithm (DPOA) as shown in Algorithm 2. Assuming that k vehicle batches are scheduled simultaneously each time, and that the initial batch of toll booth loads and vehicle batches form a $slice$, the length of the $slice$ is $k + 1$. Besides, we extract the lines 3-9 from Algorithm 1 and replace $\vec{L}$ with another batch to create a prepared

Algorithm 2 DP Optimization Algorithm (DPOA)

Input:
m: Number of toll booths;
k: The number of vehicle batches dispatched at once;
$\mathcal{B}$: A group of vehicle batches.
Output:
$\vec{L}$: Service load of toll booths.
1: $\vec{L} \leftarrow [0] * m$;
2: **while** $\mathcal{B}$ **do**
3: $\quad slice \leftarrow [\vec{L}, \mathcal{B}[0], \ldots, \mathcal{B}[k-1]]$;
4: $\quad \mathcal{B} \leftarrow \mathcal{B}[k : \text{end}]$;
5: $\quad$ **for** $j \leftarrow 0$ to k **do**
6: $\quad\quad a \leftarrow [\infty] \times (k - j + 1)$;
7: $\quad\quad d$.append(a);
8: $\quad\quad d[j, 0] \leftarrow \text{slice}[j]$;
9: $\quad$ **end for**
10: $\quad$ **for** $r \leftarrow 1$ to k **do**
11: $\quad\quad$ **for** $s \leftarrow 0$ to $k - r$ **do**
12: $\quad\quad\quad$ **for** $t \leftarrow s$ to $s + r - 1$ **do**
13: $\quad\quad\quad\quad temp \leftarrow f(d[s, t-s], d[t+1, s+r-t-1])$;
14: $\quad\quad\quad\quad$ **if** $\max(d[s, r]) > \max(temp)$ **then**
15: $\quad\quad\quad\quad\quad d[s, r] \leftarrow temp$;
16: $\quad\quad\quad\quad$ **end if**
17: $\quad\quad\quad$ **end for**
18: $\quad\quad$ **end for**
19: $\quad$ **end for**
20: $\quad \vec{L} \leftarrow d[0, k]$;
21: **end while**

function f, which takes two batches as input and outputs their merging results.

We use a dynamic programming array d to record the optimal result (a new batch) after merging from the *i-th* batch to the *j-th* batch of the $slice$. The base case of the dynamic transfer process is when $i = j$:

$$d[i, j] = slice[i]. \tag{1}$$

During the initialization phase, d is initialized as a three-dimensional array, shaped like an upper triangular matrix, where each element is an array:

$$d = \begin{bmatrix} slice[0] & [+\infty] & [+\infty] & \ldots & [+\infty] \\ & slice[1] & [+\infty] & \ldots & [+\infty] \\ & & slice[2] & \ldots & [+\infty] \\ & & & \ddots & \vdots \\ & & & & slice[k] \end{bmatrix}. \tag{2}$$

For $i < j$, the transfer process is as follows:

$$d[i, j] = \min_{i \leq p \leq j-1} \max\left(f(d[i, p], d[p+1, j])\right). \tag{3}$$

The final value of d is computed using three nested loops in lines 10-19 of Algorithm 2. These loops represent, from outer to inner, the range of the merging batches, the various starting points of that range, and the determination of the optimal solution for merging the batches within that range by subproblems.

When k vehicle batches are optimized, the time complexity of initializing the dynamic programming array d in Algorithm 2 is $O(k)$. The time complexity of executing the

Algorithm 3 Local Optimal Greedy Algorithm (LOGA)

Input:
m: Number of toll booths;
$\vec{P}$: Efficiency ratio of heterogeneous toll booths;
$\mathcal{B}$: A group of vehicle batches.
Output:
$\vec{L}$: Service load of toll booths.

$\vec{P} \leftarrow sort\_ascend(\vec{P})$;
Initialize A, a as an empty array;
for $i \leftarrow 0$ to $m-2$ **do**
  for $j \leftarrow i+1$ to $m-1$ **do**
    a.append($(\vec{P}[i] - \vec{P}[j])/(\vec{P}[i] \times \vec{P}[j])$);
  end for
  A.append(a);
end for
$\vec{L} \leftarrow [0] * m$;
for all B in $\mathcal{B}$ **do**
  $B \leftarrow sort\_descend(service\_time(B))$;
  $\vec{I} \leftarrow [0, 1, \ldots, m-1]$;
  for all v in B **do**
    $i, j \leftarrow 0, 1$;
    while $j <$ len($\vec{I}$) **do**
      if $\vec{L}[\vec{I}[i]] \leq \vec{L}[\vec{I}[j]]$ **then**
        $j \leftarrow j+1$;
      else if $v \times A[\vec{I}[i], \vec{I}[j] - \vec{I}[i]] > \vec{L}[\vec{I}[i]] - \vec{L}[\vec{I}[j]]$ **then**
        $j \leftarrow j+1$;
      else
        $i \leftarrow j$;
        $j \leftarrow j+1$;
      end if
    end while
    $\vec{L}[\vec{I}[i]] \leftarrow \vec{L}[\vec{I}[i]] + v/\vec{P}[\vec{I}[i]]$;
    Remove $\vec{I}[i]$ from $\vec{I}$;
  end for
end for

f function is $O(m \log m)$, then the time complexity of the dynamic programming process is $O(m \log m * k^3)$. Hence the total time complexity is $O(m \log m * k^3)$.

4.2. Heterogeneous Toll Booths

The dynamic adjustment of resources and configurations at toll booths leads to heterogeneous service efficiencies. Therefore, we consider load balancing in this realistic context. As defined in Section 3, the ratio of each toll booth's service efficiency to the standard service efficiency is denoted by $\vec{P}$. At this point, we consider the output of the *Service Estimator* module as the Standard Service Time (SST), while the Actual Service Time (AST) of the vehicle varies depending on the efficiency of the toll booth.

It is intuitive that at each moment τ, we are able to achieve a local optimal solution, similar to Algorithm 1. So we proposed a Local Optimal Greedy Algorithm (LOGA) as shown in Algorithm 3.

Let $p \neq q$ and $p, q \in \{1, 2, \ldots, m\}$, with $\vec{P}[p] > \vec{P}[q]$. To minimize the maximum load (cumulative AST) when assigning a vehicle v, the following considerations should be made:

- If $L_p \leq L_q$, booth q should be eliminated because booth p is faster and has less load.
- If $L_p > L_q$, consider:

$$L_q + \frac{v}{\vec{P}[q]} > L_p + \frac{v}{\vec{P}[p]}. \tag{4}$$

This inequality indicates that the load at booth q exceeds that at booth p, hence booth q should be eliminated. It morphs into:

$$\frac{v(\vec{P}[p] - \vec{P}[q])}{\vec{P}[p]\vec{P}[q]} > L_p - L_q. \tag{5}$$

Conversely, if

$$\frac{v(\vec{P}[p] - \vec{P}[q])}{\vec{P}[p]\vec{P}[q]} \leq L_p - L_q. \tag{6}$$

then booth p should be eliminated.

Based on the above discussion, we can derive an efficient assignment strategy. At any moment τ, vehicles should be arranged in descending order of their SST and assigned accordingly. Among the remaining toll booths that have not yet been assigned a vehicle, initiate the search process starting with two booths. Iteratively eliminate one toll booth at a time based on the above judgment criteria. Ultimately, identify the most suitable toll booth, assign the vehicle to it, and remove that booth from the list of remaining booths.

To simplify the search process, we sort $\vec{P}$ in descending order and define $a_{p,q} = \frac{\vec{P}[p] - \vec{P}[q]}{\vec{P}[p]\vec{P}[q]}$ to precompute the following upper triangular matrix A:

$$A = \begin{bmatrix} a_{0,1} & a_{0,2} & \cdots & a_{0,m-1} \\ & a_{1,2} & \cdots & a_{1,m-1} \\ & & \ddots & \vdots \\ & & & a_{m-2,m-1} \end{bmatrix}. \tag{7}$$

By utilizing matrix A in Algorithm 3, the judgment process becomes more concise and clearer.

Theorem 2. *At any moment τ, the Algorithm 3 obtains a local optimal solution.*

Proof. At a certain moment τ, assume that the booth chosen by vehicle v reaches the maximum load. We now show that making the maximum load smaller is impossible. When v selects a booth, it has already compared all the remaining booths and selected the one that minimizes the load, so we do not have to try to adjust v to these booths. When adjusting v to a booth that has already been selected before v, there must be at least one vehicle $v' \geq v$ that is adjusted to or after the original position of v. Consequently, the AST generated by v' must be greater than or equal to the time generated by v, thus a smaller maximum load cannot be achieved. So the Algorithm 3 achieves a local optimal solution. □

In Algorithm 3, each time a batch is dispatched, the time complexity for sorting is $O(m \log m)$. Allocating all vehicles within the batch through a nested loop has a time complexity of $O(m^2)$, resulting in a total time complexity of $O(m^2)$.

Algorithm 4 Reverse Load Greedy Algorithm(RLGA)

Input:
m: Number of toll booths;
T: Time period;
$\vec{P}$: Efficiency ratio of heterogeneous toll booths;
$\mathcal{B}$: A group of vehicle batches.
Output:
$\vec{L}$: Service time of toll booths.

$\vec{L}, \vec{\theta} \leftarrow [0] * m$;
for all p in $\vec{P}$ **do**
  $\vec{\theta}$.append($p \times T$);
end for
$\vec{L}^{'} \leftarrow \vec{\theta}$;
for all B in $\mathcal{B}$ **do**
  $\vec{I} \leftarrow sort\_descend(index(\vec{L}^{'}))$;
  $B \leftarrow sort\_descend(service\_time(B))$;
  $pos \leftarrow 0$;
  for all v in B **do**
    $\vec{L}^{'}[\vec{I}[pos]] \leftarrow \vec{L}^{'}[\vec{I}[pos]] - v$;
    $\vec{L}[\vec{I}[pos]] \leftarrow \vec{L}[\vec{I}[pos]] + v/\vec{P}[\vec{I}[pos]]$;
    if $\vec{L}^{'}[\vec{I}[pos]] \leq 0$ **then**
      $\vec{L}^{'} \leftarrow \vec{L}^{'} + \vec{\theta}$;
    end if
    $pos \leftarrow pos + 1$;
  end for
end for

The comparison for each allocation in Algorithm 3 is rigorous and we observed that heterogeneous toll booths require different SST to achieve the same AST. In order to simplify the process, we propose a greedy algorithm, called Reverse Load Greedy Algorithm (RLGA), that heuristically assigns vehicles by considering the reverse SST each booth requires to reach a specific point in AST. To implement this, we set a time period T and calculate the initial reverse SST over one period. When the reverse SST of a booth becomes less than or equal to zero, we update the reverse SSTs of all booths by the period T until all reverse SSTs are positive. Let $\vec{L}^{'}$ represent the reverse SST of each booth, then the update process is:

$$\vec{L}^{'} = \vec{L}^{'} + T \times \vec{P}. \tag{8}$$

The choice of the time period T can affect the effectiveness of Algorithm 4. If T is too large, most vehicles will be assigned to the fast booths, leaving other booths with minimal load or even idle, resulting in poor load balancing. Conversely, if T is too small, updates will be frequent, causing unnecessary delays. Therefore, a balance must be struck when setting T.

When we set a reasonable period T, for each batch allocation, Algorithm 4 has a time complexity of $O(m \log m)$ for sorting and $O(m)$ for updating the reverse load, resulting in a total time complexity of $O(m \log m)$. Regardless, compared to Algorithm 3, Algorithm 4 features a simpler process and incurs fewer computational costs, especially when there is a predetermined completion deadline or a long time period is set.

TABLE 1: Normal Distributions Based on Vehicle Attributes

Vehicle type	Length of vehicle (feet)	Normal Distribution
Motorcycle	Any	$\mathcal{N}(12, 7^2)$
Auto	≤ 15	$\mathcal{N}(20, 8.5^2)$
Auto	> 15	$\mathcal{N}(25, 10^2)$
Truck	Any	$\mathcal{N}(30, 12^2)$

5. Performance Evaluations

In this section, we present a comprehensive analysis of the performance of our proposed algorithms. We evaluate their effectiveness through a series of experiments. The results highlight the advantages of our approaches in terms of standard deviation and maximum load. Additionally, we compare our methods with baselines to demonstrate the improvements achieved.

5.1. Experimental Setup

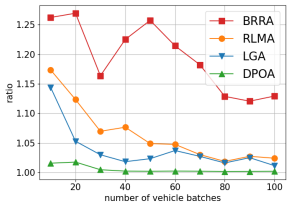
We constructed a simulation environment based on traffic data from two highway segments in the Next Generation Simulation (NGSIM) dataset [12]: US-101 and I-80. Both subsets cover different time periods, corresponding to different levels of congestion. We extracted continuous data from two time periods and generated vehicle batches with a 1-second limit. Since US-101 and I-80 include 5 and 6 main lanes respectively, and toll plazas generally have more toll booths than these numbers, we extracted 200 continuous batches from the generated data and divided them into two segments and combined, resulting in 100 vehicle batches for 10 lanes and 12 lanes.

In the simulation environment, we determined the service time for each vehicle based on the distribution patterns from the existing record data, using different normal distributions according to vehicle type and length. The specific normal distributions followed are shown in Table 1. The service time was strictly ensured to be between 3 and 60 seconds. For heterogeneous service efficiencies, we start from 0.5 and increase by 0.25 each time, repeating each number once, thereby obtaining the vector $\vec{P}$.

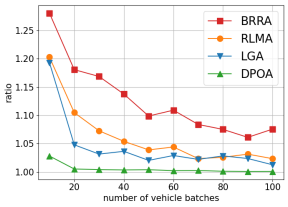
To evaluate the performance of our proposed algorithms, we compared them against several well-known baseline algorithms commonly used for load balancing and scheduling problems. Below, we briefly describe each comparative algorithm:

- *Batch Round-Robin Algorithm (BRRA)* [27]: The BRRA assigns vehicles to toll booths in a cyclic manner. It matches the number of toll booths to the number of vehicles in each batch, ensuring each toll booth receives one vehicle from the batch.
- *Randomized Load Minimization Algorithm (RLMA)* [28]: The RLMA generates multiple (10 in our experiments) random allocation schemes for each batch of vehicles and selects the one that minimizes the largest current load of the toll booths.

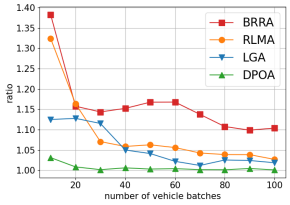
We evaluate the algorithms' performance with two metrics. On one hand, we calculate the standard deviation of the toll booths' loads after allocation, which directly reflects the



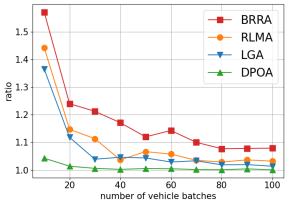
(a) 7:50 a.m. to 8:05 a.m., representing the buildup of congestion in US-101.



(b) 8:20 a.m. to 8:35 a.m., representing full congestion during the peak period in US-101.



(c) 4:00 p.m. to 4:15 p.m., representing the buildup of congestion in I-80.



(d) 5:15 p.m. to 5:30 p.m., representing full congestion during the peak period in I-80.

Figure 3: The variation in the ratio of the maximum load to the ideal load using US-101 and I-80 data in the general scenario.

TABLE 2: Comparison of standard deviations for algorithms across time periods on US-101 and I-80 datasets in general scenarios.

	US-101		I-80	
	7:50-8:05	8:20-8:35	4:00-4:15	4:15-5:30
BRRA	63.2	44.7	55.2	49.1
RMLA	16.1	20.2	19.61	22.67
LGA	7.0	7.2	9.5	9.0
DPOA	**2.2**	**0.6**	**1.4**	**0.9**

final load balancing effect. On the other hand, to visually present the results, we compute the ideal load for each toll booth, assuming an even distribution of loads across all toll booths, which is usually impossible for any algorithm to achieve:

$$L_{\text{ideal}} = \frac{\sum_{i=1}^{T} \sum_{v \in B_i} v}{m}. \tag{9}$$

For heterogeneous issue:

$$L_{\text{ideal}} = \frac{\sum_{i=1}^{T} \sum_{v \in B_i} v}{\sum_{i=1}^{10} \vec{P}[i]}. \tag{10}$$

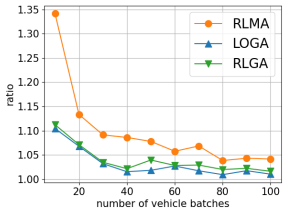
We then analyze the results using the ratio of the maximum load obtained by each algorithm to the ideal load. Assume that L^* is the current maximum load generated by the optimal scheduling algorithm, since the ideal load $L_{ideal} \leq L^*$, we have:

$$r = \frac{L}{L_{\text{ideal}}} \geq \frac{L}{L^*}, \tag{11}$$

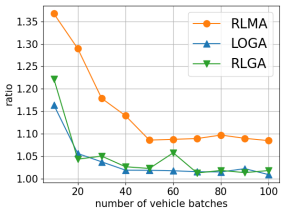
where the right-hand term is the competition ratio.

5.2. Result Analysis

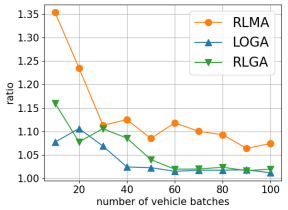
Since RLMA is non-deterministic, each run may yield different results. To mitigate randomness, we executed



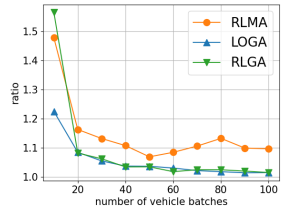
(a) 7:50 a.m. to 8:05 a.m., representing the buildup of congestion in US-101.



(b) 8:20 a.m. to 8:35 a.m., representing full congestion during the peak period in US-101.



(c) 4:00 p.m. to 4:15 p.m., representing the buildup of congestion in I-80.



(d) 5:15 p.m. to 5:30 p.m., representing full congestion during the peak period in I-80.

Figure 4: The variation in the ratio of the maximum load to the ideal load using US-101 and I-80 data in the heterogeneous scenario.

TABLE 3: Comparison of standard deviations for algorithms across time periods on US-101 and I-80 datasets in heterogeneous scenarios.

	US-101		I-80	
	7:50-8:05	8:20-8:35	4:00-4:15	4:15-5:30
RMLA	33.9	50.1	49.6	47.6
LOGA	**9.1**	**9.1**	**8.6**	**7.7**
RLGA	12.1	13.3	15.0	14.3

RLMA ten times in each experiment.

In general scenario, we set the batch size for each optimization in DPOA to be 10. Table 2 shows the standard deviation of toll booth loads when scheduling up to 100 batches. It can be seen that the comparative algorithm RMLA outperforms BRRA. Therefore, we will primarily compare RMLA with the proposed algorithm in subsequent analyses. When comparing LGA to RMLA, the standard deviation was reduced by 56.5%, 64.4%, 51.6%, and 60.3% for the different experimental setups. When comparing DPOA to RMLA, the standard deviation was reduced by 86.3%, 97.0%, 92.9%, and 96.0% respectively. These results demonstrate that LGA and DPOA have significantly lower standard deviations, indicating that they achieve better load balancing. Figure 3 display the ratio of maximum load to ideal load with one run of RLMA.

In the heterogeneous scenario, for the RLGA, we set the period T as the value of the maximum vehicle service time in the first batch divided by the highest service speed ratio of the toll booths. When service efficiencies are different, the coarse-grained BRRA further reduces its effectiveness, resulting in poor performance and is therefore no longer used for comparison. Table 3 shows the standard deviation of toll booth loads when scheduling up to 100 batches. We can see that LOGA improves the standard deviation by 73.2%,

81.8%, 82.7%, and 83.8% over RMLA, respectively. And RLGA improves the standard deviation by 64.3%, 73.4%, 69.8%, and 70.0% over RMLA, respectively. Figure 4 shows the ratio of maximum load to ideal load with one run of RLMA, and it is evident that LOGA and RLGA both perform better than RLMA.

From the results, it is evident that the BRRA performs the worst because of its coarse-grained scheduling in general scenario. Additionally, both in general and heterogeneous scenarios, RMLA is not only weaker than the proposed algorithm in terms of performance, but it is also less stable due to the huge feasible solution space, and large fluctuations may occur in multiple runs. However, our proposed methods overcome these limitations effectively.

6. Conclusion

This paper addresses the challenges of traffic congestion and safety hazards at toll plazas through the introduction of TollHelper, a comprehensive framework for rational vehicle scheduling, which aims to improve service efficiency and enhance safety by guiding simultaneously arriving vehicles to different toll booths. In general scenarios, we offer efficient greedy algorithm for scheduling vehicles arriving simultaneously, alongside a dynamic programming approach that further enhances scheduling effectiveness. Moreover, we tackle the issue of heterogeneous service capacities with two algorithms: one providing local optimal solutions for each allocation and another employing a heuristic to manage service time variations effectively. Through extensive simulations based on real data, our methods demonstrate superior performance compared to existing scheduling approaches. This research not only contributes novel algorithms but also provides practical insights into managing traffic flow at toll plazas, paving the way for enhanced operational efficiency and improved safety in urban transportation networks.

Acknowledgments

This work is supported in part by the 3rd Xinjiang Scientific Expedition Program (2021xjkk1300), the SIAT-SunTang Big Data&AI Joint Innovation Laboratory (E3Z092), Shenzhen Science and Technology Plan Project (Shenzhen-Hong Kong-Macau Category C, No. SGDX20220530111001003), and Shenzhen Science and Technology Program (CJGJZD20230724093659004).

References

[1] Y. Bao, F. Xiao, Z. Gao, and Z. Gao, "Investigation of the traffic congestion during public holiday and the impact of the toll-exemption policy," *Transportation Research Part B: Methodological*, vol. 104, pp. 58–81, 2017.

[2] X. Lin, Y. O. Susilo, C. Shao, and C. Liu, "The implication of road toll discount for mode choice: Intercity travel during the chinese spring festival holiday," *Sustainability*, vol. 10, no. 8, 2018.

[3] Z. Zheng, Z. Wang, S. Liu, and W. Ma, "Exploring the spatial effects on the level of congestion caused by traffic accidents in urban road networks: A case study of beijing," *Travel Behaviour and Society*, vol. 35, p. 100728, 2024.

[4] X. Wang, L. T. Yang, H. Li, M. Lin, J. Han, and B. O. Apduhan, "Nqa: A nested anti-collision algorithm for rfid systems," *ACM Trans. Embed. Comput. Syst.*, vol. 18, no. 4, jul 2019.

[5] N. Yuan, M. Ma, S. Liang, W. Wang, and H. Zhang, "Optimal control method of freeway based on tollbooths lane configuration and variable speed limit control," *Physica A: Statistical Mechanics and its Applications*, vol. 603, p. 127801, 2022.

[6] A. Petrovic, M. Nikolic, U. Bugaric, B. Delibasic, and P. Lio, "Controlling highway toll stations using deep learning, queuing theory, and differential evolution," *Engineering Applications of Artificial Intelligence*, vol. 119, p. 105683, 2023.

[7] J. Oskarbski, L. Gumińska, and K. Żarski, "Influence of toll collection method on motorways on traffic safety and efficiency," in *Management Perspective for Transport Telematics*, J. Mikulski, Ed. Cham: Springer International Publishing, 2018, pp. 142–156.

[8] B. Singichetti, J. L. Conklin, K. Hassmiller Lich, N. S. Sabounchi, and R. B. Naumann, "Congestion pricing policies and safety implications: a scoping review," *Journal of Urban Health*, vol. 98, no. 6, pp. 754–771, 2021.

[9] M. Saad, M. Abdel-Aty, and J. Lee, "Analysis of driving behavior at expressway toll plazas," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 61, pp. 163–177, 2019, special TRF issue: Driving simulation.

[10] P. Song, N. Sze, O. Zheng, and M. Abdel-Aty, "Addressing unobserved heterogeneity at road user level for the analysis of conflict risk at tunnel toll plaza: A correlated grouped random parameters logit approach with heterogeneity in means," *Analytic Methods in Accident Research*, vol. 36, p. 100243, 2022.

[11] I.-D. Ramandanis, I. Politis, and S. Basbas, "Assessing the environmental and economic footprint of electronic toll collection lanes: A simulation study," *Sustainability*, vol. 12, no. 22, 2020.

[12] U.S. Department of Transportation Federal Highway Administration, "Next generation simulation (ngsim) vehicle trajectories and supporting data," Dataset, 2016, provided by ITS DataHub through Data.transportation.gov. Accessed 2024-07-01 from http://doi.org/10.21949/1504477.

[13] S. A. Elsagheer Mohamed and K. A. AlShalfan, "Intelligent traffic management system based on the internet of vehicles (iov)," *Journal of advanced transportation*, vol. 2021, no. 1, p. 4037533, 2021.

[14] A. Ait Ouallane, A. Bakali, A. Bahnasse, S. Broumi, and M. Talea, "Fusion of engineering insights and emerging trends: Intelligent urban traffic management system," *Information Fusion*, vol. 88, pp. 218–248, 2022.

[15] T. Zhou, Y. Sun, X. Wang, R. Dai, R. Cao, and X. Wu, "Proactive integrated traffic control to mitigate congestion at toll plazas," *IET Intelligent Transport Systems*, vol. 17, no. 8, pp. 1575–1587, 2023.

[16] Z. Pan, A. Sharma, J. Y.-C. Hu, Z. Liu, A. Li, H. Liu, M. Huang, and T. Geng, "Ising-traffic: Using ising machine learning to predict traffic congestion under uncertainty," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 8, 2023, pp. 9354–9363.

[17] K. Ramana, G. Srivastava, M. R. Kumar, T. R. Gadekallu, J. C.-W. Lin, M. Alazab, and C. Iwendi, "A vision transformer approach for traffic congestion prediction in urban areas," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 3922–3934, 2023.

[18] H. Yang, Z. Li, and Y. Qi, "Predicting traffic propagation flow in urban road network with multi-graph convolutional network," *Complex & Intelligent Systems*, vol. 10, no. 1, pp. 23–35, 2024.

[19] H. T. Abdelwahab and M. A. Abdel-Aty, "Artificial neural networks and logit models for traffic safety analysis of toll plazas," *Transportation Research Record*, vol. 1784, no. 1, pp. 115–125, 2002.

[20] L. Xing, J. He, Y. Li, Y. Wu, J. Yuan, and X. Gu, "Comparison of different models for evaluating vehicle collision risks at upstream diverging area of toll plaza," *Accident Analysis & Prevention*, vol. 135, p. 105343, 2020.

[21] L. Xing, L. Yu, O. Zheng, and M. Abdel-Aty, "Explore traffic conflict risks considering motion constraint degree in the diverging area of toll plazas," *Accident Analysis & Prevention*, vol. 185, p. 107011, 2023.

[22] W. Mo, J. Lee, M. Abdel-Aty, S. Mao, and Q. Jiang, "Dynamic short-term crash analysis and prediction at toll plazas for proactive safety management," *Accident Analysis & Prevention*, vol. 197, p. 107456, 2024.

[23] W. Xiang, C. Wang, X. Li, Q. Xue, and X. Liu, "Optimizing guidance signage system to improve drivers' lane-changing behavior at the expressway toll plaza," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 90, pp. 382–396, 2022.

[24] C. S. Bari, S. Chandra, A. Dhamaniya, S. Arkatkar, and Y. V. Navandar, "Service time variability at manual operated tollbooths under mixed traffic environment: Towards level-of-service thresholds," *Transport Policy*, vol. 106, pp. 11–24, 2021.

[25] C. S. Bari, S. Chandra, and A. Dhamaniya, "Service headway distribution analysis of fastag lanes under mixed traffic conditions," *Physica A: Statistical Mechanics and its Applications*, vol. 604, p. 127904, 2022.

[26] Y. Navandar, H. Golakiya, A. Dhamaniya, and D. Patel, "Vehicle class wise service time prediction models for tollbooths under mixed traffic conditions," in *ASCE India Conference 2017*. American Society of Civil Engineers Reston, VA, 2017, pp. 794–806.

[27] J. Liu, H. Zheng, L. Zha, E. Tian, and C. Peng, "Secure consensus for multiagent systems with hybrid cyber attacks: A multi-round-robin protocol-based approach," *Information Sciences*, vol. 677, p. 120878, 2024.

[28] Y. Nakatsukasa and J. A. Tropp, "Fast and accurate randomized algorithms for linear systems and eigenvalue problems," *SIAM Journal on Matrix Analysis and Applications*, vol. 45, no. 2, pp. 1183–1214, 2024.