Enhanced Delta-tolling: Traffic Optimization via Policy Gradient Reinforcement Learning

Hamid Mirzaei∗, Guni Sharon †, Stephen Boyles‡, Tony Givargis∗ and Peter Stone§
∗Department of Computer Science, University of California Irvine, Irvine, CA 92617, USA
†Department of Computer Science & Engineering, Texas A&M University, College Station, TX 77843, USA
‡Civil, Architectural and Environmental Engineering, The University of Texas at Austin, Austin, TX 78712, USA
§Department of Computer Science, University of Texas at Austin, Austin, TX 78712, USA
mirzaeib@uci.edu, gunisharon@gmail.com, sboyles@mail.utexas.edu, givarigs@uci.edu, pstone@cs.utexas.edu

Abstract—In the micro-tolling paradigm, a centralized system manager sets different toll values for each link in a given traffic network with the objective of optimizing the system’s performance. A recently proposed micro-tolling scheme, denoted Δ-tolling, was shown to yield up to 32% reduction in total travel time when compared to a no-toll scheme. Δ-tolling, computes a toll value for each link in a given network based on two global parameters: β which is a proportional parameter and R which controls the rate of toll change over time. In this paper, we propose to generalize Δ-tolling such that it would consider different R and β parameters for each link. A policy gradient reinforcement learning algorithm is used in order to tune this high-dimensional optimization problem. The results show that such a variant of Δ-tolling far surpasses the original Δ-tolling scheme, yielding up to 38% reduced system travel time compared to the original Δ-tolling scheme.

I. INTRODUCTION

Advancements in connected and automated vehicle technology present many opportunities for highly optimized traffic management mechanisms [1]. One such mechanism, micro-tolling, has been the focus of a line of recently presented studies [2, 3, 4]. In the micro-tolling paradigm, tolls can be charged on many or all network links, and changed frequently in response to real-time observations of traffic conditions. Toll values and traffic conditions can then be communicated to vehicles which might change routes in response, either autonomously, or by updating directions given to the human driver. A centralized system manager is assumed to set toll values with the objective of optimizing the traffic flow. Many methods for computing such tolls were presented over the last century most of which made very specific assumptions regarding the underlying traffic model. For instance, assuming that demand is known or fixed [5], assuming that links’ capacity is known or fixed, assuming that the user’s value of time (VOT) is homogeneous [6], assuming traffic follows specific latency functions [7], or assuming traffic patterns emerge instantaneously [8].

A recent line of work [2, 3] suggested a new tolling scheme denoted Δ-tolling. Unlike previous tolling schemes, Δ-tolling makes no assumptions regarding the demand, links’ capacity, users’ VOT, and specific traffic formation models. Δ-tolling sets a toll for each link equal to the difference (denoted Δ) between its current travel time and free flow travel time multiplied by a proportionality parameter β. The rate of change in toll values between successive time steps is controlled by another parameter R. Despite being extremely simple to calculate, Δ-tolling was shown to yield optimal system performance under the stylized assumptions of a macroscopic traffic model using the Bureau of Public Roads (BPR) type latency functions [9]. Moreover, Δ-tolling presented significant improvement in total travel time and social welfare across markedly different traffic models and assumptions. In fact, the simple working principle of Δ-tolling is what allows it to act as a model-free mechanism. Whereas the original Δ-tolling algorithm required a single β and R parameter for the entire network, the main contribution of this paper is a generalization of Δ-tolling to accommodate separate parameter settings for each link in the network. While conceptually straightforward, we demonstrate that doing so enables significant performance improvements in realistic traffic networks.

The increased representational power of Enhanced Δ-tolling compared to Δ-tolling does come at the cost of necessitating that many more parameters be tuned. A secondary contribution of this paper is a demonstration that policy gradient reinforcement learning methods can be leveraged to set tune these parameters effectively. Our detailed empirical study in Section V validates our claim that Enhanced Δ-tolling has the potential to improve upon the already impressive results of Δ-tolling when it comes to incentivizing self-interested agents to coordinate towards socially optimal traffic flows.

II. PROBLEM DEFINITION AND TERMINOLOGY

We consider a scenario where a set of agents must be routed across a traffic network given as a directed graph, \( G(V, E) \). Each agent \( a \) is affiliated with a source node, \( s_a \in V \), a target node, \( t_a \in V \), a departure time, \( d_a \), and a VOT, \( c_a \) (the agent’s monetary value for a delay of one unit of time).

Agents are assumed to be self-interested and, hence, follow the least cost path leading from \( s_a \) to \( t_a \). The cost of a path, \( p \), for an agent, \( a \), is a function of the path’s latency, \( l_p \), and tolls along it, \( \tau_p \). Formally, \( cost(p, a) = l_p \cdot c_a + \tau_p \). The value of
time, $c_a$, is assumed to be constant per agent. Although this assumption might not hold in real-world, it follows common practice in the transportation literature [3, 10, 11].

Since traffic is dynamically evolving, travel times and toll values might change over time, agents are assumed to continually re-optimize their chosen route. As a result, an agent might change its planned route at every node along its path. Each link in the network, $e \in E$, is affiliated with a dynamically changing toll value $\tau_e$ where for any path, $p$, $\tau_p = \sum_{e \in p} \tau_e$. Moreover, each link is affiliated with a latency $l_e$ representing the travel time on link $e$. Similar to $\tau_e$, $l_e$ is dynamically changing as a function of the traffic state.

The objective of the system manager is to assign tolls such that if each agent maximizes its own self interest, the system behavior will maximize social welfare. Denoting the latency suffered by agent $a$ as $l_a$, social welfare is defined as $\sum_a l_a \cdot c_a$\footnote{The tolls are not included in the calculation of social welfare, because we assume that toll revenues are transfer payments which remain internal to society.}. The system manager addresses the micro-tolling assignment problem which is defined as follows.

**Given:** $L^\tau$ - the vector of links’ latencies at time step $t$.

**Output:** $\tau^{t+1}$ - the vector of tolls applied to each link at the next time step.

**Objective:** Optimize social welfare.

**Assumption:** Agents are self interested i.e., they travel the least cost path $\arg \min_p \{\text{cost}(p, a)\}$ leading to their assigned destination $(t_a)$.

### III. BACKGROUND AND RELATED WORK

The approach suggested in this paper for solving the micro-tolling assignment problem builds on two previously presented algorithms: $\Delta$-tolling, and Finite Difference policy Gradient Reinforcement Learning (RL).

#### A. $\Delta$-tolling

It is well known that charging each agent an amount equivalent to the cost it inflicts on all other agents, also known as marginal-cost tolling, results in optimal social welfare [7].

Applying a marginal-cost tolling scheme, when differentiable latency functions are not assumed, requires knowing in advance the marginal delay that each agent will impose on all others. This, in turn, requires knowledge of future demand and roadway capacity conditions, as well as counterfactual knowledge of the network states without each driver.

$\Delta$-tolling [2, 3] was recently suggested as a model-free scheme for evaluating marginal cost tolling. It requires observing only the latency (travel time) on each link and makes no assumption on the underlying traffic model. $\Delta$-tolling involves charging a toll on each link proportional to its delay (the difference between observed and free-flow travel times). $\Delta$-tolling requires tuning of only two parameters: a proporationality constant ($\beta$), and a smoothing parameter ($R$) used to damp transient spikes in toll values.

Algorithm 1 describes the toll value update process of $\Delta$-tolling. For each link, $\Delta$-tolling first computes the difference ($\Delta$) between its current latency ($l_e^t$) and its free flow travel time (denoted by $T_e$). We use $i$ to denote the current time step. Next, the toll for link $e$ at the next time step ($\tau_e^{t+1}$) is updated to be a weighted average of $\Delta$ times beta and the current toll value. The weight assigned to each of the two components is governed by the $R$ parameter ($0 < R \leq 1$).

The $R$ parameter determines the rate in which toll values react to observed traffic conditions. When $R = 1$ the network’s tolls respond immediately to changes in traffic on the one hand but leave the system susceptible to oscillation and spikes on the other hand. By contrast, as $R \rightarrow 0$ the tolls are stable, but are also unresponsive to changes in traffic conditions.

Sharon et al. [2, 3] showed that the performance of $\Delta$-tolling is sensitive to the values of both the $R$ and $\beta$ parameters. Their empirical study suggests that values of $\beta = 4$ and $R = 10^{-4}$ result in the best performance. However, they do not present a procedure for optimizing these parameters and relay on brute force search for finding the optimal values through trial and error.

#### B. Policy gradient RL

Policy gradient RL is a general purpose optimization method that can be used to learn a parameterized policy based on on-line experimental data. While there are several different methods for estimating the gradient of the policy performance with respect to the parameters [12], one of the most straightforward, and the one we use in this paper, is Finite Difference Policy Gradient RL (FD-PGRL) [13] which is based on finite differences. In this subsection we review the methods and formulations presented in [13].

FD-PGRL is presented in Algorithm 2. Under this framework, the policy is parameterized using the parameter vector $\pi = [\theta_1, \ldots, \theta_N]^T$. The algorithm starts with the initial parameters $\pi^0 = [\theta^0_1, \ldots, \theta^0_N]^T$ (line 1). At each step $k$, the policy gradient is estimated by running a set of randomly generated policies $\Pi_k = \{\pi^k_1, \ldots, \pi^k_M\}$ (lines 5- 7) where each policy is defined as:

$$\pi^k_m = [\theta^k_1 - \delta^k_1, m, \ldots, \theta^k_N - \delta^k_{N, m}]^T,$$  \hspace{1cm} (1)

where $\delta^k_{n, m} \in \{-\epsilon_n, 0, \epsilon_n\}$. The generated policies in (1) are obtained by randomly changing each parameter from the previous policy by a small $\epsilon_n$, relative to $\theta_n$. The cost of each newly created policy, $\pi^k_m$, is observed and denoted by $c^k_m$ (lines 8- 9).

To estimate the policy gradient, the policy set in (1) is partitioned to three subsets (lines 11- 14) for each dimension depending on whether the change in the policy in that
Algorithm 2: Finite Difference Policy Gradient RL

1. \( \pi^0 \leftarrow [\theta_1^0, \ldots, \theta_N^0]^{T}; \)
2. \( k \leftarrow 0; \)
3. while improving do
   4. \( k \leftarrow k + 1; \)
   5. generate \( \Pi^k = \{\pi_k^1, \ldots, \pi_M^k\}, \)
   6. \( \pi_k^m = [\theta_{k+1}^1, \ldots, \theta_{k+n}^k, \theta_{k+n+1}^1, \ldots, \theta_{k+n+1}^k]^{T}; \)
   7. \( \delta_{n,m}^k \sim \text{Uniform}(-\epsilon_n, 0, \epsilon_n); \)
   8. for each \( m \in \{1, \ldots, M\} \) do
      9. \( c_m^k \leftarrow \text{run}(\pi_k^m); \)
    10. for each \( n \in \{1, \ldots, N\} \) do
        11. partition \( \Pi^k \) to
        12. \( \Pi_{-\epsilon,n}^k = \{\pi_k^m : \delta_{n,m}^k = -\epsilon\}, \)
        13. \( \Pi_{0,n}^k = \{\pi_k^m : \delta_{n,m}^k = 0\}, \)
        14. \( \Pi_{+\epsilon,n}^k = \{\pi_k^m : \delta_{n,m}^k = \epsilon\}; \)
        15. \( \tau_{c_m,n}^k \leftarrow \text{average}(c_m^k : \pi_k^m \in \Pi_{-\epsilon,n}^k); \)
        16. \( \tau_{c_m,n}^k \leftarrow \text{average}(c_m^k : \pi_k^m \in \Pi_{0,n}^k); \)
        17. \( \tau_{c_m,n}^k \leftarrow \text{average}(c_m^k : \pi_k^m \in \Pi_{+\epsilon,n}^k); \)
        18. if \( \tau_{c_m,n}^k < \tau_{c_m,n}^k < \tau_{c_m,n}^k < \tau_{c_m,n}^k \) then
           19. \( a_{n}^k \leftarrow 0; \)
        20. else
           21. \( a_{n}^k \leftarrow \tau_{c_m,n}^k - \tau_{c_m,n}^k; \)
        22. \( A^k = [a_1^k, \ldots, a_N^k]^{T}; \)
4. end of each step

The adjustment vector \( A^k \) is normalized and multiplied by a constant step size \( \eta \) to update the parameter vector at the end of each step \( k \) (lines 22-23).

Unlike other policy gradient methods that rely on within-episode reward signals to search for an optimal policy, or those in which the agent must learn the policy with no prior knowledge of a reasonably-performing starting policy (for example [14] and [15]), in the method employed in this paper, the policy is parameterized with a finite set of parameters and the overall system performance at each episode is optimized using an empirical estimate of the policy gradient based on finite differences. This approach is well-suited for the traffic optimization problem for two reasons. First, the agent can leverage an existing policy with reasonable system performance. Second, the agent is required to proceed towards the optimal policy only by slight changes of the policy parameters in contrast to approaches in which randomized exploration policies can be executed more freely. Our empirical study suggests that considering such slight changes results in a total cost that is within an acceptable bound. Furthermore, using other RL methods to learn actual tolls in real-time instead of \( \Delta \)-tolling parameters requires modeling traffic as Markov Decision Process which is a challenging task (see [16]).

IV. ENHANCED \( \Delta \)-TOLLING

We now present the main contribution of this paper, the Enhanced \( \Delta \)-tolling mechanism for solving the micro-tolling assignment problem. Enhanced \( \Delta \)-tolling extends the \( \Delta \)-tolling mechanism that is presented in Section III-A. \( \Delta \)-tolling uses two global variables that are used to set tolls on every link in the network. Since different links possess different attributes e.g., capacity, length, speed limit, etc., optimizing the \( \beta \) and \( R \) parameters per link can potentially yield greater benefits (higher social welfare, lower total travel time). However, doing so would require optimizing a set of \( 2|E| \) parameters instead of only two. Optimizing such a high dimensional function cannot be done efficiently in a brute force way.

This paper introduces Enhanced \( \Delta \)-tolling which extends \( \Delta \)-tolling by first, considering unique \( \beta \) and \( R \) parameters per link and second, incorporating policy gradient RL for optimizing these parameters.

In order to apply policy gradient RL (specifically FD-PGRL, as described in Section III-B), the traffic assignment policy that maps the current state of the traffic to the appropriate actions, which are assigning tolls to each link of the network, should be parameterized. Since the \( \Delta \)-tolling scheme, inherently implemented a policy that takes into account the real-time state of the traffic by assigning tolls proportional to the current links delay, we only use RL policy gradient method to optimize the performance metric at the end of each traffic cycle. Therefore, we define the cost to be the total travel time at the end of each day and consider the following three parametrization of \( \Delta \)-tolling:

\[
\pi_R = [\beta, R_1, \ldots, R_n] \\
\pi_\beta = [R, \beta_1, \ldots, \beta_n] \\
\pi_{R,\beta} = [R_1, \ldots, R_n, \beta_1, \ldots, \beta_n]
\]

The experimental results presented by Sharon et al. [3] suggest that there is some correlation between the optimally performing \( \beta \) and \( R \) values. However, no conclusions were presented regarding how they correlate and their individual impact on the convergence rate in a parameter tuning procedure.

As the relation between the \( \beta \) and \( R \) parameters remains unclear, we consider three variants of Enhanced \( \Delta \)-tolling based on the parameterized policies listed in (4):

- \( E\Delta \)-tolling\( _{\beta} \): this variant uses a global \( R \) parameter and link specific \( \beta \) parameters (\( |E| + 1 \) parameters in total). It
should perform well under the assumption that there is a correlation between the best performing $\beta$ and $R$ values and when FD-PGRL estimates the gradient over link specific $\beta$ parameters more accurately than it does for link specific $R$ parameters.

$E_{\Delta}$-tolling$_R$: this variant uses a global $\beta$ parameter and link specific $R$ parameters ($|E|+1$ parameters in total). It should perform well under the assumption that there is a correlation between the best performing $\beta$ and $R$ values and when FD-PGRL estimates the gradient over link specific $\beta$ parameters more accurately than it does for link specific $R$ parameters.

$E_{\Delta}$-tolling$_{\beta,R}$: this variant uses link specific $\beta$ and $R$ parameters ($2|E|$ parameters in total). It should perform best if there is no correlation between the best performing $\beta$ and $R$ values and if sufficient computation time is given (converging on $2|E|$ parameters is usually slower than on $|E|+1$).

V. EMPIRICAL STUDY

Our experimental evaluation focuses on real-life road networks. Traffic is evaluated using the cell transmission model (CTM) [17, 18] which is a discrete, explicit solution method for the hydrodynamic theory of traffic flow proposed in [19] and [20].

CTM is frequently used in dynamic traffic assignment. The time step used in this model is typically short, on the order of a few seconds. When used with Enhanced $\Delta$-tolling, this allows for a truly adaptive toll which can be updated based on observed traffic conditions.

A. Scenario specification

Demand model: demand is given as a trip table, where every entry is affiliated with a single agent ($a$) and specifies: a source node ($s_a$), a target node ($t_a$), and a departure time step ($t_a$).

Agent model: let $l^p_i$ be the sum of latency along path $p$ during time step $i$ and let $t^p_i$ be the sum of tolls along $p$ during time step $i$. When agent $a$ reaches a diverge node $n$ at time step $i$ all paths ($P_a$) leading from $n$ to destination $t_a$ are considered. Agent $a$ is assigned the minimal cost path i.e., $\text{arg min}_{P_a} \{ \tau^p_n + l^p_i \cdot c_a \}$.

B. Experiments and results

For running CTM we used the DTA simulator [21] implemented in Java. Whenever a vehicle is loaded onto the network, it is assigned a VOT randomly drawn from a Dagum distribution with parameters $\hat{a} = 22020.6, \hat{b} = 2.7926$, and $\hat{c} = 0.2977$, reflecting the distribution of personal income in the United States [22, 23].

The step size in FD-RPGS, $\eta$, is 0.4. The policy perturbation parameter, $\epsilon$ (see Line 2 in Algorithm 2) is set to 0.01 and the number of policy runs at each step, $M$, is 60 for all the experiments. These values presented best performance overall. Our empirical study focuses on three traffic scenarios:

Sioux Falls: [24] — this scenario is common in the transportation literature [25], and consists of 76 directed links, 24 nodes (intersections) and 28,835 trips spanning 3 hours.

Downtown Austin: [26] — this network consists of 1,247 directed links, 546 nodes and 62,836 trips spanning 2 hours during the morning peak.

Uptown San Antonio: this network consists of 1,259 directed links, 742 nodes and 223,479 trips spanning 3 hour during the morning peak.

The networks affiliated with each scenario are depicted in Figure 1. All of these traffic scenarios are available online at: https://goo.gl/SyvV5m

1) System performance: Our first set of results aims to evaluate the performance of the different variants of Enhanced $\Delta$-tolling, by comparing them with each other and basic $\Delta$-tolling. Figure 2 presents normalized values of total latency summed over all trips (top figure) and social welfare that is the summation of costs, i.e., latency times VOT, over all agents (bottom figure). The values are normalized according to the system’s performance when no tolls are applied. Table I presents the total latency and social welfare performance when applying no-tolls (representing the value of 1.0 in Figure 2).

The results present a clear picture in which $\Delta$-tolling improves on applying no tolls in both total latency and social welfare. $E_{\Delta}$-tolling$_R$ further improve the system’s performance and both $E_{\Delta}$-tolling$_{\beta}$ and $E_{\Delta}$-tolling$_{\beta,R}$ achieve the best performance.

The fact that $E_{\Delta}$-tolling$_R$ results in system performance which is similar to $E_{\Delta}$-tolling$_{\beta,R}$ suggests that there is a correlation between the best performing $\beta$ and $R$ values. The slight superiority of $E_{\Delta}$-tolling$_{\beta,R}$ over $E_{\Delta}$-tolling$_R$ is due to faster convergence which will be discussed later in this section. The fact that $E_{\Delta}$-tolling$_{\beta}$ performs worse than $E_{\Delta}$-tolling$_R$ suggests that policy FD-PGRL estimates the gradient over link specific $R$ parameters more accurately than it does for link specific $\beta$ parameters.

2) Convergence rate: applying $E_{\Delta}$-tolling to real-life traffic raises two concerns:

1) Convergence rate - the system should converge to a good solution with as few learning iterations as possible.

2) Worst case performance - during the learning process $E_{\Delta}$-tolling should perform at least as well as $\Delta$-tolling.
<table>
<thead>
<tr>
<th>Scheme</th>
<th>S. Falls</th>
<th>Austin</th>
<th>S. Antonio</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Δ-tolling</td>
<td>962,000</td>
<td>1,640,900</td>
<td>2,300,700</td>
<td>4,903,600</td>
</tr>
<tr>
<td>EΔ - β</td>
<td>943,076</td>
<td>1,619,928</td>
<td>2,257,830</td>
<td>4,820,834</td>
</tr>
<tr>
<td>EΔ - R</td>
<td>779,990</td>
<td>1,360,861</td>
<td>2,144,502</td>
<td>4,285,353</td>
</tr>
<tr>
<td>EΔ - β,R</td>
<td>777,469</td>
<td>1,415,094</td>
<td>2,162,006</td>
<td>4,354,569</td>
</tr>
</tbody>
</table>

TABLE II: Area under the convergence curves from Figure 3.

VI. DISCUSSION AND FUTURE WORK

The promising experimental results reported in Section V suggest that EΔ-tolling can have practical applications where traffic optimization is performed constantly and in real-time through manipulations to the $R$ and/or $β$ parameters. Nonetheless, implementation of EΔ-tolling raises several practical issues that must first be addressed.

Limitations: EΔ-tolling is limited in its convergence rate. General traffic patterns might change frequently, preventing EΔ-tolling from advancing in a promising direction. Practitioners must evaluate the convergence rate of EΔ-tolling versus the rate in which traffic patterns change in order to determine the applicability of EΔ-tolling in a specific network.

Assumptions: EΔ-tolling assumes that all agents traversing the network are self-interested and responsive to tolls in real time. Real-world scenarios might violate these assumptions and the trends observed in our results cannot be assumed in such cases.

Practical aspects of EΔ-tolling present many promising directions for future work. Since the convergence rate of EΔ-tolling plays an important role in determining its applica-
bility, one promising direction for future work is developing heuristics and utilizing advanced RL methods to guide the gradient exploration towards promising directions in order to facilitate faster learning.

Examining the effects of partial compliance to tolls is another promising direction. Building on recent study that examines the effects of partial compliance on similar micro-tolling schemes [27], studying the practical impacts of partial compliance on Enhanced $\Delta$-tolling is a promising direction to pursue.

VII. CONCLUSION

This paper introduced Enhanced $\Delta$-tolling, a micro-tolling assignment scheme that builds on the previously suggested $\Delta$-tolling scheme. The previously suggested $\Delta$-tolling scheme makes use of two global parameters, $\beta$ and $\Delta$, to tune the system for optimized performance (minimal total latency or maximal social welfare). Enhanced $\Delta$-tolling generalizes $\Delta$-tolling in two complementary ways. First, recognizing that different links in the network have different attributes (length, capacity, speed limit) Enhanced $\Delta$-tolling considers individual $\beta$ and $\Delta$ parameters per link. Second, given the resulting large parameter set (twice the number of links), Enhanced $\Delta$-tolling suggests a policy gradient RL approach for tuning these parameters. Experimental results suggest that tuning the $\Delta$ parameter while keeping a global $\beta$ parameter performs best overall (w.r.t total latency, social welfare, worst case performance, and convergence rates).

VIII. ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation under NSF grant number 1563652.

REFERENCES


