Black-Box Optimization of Parameterized Link-Dependent Road Tolling

John Fang

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 minimizing the total travel time of agents in the system. In this thesis, we extend previous work on the Enhanced $\Delta$-Tolling algorithm for updating link tolls. Specifically, we study the performance of this algorithm when the parameters are tuned with different black-box optimization techniques. We describe the relevant parameter tuning problem and its black box formulation, including constraints and assumptions. Next, we conduct several empirical studies on how to solve this problem, examining the optimization from different lenses of criteria. We show empirically that using CMA-ES for parameter tuning can yield up to 23% reduced system travel time compared to the previous best Enhanced $\Delta$-Tolling performance and up to 52% reduced system travel time compared to a baseline with no tolls. Furthermore, we explore using a hybrid technique combining CMA-ES and finite difference gradient descent, and show that this approach can sometimes achieve comparable performance to CMA-ES while reducing worst-case performance. Finally, we show that by reducing the population size for CMA-ES, we can reduce the number of function evaluations by at least 75% while still achieving system performance that is better than the previous best approach.

1 Introduction

Advancements of technology present several opportunities for optimized traffic management mechanisms. Automated traffic management systems can improve the time spent by drivers on a road by managing the flow of traffic through a road network. A recent line of research [1, 2] has focused on micro-tolling, a mechanism for automated traffic management. At a high level, micro-tolling influences the flow of traffic by charging tolls on some subset of the links in the road network. The tolls are set by a central system manager, which can make real-time observations and use this information to update toll prices. Drivers try to reach their destination using information such as toll values and latency on the road network to decide which route to take, while the system manager seeks to optimize the total system flow.

In the micro-tolling scheme, the question of interest is “how should the system manager set toll prices?”. A recent line of work by Sharon et al. [2, 3] has introduced a scheme called $\Delta$-Tolling, which updates a link’s toll price based on the difference between the link’s current travel time and its free-flow travel time. This scheme is simple to calculate, model-free, and makes minimal assumptions about the network and agents. Furthermore, this scheme only depends on two parameters, so brute-force search is a feasible tuning strategy. This work empirically showed that $\Delta$-Tolling yields good performance in terms of total travel time and social welfare across various different traffic models and assumptions.

The original $\Delta$-Tolling scheme uses two parameters, $\beta$ and $R$, which affects how reactive toll prices on all links are. In a follow up work, Mirzaei et al. [4] proposed Enhanced $\Delta$-Tolling, a generalization of $\Delta$-Tolling which allows $\beta$ and $R$ to vary across different road links. Since links in the network have different capacities, speed limits and demand, it may make sense to allow the toll values to change at different rates on each link and thus achieve better system performance. Compared to $\Delta$-Tolling, $E\Delta T$ has many more parameters, but empirical results showed that $E\Delta T$ was able to achieve even better optima than $\Delta$-Tolling. The key challenge with this extension is optimizing parameters in a way that yields the full benefits of $E\Delta T$.

Although Mirzaei et al. did some initial evaluation of $E\Delta T$ using finite-difference gradient descent, there are still many questions to be answered. This thesis further examines the $E\Delta T$ approach and addresses the following questions:
We also let $E$.

We make the following contributions: we evaluate $E$.

Also, each edge $e$.

This model defines two time-dependent values: 1) $\sum_{e \in E} e \cdot \text{c} = \text{the value of time (VOT)}$. In other words, the monetary value of one unit of time

Also, each edge $e \in E$ has two values associated with it, the toll on this edge $\tau e$ and the latency to travel this edge $l e$, both of which change over time. These definitions naturally extend to entire paths as the sum of values over all edges in the path. More formally, for a path $p$ we define $\tau p = \sum_{e \in p} \tau e$ and $l p = \sum_{e \in p} l e$.

We assume that each agent $a$ acts in its own interests by taking the path $p$ from $a.s$ to $a.t$ which minimizes the cost function $\text{cost}(p, a) = \tau p + a.c \cdot l p$. Since $\tau$ and $l$ change over time, agents recompute their best path as they travels through the network. Under this assumption, the problem for a traffic manager is to set tolls in such a way as to maximize the social welfare of the entire system. More formally, given the latency $l e$ of each edge at time step $t$, the traffic manager must find the toll values $\tau e$ for each edge at time step $t + 1$ which minimizes $\sum_{a \in A} a.c \cdot a.l$ where $a.l$ is the latency for $a$ to complete its path.

For empirical evaluations, we used simulation to model traffic through a road network. We used the dynamic traffic assignment (DTA) simulator [8] implemented in Java. This simulator implements the cell transmission model (CTM) [9] of traffic, which is a discrete, explicit solution method for the hydrodynamic theory of traffic flow proposed in [10, 11]. The cell transmission model models flow by dividing each link into a set of cells, where each cell’s length is the distance a vehicle travels in one time step at free-flow conditions. This model defines two time-dependent values: 1) $N_i(t)$, the maximum number of vehicles that cell $i$ can hold at time step $t$; 2) $Q_i(t)$, the maximum number of vehicles that can enter cell $i$ between time steps $t$ and $t + 1$. We also let $n_i(t)$ denote the number of vehicles in cell $i$ at time step $t$. Then $y_i(t)$, the number of vehicles that flow from cells $i - 1$ to $i$ at time step $t$, is bounded from above by 1) the number of vehicles in cell $i - 1$; 2) the amount of empty space in cell $i$; and 3) the capacity for flow into cell $i$. Formally, we can write this as

$$y_i(t) = \min\{n_{i-1}(t), N_i(t) - n_i(t), Q_i(t)\}$$
Then to update the number of vehicles in each cell, we use the following recursion

\[ n_i(t + 1) = n_i(t) + y_i(t) - y_{i+1}(t) \]

which states that the number of vehicles at cell \( i \) at time step \( t + 1 \) is the number of vehicles in that cell at the previous time step, plus the number of vehicles flowing in from cell \( i - 1 \) and minus the number of vehicles flowing out to cell \( i + 1 \).

3 Background

We build on existing work with \( \Delta \)-Tolling and E\( \Delta \)T. In this section, we formally define these two methods and discuss the results that have been achieved with these methods in previous papers.

3.1 \( \Delta \)-Tolling

\( \Delta \)-Tolling [3, 2] is a recently-introduced scheme for automatically adjusting the toll prices on a road network. This scheme does not make any assumptions about the demand, link capacities, users’ VOT and the underlying traffic model. It only requires observing the latency (travel time) on each link. At a high level, this scheme works by changing tolls on each link based on the difference between observed and free-flow travel times. The toll update is in the form of an exponential moving average, which places greater significance to recent values and smooths values to prevent transient spikes. \( \Delta \)-Tolling requires tuning of only two parameters: \( \beta \), a scaling constant for the difference between observed and free-flow travel times, and \( R \), a smoothing factor that controls the weight of past toll values in the exponential moving average.

Algorithm 1: Updating tolls according to \( \Delta \)-Tolling.

\begin{verbatim}
while true do
    for each link \( e \in E \) do
        \[ \Delta \leftarrow l_e - T_e \]
        \[ \tau_{e+1} \leftarrow R(\beta \Delta) + (1 - R)\tau_e \]
        \[ i \leftarrow i + 1 \]
end
\end{verbatim}
Algorithm 1 formally describes the toll update process of ∆-Tolling. For each time step $i$, ∆-Tolling first computes the difference ($\Delta$) between the current latency ($l_e^i$) and the free flow travel time ($T_e$) of each link $e$. Next, the new toll value ($\tau_e^{i+1}$) is set to a weighted sum of $\Delta \cdot \beta$ and the current toll value. The weight assigned to each component is controlled by the $R$ parameter, where $R \in (0, 1]$.

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

### 3.2 Enhanced ∆-Tolling

The original ∆-Tolling scheme only had two parameters, $R$ and $\beta$, which affected how all links in the network adjusted to latency. One could imagine that having per-link parameters may improve the algorithm’s performance: how quickly the toll on a particular link should change may depend on various factors such as the link’s usage, location, capacity, etc.

Mirzaei et al. [4] proposed Enhanced ∆-Tolling (E∆T), a generalization of ∆-Tolling where the network is not restricted to use the same $R$ and $\beta$ values in all links. They considered three potential sets of parameters:

$$
\begin{align*}
\theta_R &= [R_1, \ldots, R_n, \beta] \\
\theta_\beta &= [R, \beta_1, \ldots, \beta_n] \\
\theta_{R,\beta} &= [R_1, \ldots, R_n, \beta_1, \ldots, \beta_n]
\end{align*}
$$

where $n = |E|$ is the number of edges in the road network. $\theta_R$ has an $R$ parameter for every link but maintains a global $\beta$, while $\theta_\beta$ does the opposite. Both of these are generalized by $\theta_{R,\beta}$, which has $R$ and $\beta$ parameters for every link.

Although Enhanced ∆-Tolling is more flexible than the ∆-Tolling algorithm, the number of parameters that needs to be set scales linearly with the size of the network. Thus, finding a good set of parameters requires searching in a high-dimensional space. Mirzaei et al. previously did some initial evaluation of E∆T, using finite different gradient descent to optimize parameters. They found that $\theta_R$ and $\theta_{R,\beta}$ had comparable performance, and both performed better than $\theta_\beta$. In this thesis, we will use the $\theta_R$ definition, as it gives good performance and has approximately half as many parameters as $\theta_{R,\beta}$.

### 4 Optimizing Enhanced ∆-Tolling

In order to evaluate the benefits of E∆T, we must perform some parameter tuning. In this section, we demonstrate how parameter tuning can be framed as an optimization problem. Furthermore, we describe desirable properties for an optimization technique and describe the three techniques that we chose.

#### 4.1 Black-Box Formulation

In a generic optimization setting, given a function $f : \Theta \rightarrow \mathbb{R}$, one tries to find the element $\theta_{opt} \in \Theta$ that minimizes $f$. In the E∆T context, $\Theta$ is a multi-dimensional space representing all possible settings of E∆T parameters. However, instead of directly using $R$ and $\beta$ parameters, we define $\Theta$ in terms of $\alpha$ and $\beta$ parameters. That is, for $(\alpha_1, \ldots, \alpha_n, \beta) \in \Theta$, the corresponding E∆T parameters are $R_1 = 10^{\alpha_1}, \ldots, R_n = 10^{\alpha_n}, \beta = \beta$. Then $f(\theta)$ is the total travel time of all agents in the system under the E∆T parameters corresponding to $\theta$.

We do not have a closed-form expression for the function $f$, but can evaluate $f(\theta)$ given any particular $\theta$ using the simulator discussed in Section 5.1. Also, we do not know the form of $f$ with respect to $\theta$ (e.g. linear, quadratic, etc.). Hence, black-box optimization techniques are the natural tool for solving this problem.

#### 4.2 Optimization Desiderata

Given the generic formulation in section 4.1, there are many optimization techniques which could, in principle, apply to this problem. However, there are additional properties that would be desirable for this specific application which we consider.
- **Optimality**: the technique should find the minimal $f(\theta)$
- **Scalability**: the technique should handle large $D = |\Theta|$. Since the number of parameters for E△T scales linearly with the number of road links ($|E|$), we would like a technique that can still find good solutions even when searching in a high-dimensional space.
- **Data Efficiency**: the technique should evaluate $f$ as little as possible. In a real-world application of EAT, each evaluation would correspond to observing real traffic for a period of time. Hence, we would like to learn $\theta_{opt}$ without taking too long.
- **Robustness**: the technique should avoid evaluating parameters that give very poor values while searching for $\theta_{opt}$. In a real-world application, an evaluation with poor performance would correspond to having slow traffic for a period of time. In this thesis, we measure a technique’s “robustness” as the percentage increase of its worst-case performance over Δ-Tolling’s ($R = 10^{-4}, \beta = 4$) worst-case performance.

### 4.3 Optimization Techniques

Given the constraints described in section 4.2, we eliminated techniques from consideration which usually do not scale to high dimensions, did not apply to our problem, or had other limitations. We considered three optimization techniques, which are described briefly below.

#### 4.3.1 Finite-Difference Gradient Descent (FD)

Gradient descent is a technique that iteratively updates the function parameters to find the function’s minima. The technique begins with some initial parameters $\theta_0$. In each iteration, we obtain the gradient of the function $\nabla f$ at $\theta_0$, which can be thought of as a vector pointing in the direction of steepest descent. The current parameters are updated by moving in the direction of the gradient by some step-size $\eta$. We repeat this procedure until $f(\theta_i)$ converges. More formally, the update procedure is

$$\theta_{i+1} = \theta_i - \eta \cdot \nabla f(\theta_i)$$

Although the gradient cannot be computed analytically for E△T since we do not have a closed-form formula for $f$, there are methods for estimating the gradient. In this thesis, we will use the method of finite differences [12], which is a relatively straightforward way to estimate the gradient.

At each iteration, the finite-difference gradient is estimated by evaluating $f$ on a set of $k$ randomly generated perturbations of the current parameters. The perturbations are generated by either incrementing or decrementing each parameter by a small constant $\varepsilon$. To estimate the gradient, the perturbation set is partitioned into three subsets for each dimension depending on whether the change in that dimension is negative, positive or zero. The gradient direction is estimated based on the average performance of the perturbed policies in each group. For further details, please refer to [12].

E△T was evaluated with finite difference gradient descent in a previous work [4]. We include it here for completeness.

#### 4.3.2 Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES)

CMA-ES is an evolutionary strategy that has been successfully applied to many optimization problems [13]. The idea is to draw candidates from a multivariate normal distribution, which is parameterized with a mean $m$ and a covariance matrix $C$. After evaluating a population of candidates, the distribution is updated to fit the best-performing candidates and the next population is drawn from the updated distribution. A high-level description is provided below. For further details, see [5].

1. Select an initial step-size $\sigma_k$ and mean $m_k$ for the distribution and let $C = I$. Note that in the initial iteration, $\sigma_k$ is also the standard deviation of the multivariate normal distribution in all dimensions.

2. Randomly sample candidate parameters from a normal distribution. $\theta_i \sim \mathcal{N}(m_k, \sigma_k^2 C)$.

3. Evaluate performance $f(\theta_i)$ for all candidate parameters.
4. Sort all $\theta_i$ using the $f$-values. Use the top half best performing candidates to update $m_k$, $\sigma_k$ and $C_k$.

5. Go back to Step 2 and repeat.

4.3.3 Nelder-Mead

Finally, a third technique is Nelder-Mead, which maintains a set of $D + 1$ points that can be viewed as the vertices of a simplex, where $D = \dim(\Theta)$ is the number of $E\Delta T$ parameters. At each iteration, the algorithm tries to replace the worst-performing vertex with a better vertex using reflection, expansion or contraction with respect to the best side’s centroid, $c$. Note that the potential vertices all lie on the line connecting the worst-performing vertex and the centroid. If a new vertex cannot be found, then the entire simplex is shrunk. A simplified description of the algorithm is provided below. For further details, see [6, 7]

1. Evaluate performance for all points in the simplex.

2. Sort all points using the $f$-values. Denote $\theta_i$ to be the point with the $i$th-lowest $f$-value, so $\theta_1$ is the best-performing point and $\theta_{D+1}$ is the worst-performing point.

3. Compute the centroid $c$ of the best face.

$$c = \frac{1}{d} \sum_{1 \leq j \leq D} \theta_j$$

4. Reflection: create a new point $\theta_r$ by reflecting the worst point $\theta_{D+1}$ across the centroid $c$. If $\theta_r$ performs better than $\theta_2$ but not as well as $\theta_1$, then accept $\theta_r$ and go back to Step 2.

5. Expansion: if $\theta_r$ performs better than the best point $\theta_1$, then create a new point $\theta_e$ by moving $\theta_r$ further away from $c$. Accept whichever point performs better between $\theta_r$ and $\theta_e$, then go back to Step 2.

6. Contraction: if $\theta_r$ performs worse than the second-best point $\theta_2$, then there are two cases:

   (a) Outer Contraction: if $\theta_r$ performs better than the worst point $\theta_{D+1}$, then create a new point $\theta_c$ by moving $c$ towards $\theta_r$. If $f(\theta_c) \leq f(\theta_r)$ then accept $\theta_c$ and go back to Step 2

   (b) Inner Contraction: if $\theta_{D+1}$ performs better than $\theta_r$, then create a new point $\theta_c$ by moving $c$ towards $\theta_{D+1}$. If $f(\theta_c) < f_{D+1}$ then accept $\theta_c$ and go back to Step 2

7. Shrinking: if all previous steps have failed, shrink the simplex by moving all points towards the best point $\theta_1$. Then go back to Step 2

5 Empirical Study

In this section, we present empirical results from applying each optimization technique discussed in Section 4.3 on a few different traffic scenarios. We then analyze these results from the lens of each criteria mentioned in Section 4.2.

5.1 Traffic Simulation

For empirical evaluations, we simulate traffic through a road network using the CTM model described in Section 2. We simulate three traffic scenarios: Sioux Falls [14], downtown Austin [15], and San Antonio [1]. The road network size and the number of trips in each scenario is summarized in Table 1. The networks for these scenarios are shown in Figure 1 and the traffic scenarios are available online at https://goo.gl/SyvV5m
### Table 1: Statistics of the traffic scenarios used in the experiments

<table>
<thead>
<tr>
<th>Scenario</th>
<th># Links</th>
<th># Nodes</th>
<th># Trips</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sioux Falls</td>
<td>76</td>
<td>24</td>
<td>28,835</td>
<td>3 hours</td>
</tr>
<tr>
<td>Downtown Austin</td>
<td>1,247</td>
<td>546</td>
<td>62,836</td>
<td>2 hours, morning peak</td>
</tr>
<tr>
<td>San Antonio</td>
<td>1,259</td>
<td>742</td>
<td>223,479</td>
<td>3 hours, morning peak</td>
</tr>
</tbody>
</table>

5.2 Experimental Setup

We experiment with each of the techniques discussed in Section 4.3 to determine which work well for $E\Delta T$. We simulate the road network under different variations of $E\Delta T$ using the CTM simulator mentioned in Section 5.1. Following the work in [2], we set each vehicle’s value of time by randomly sampling from a Dagum distribution, a type of continuous probability distribution that is used to model income and wealth distribution. Specifically, we use a Dagum distribution with parameters $\hat{a} = 22020.6$, $\hat{b} = 2.7926$, and $\hat{c} = 0.2977$; this distribution models the distribution of personal income in the United States [16].

Each experiment involves 5 separate trials of optimization. For all techniques, we start optimization with all $\alpha$ parameters set to $-4$ and the $\beta$ parameter set to 4, since these values were found to work well for the original $\Delta$-Tolling [2]. During each function evaluation, all link tolls are initially 0.

For FD, we follow previous work by setting the step-size to $\eta = 0.4$ and the perturbation size to $\varepsilon = 0.01$. For CMA-ES, we experiment with different values for the initial standard deviation $\sigma$. FD and CMA-ES both optimize for 100 generations, where each generation uses a population of $k = 60$ candidate parameters. For Nelder-Mead, we initialize the simplex by randomly sampling from the normal distribution $N(-4, 0.2)$ for every $\alpha$ parameter; this approach performed better than the standard initialization method of selecting one point and generating all other points by moving a fixed distance along each dimension. Since Nelder-Mead does not have the same notion of “generations” as the other two techniques, we optimize for the same number of function evaluations. We use SciPy’s implementation of Nelder-Mead, using the adaptive parameter to set all other Nelder-Mead parameters.

For FD and CMA-ES, we measure the performance by generation. There are two main metrics of interest: 1) total travel time of all agents for the mean parameters. This is labeled “Total Travel Time” in the figures; 2) total travel time of all agents for the worst-performing parameters. This is labeled “Max Total Travel Time” in the figures. The former metric is useful for seeing the approximate performance for each generation, while the latter will be used for robustness analysis as it represents the worst-case performance that agents could experience during optimization. For Nelder-Mead, we instead plot the value of each function evaluation since this method does not have generations.

5.3 Optimality Analysis

Figure 2 shows the performance of the three techniques during optimization on the Sioux Falls network. These results show that CMA-ES can outperform FD in terms of final optima by a significant margin. This result may be because CMA-ES is able to increase the variance of its multivariate normal distribution during optimization, making it less susceptible to getting stuck in local optima. On the other hand, the finite different gradient descent method always samples points that are a fixed distance away from the current parameters and thus may not be able to escape a local optimum. The optimization curves for CMA-ES plateau at similar optima even with different initial standard deviation $\sigma$, which can also be explained by the fact that CMA-ES adjusts its normal distribution during optimization; since CMA-ES adapts the size of the distribution, it is robust to different initial parameters.

The Nelder-Mead method plateaus at an optimum that is slightly better than FD, but worse than CMA-ES. We hypothesize that this can also be explained by each method’s ability to explore the parameter space. Of the possible transformations in the Nelder-Mead method, only Expansion can increase the size of the simplex. Furthermore, this expansion is limited: only a single point is expanded at a time and the expanded point’s distance from the centroid is a constant multiple of the original point’s distance to the centroid.

We compare the best performing technique, $E\Delta T$ using CMA-ES, against a few baselines from previous works. First, we compare against the total travel time when applying no tolls and when optimizing $E\Delta T$ with FD. The latter had the best performance prior to this thesis. Additionally, since we created a framework
Figure 2: Optimization of $E \Delta T$ parameters with FD and CMA-ES on Sioux Falls. FD-OPT corresponds to finite difference gradient descent with $\eta = 0.4, \varepsilon = 0.01$. The variations of CMA-ES correspond to different values for the initial standard deviation $\sigma$. The curve shows the average over 5 runs and the shaded area shows the standard error. The x-axis is the number of optimization generations and the y-axis is the total travel time in hours.

Figure 3: Optimization of $E \Delta T$ parameters with Nelder-Mead on Sioux Falls. The curve shows the average over 5 runs and the shaded area shows the standard error. The x-axis is the number of function evaluations and the y-axis is the total travel time in hours.
Table 2: Total travel time (hours) achieved by different tolling schemes. **No tolls** and **Δ-Tolling** had 10 trials. **EΔT (Hybrid)** had 3 trials. All other schemes had 5 trials.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Sioux Falls</th>
<th>Austin</th>
<th>San Antonio</th>
</tr>
</thead>
<tbody>
<tr>
<td>No tolls</td>
<td>11,859</td>
<td>21,590</td>
<td>26,362</td>
</tr>
<tr>
<td>Δ-Tolling</td>
<td>9,620</td>
<td>16,409</td>
<td>23,007</td>
</tr>
<tr>
<td>Static tolls (CMA-ES)</td>
<td>7,400</td>
<td>8,725</td>
<td>20,434</td>
</tr>
<tr>
<td>EΔT (FD)</td>
<td>7,876</td>
<td>13,496</td>
<td>21,415</td>
</tr>
<tr>
<td>EΔT (CMA-ES)</td>
<td>6,968</td>
<td>10,390</td>
<td>20,527</td>
</tr>
<tr>
<td>EΔT (Hybrid)</td>
<td>7,062</td>
<td>10,821</td>
<td>20,740</td>
</tr>
</tbody>
</table>

for optimizing micro-tolling parameters, we introduce a third baseline that was not previously explored: optimized static tolls. In a static toll scheme, the central network manager sets a fixed toll value for each link in the network. Unlike dynamic tolls, toll values do not change in response to observations on the road network. We apply CMA-ES with initial standard deviation \( \sigma = 0.05 \) to find the best static tolls.

Table 2 presents the total travel time for each of these variations. The results show that EΔT with CMA-ES outperforms all other tolling schemes, achieving the lowest total travel time of 6,968 hours on Sioux Falls. This is a 41.2% improvement over the baseline with no tolls and an 11.5% improvement over the previous best performance achieved by EΔT with FD. Additionally, the results confirms our intuition that dynamic tolling can achieve better performance than static tolling. Surprisingly, static tolling performs quite well, outperforming most versions of Δ-Tolling on Sioux Falls and achieving the best performance on the larger networks. This result may be because EΔT starts with no tolls in each function evaluation, so the system may have poor performance while the toll values ramp up. On the other hand, static tolls can start at good toll values from the very beginning. We leave further investigation of static tolling for future work.

5.4 Robustness Analysis

Using the worst-case performance over the optimization, we can also compare each technique from a robustness standpoint. This data is provided in Table 3.

For CMA-ES, we can see from the cmaes-0.2 curve that setting the initial standard deviation too high causes this technique to sample candidate parameters that are far from the mean and may perform poorly. On the other hand, CMA-ES with a lower \( \sigma \) has a slightly higher max total travel time than FD, but is fairly robust. Hence, there is a trade-off in selecting an initial \( \sigma \) that is large enough to find the optima quickly, but also not be so large that CMA-ES samples parameters that perform very poorly. Of the standard deviation values we tested, \( \sigma = 0.05 \) seems to have the best trade-off between exploration and exploitation achieving optimal total travel time while having a reasonable max total travel time. The percentage increase of worst-case performance is 10.4% on Sioux Falls.

Figure 2 shows that Nelder Mead also evaluates poor-performing candidates and does not follow our robustness criteria. This is likely due to the initialization of the simplex, which can sample points further away from \( \alpha = -4, \beta = 4 \) than other techniques we tested. Hence, we might be able to optimize with Nelder-Mead with a lower worst-case performance by using a smaller initial simplex. However, the version of Nelder-Mead that we tested was not able to achieve better optima than the best version of CMA-ES. We would not expect to achieve better optima by shrinking the initial simplex, which decreases exploration. Thus, we did not perform further exploration with Nelder-Mead.

5.4.1 Hybrid Approach

Although CMA-ES achieves the best final optima, its max total travel time starts at a higher value than other methods such as FD. If these values are unacceptable for the optimization, then we would like to find a way to reduce the worst-case performance while still achieving similar end results. One possible approach to do this is to combine both optimization methods; that is, begin optimizing with FD, then switch over to CMA-ES after some number of generations. We perform some exploratory experimentation with this hybrid approach, optimizing with FD for the first 40 generations and switching to CMA-ES for the remaining 60
generations. Due to time constraints, we only perform 3 trials in each experiment. The optimization curves are presented in Figure 4 and numbers are shown in Table 3.

In terms of the two metrics we’ve been working with, this experiment suggests that it is possible to enjoy the benefits of both FD and CMA-ES: the worst-case evaluation for the hybrid method is only 4.1% worse than the ∆-Tolling baseline on Sioux Falls, but still achieves a total travel time similar to CMA-ES at the end. It may be possible to achieve better results, but we leave further exploration of the trade-offs for future work. However, one caveat of the hybrid approach is that the max total travel time spikes when switching between optimization methods. Although this technique is “robust” according to the definition in Section 4.2 since the max total travel time is never significantly worse than ∆-Tolling, it may evaluate parameters that are much worse than recent evaluations when switching between FD and CMA-ES. A sudden degradation in system performance may be a sub-optimal experience for drivers that have become accustomed to lowered total travel times.

5.5 Scalability Analysis

In addition to the experiments on Sioux Falls presented in the previous subsections, we ran the same experiments on the Austin and San Antonio networks to see if we observe similar behavior even on larger road networks with many more road links. We ran these experiments with FD and CMA-ES (σ = 0.05), opting not to include Nelder-Mead since CMA-ES performs better from both an optimality and robustness standpoint.

The optimization curves are shown in Figure 5 and the final performance is reported in Table 2. Across all scenarios considered, CMA-ES consistently outperformed FD and achieved a lower average total travel time. The relative improvement of CMA-ES over FD varies by scenario, achieving a 23.0% improvement on Austin but only a 4.1% improvement on San Antonio. In terms of robustness, the worst-case performance increased 8.2% on Austin and 7.7% on San Antonio compared to the ∆-Tolling baseline, which is comparable to the Sioux Falls results. We conclude that our results scales to higher dimensions, and CMA-ES is well-suited for
Figure 5: Optimization of $E\Delta T$ parameters with FD and CMA-ES on the larger networks, Austin and San Antonio. The format of each scenario’s graphs is the same as in Figure 2.

6 Data Efficiency Study

Although $E\Delta T$ has shown impressive results in terms of total travel time, the number of function evaluations is very high. If one evaluation of $f$ corresponded to measuring traffic for one day, then optimization would be impractical for real-world use. In this section, we attempt to reduce the number of function calls necessary while still learning good $E\Delta T$ parameters. We study two possible approaches for reducing the number of function evaluations necessary. First, we try simply reducing the population size parameter in our optimization techniques. Second, we try using surrogate models to approximate the objective function during optimization.

6.1 Population Tuning Approach

In previous experiments, we selected a population size of 60 in order to fairly compare against previous works, which used 60 points to approximate the gradient for finite difference gradient descent. However, it is possible that the population size is higher than necessary and we can achieve similar total system travel time using a
smaller population, especially since CMA-ES works differently from FD. Hence, one approach to improving data efficiency is to simply reduce the value of this parameter.

We experiment solely with CMA-ES, since it was the most successful technique from Section 5. For population tuning, the experimental setup is the same as in Section 5.2, except the initial standard deviation is $\sigma = 0.05$ and we experiment with different population sizes.

The total system travel time for CMA-ES with different population sizes are compared in Table 4. In the Sioux Falls and San Antonio scenarios, the performance degrades gracefully by a modest amount as the population size is decreased. Unexpectedly, the total system travel time for Austin decreases by a small amount with population sizes of 45 and 30. We currently do not have an explanation for why this occurs. These results show that $E\Delta T$ with CMA-ES using a population size of 15 outperforms $E\Delta T$ with FD across all three traffic scenarios. Hence, if one is willing to trade optimality for data efficiency, we can perform at least as well as the previous best result using only a quarter of the function evaluations. Based on the result trends, it seems likely that we could lower the population size even further and still achieve respectable system performance. However, due to time constraints we were not able to fully explore this parameter optimization and we leave this problem for future work.

### 6.2 Surrogate Model Approach

Another approach to reducing the number of function evaluations necessary is to utilize surrogate models. Suppose we had a function $f'$ that approximates $f$, where evaluating $f'$ is much cheaper than evaluating $f$. Then we could replace $f$ with $f'$ as the function we are trying to optimize and the optimization technique may be able to learn good parameters for $f$ without paying the cost of evaluation. Our general approach is as follows:

1. Optimize the objective function $f$ as usual.
2. After some number of generations, use some subset of the candidate parameters proposed so far and their corresponding evaluations as training data to create a model $f'$. The model could be a linear regression or a support vector machine, for example.
3. Continue optimization, except now optimizing $f'$ instead of $f$.
4. After some number of generations, go back to step 1.

The motivation behind step 4 is that the surrogate model $f'$ might only be able to approximate $f$ locally. During step 3, the candidate parameters may move to an area in the parameter space outside this neighborhood, where the $f$ and $f'$ evaluations differ greatly. Since our overall goal is to optimize the true objective function $f$ and not the surrogate model $f'$, we retrain $f'$ periodically during the optimization so that these do not differ too much.

The effectiveness of this approach depends heavily on how well we are able to find surrogate models $f'$ that approximate $f$. If these two functions are similar, then we can avoid paying some of the cost of evaluating $f$. However, if these functions are dissimilar, then we might evaluate parameters that perform well according to the surrogate model but perform very poorly on the true objective function, thus violating our robustness requirements. We do not make any assumptions about the underlying form of the objective function, and it is possible that $f$ is not well-behaved. Thus, it may be difficult to find an approximation $f'$.

Like in the population tuning approach, we again only consider improving the CMA-ES technique. One point worth noting is that CMA-ES updates using only the ordering of the candidate evaluations, instead of the actual value. Hence, for a set of candidate parameters, it is more important for the surrogate model $f'$ to rank the candidates in a similar order to $f$ than it is for the $f$ and $f'$ to have similar values.

We experimented with policies that follow the general approach outlined above. For step 2, we optimize the objective function $f$ for at least 5 generations before switching to the surrogate model. For training the model, we ran two experiments: using data from the previous 3 generations and using data from the previous 5 generations. The model is a support vector machine that tries to predict the $f(\theta)$ given some candidate parameters $\theta$. We select the kernel that best fits the training data, where each kernel is scored using Kendall’s tau coefficient [17], a metric for rank similarity. For step 4, the optimization switches from
Table 4: Total travel time (hours) achieved by $E\Delta T$ with CMA-ES with different population sizes for each generation. Each results is the average of 5 runs.

<table>
<thead>
<tr>
<th>$E\Delta T$ with CMA-ES (pop. size = 60)</th>
<th>Sioux Falls</th>
<th>Austin</th>
<th>San Antonio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6,968</td>
<td>10,390</td>
<td>20,527</td>
</tr>
<tr>
<td>$E\Delta T$ with CMA-ES (pop. size = 45)</td>
<td>7,019</td>
<td>10,343</td>
<td>20,588</td>
</tr>
<tr>
<td>$E\Delta T$ with CMA-ES (pop. size = 30)</td>
<td>7,078</td>
<td>10,158</td>
<td>20,637</td>
</tr>
<tr>
<td>$E\Delta T$ with CMA-ES (pop. size = 15)</td>
<td>7,409</td>
<td>10,409</td>
<td>20,763</td>
</tr>
</tbody>
</table>

Figure 6: Optimization of $E\Delta T$ parameters using CMA-ES and CMA-ES with surrogate models on Sioux Falls. $cmaes-0.05$ is the same as in Figure 2(a), and is included for comparison. The other two curves follow the policy described in Section 6.2. $cmaes-0.05$ with model (3) trains the model on 3 previous generations of data, while $cmaes-0.05$ with model (5) trains the model on 5 previous generations of data. The format of the graph is the same as in Figure 2 and each result is averaged over 5 runs.

These policy decisions were guided by experimenting with the data generated by the optimization experiments from section 5, as well as some trial and error. We used this approach to answer questions such as “to predict the $f$-values of a generation, how many previous generations should the model train on?” or “does a pairwise ranking approach perform better than a pointwise approach?”. We note that our approach uses hindsight to guide policy decisions, and hence would not translate to a real-world application: if we completed an optimization to obtain this information, then the problem is already solved and data efficiency improvements are pointless. However, we think these experiments are still valuable as they can function as an “upper bound” for this approach; if surrogate models do not work well even if we could utilize this information, then it will likely not perform better if we did not have this information.

Figure 6 compares the optimization curves for CMA-ES with surrogate models to vanilla CMA-ES. Of the 100 generations, approximately a quarter used the surrogate model while the remainder used the simulator. The CMA-ES with surrogate models curve is similar to normal CMA-ES during early generations, but performs progressively worse as the optimization continues. Near the end of the optimization, we see that the $cmaes-0.05$ with model (3) curve is not well-behaved. Whereas previous curves that we considered had a general downward trend until they plateaued, this curve has several spikes. This behavior suggests that the surrogate model’s approximation of the objective function is poor during these generations and CMA-ES is updating in a way that doesn’t minimize $f$, causing a degradation in performance. On the other hand, the $cmaes-0.05$ with model (5) does not have the same spiking behavior. This suggests that the behavior of the optimization curve depends on the quality of the model. However, it is not clear how to ensure that a
model is good enough to prevent the spiking behavior. Hence, this approach does not seem to necessarily be robust and the total travel time may sometimes jump to unacceptable levels.

In both experiments, this approach did not yield any improvement from a data efficiency standpoint. Of the 100 generations of optimization with surrogate models, approximately a quarter used the surrogate model. Since approximately ~75% as much data was used, we would expect the final performance for optimization with surrogate models to be better than normal optimization at around generation 75 for the surrogate models to provide a benefit from a data efficiency standpoint. However, we can see from Figure 6 that the normal optimization achieves much better performance by generation 75. Hence, given the same number of function evaluations, performing a normal optimization would be a better choice.

This behavior suggests that the objective function $f$ is difficult to approximate with the models $f'$ that we used, especially later into the optimization. From these experiments, we are not able to suggest a method to improve data efficiency based on a surrogate model approach, although there may exists other approaches that yield benefits.

7 Related Work

In this section, we first give a more extensive overview of road pricing and the significance of ∆-Tolling within this area. We also give a brief overview of different optimization strategies.

7.1 Road Tolling

Road tolling is a topic has been extensively studied over the past century and has the potential to reduce congestion in road networks. One tolling scheme worth noting is marginal cost tolling, which tolls a user the cost that they impose on other agents by choosing to take a certain road link. It is known that user equilibrium leads to social optimum under marginal cost tolling, but is effectively impossible to calculate. Hence, many works in the road tolling literature have aimed to approximate this scheme.

Previous works have explored several tolling schemes, which differed in the complexity of the traffic model and the assumptions about the traffic setting. The simplest traffic model is link delay functions, which assumes each link’s travel time is simply a function of its traffic volume. Several works have suggested adaptive pricing strategies that follow this assumption, such as [18, 19, 20]. Other works use more sophisticated traffic models such as a bottleneck or point-queue model, such as [21, 22]. However, all of these models do not capture more complex traffic behavior such as queue spill-backs, which occurs when a full road link affects other links that connect into it. Waller et al. [23] and Lo and Szeto [24] showed that congestion tolling on traffic models that cannot capture spatial congestion can actually be detrimental to system performance.

A more realistic traffic model is the hydrodynamic model, which treats traffic flow as a fluid described by differential equations. However, it is impractical to directly compute marginal costs due to "discontinuities in the flow model and congestion effects that transcend link boundaries", and this problem is NP-hard. In our work, we use this model to capture spatial congestion behavior in the road network. However, instead of trying to compute marginal costs, we instead evaluate the $E\Delta T$ scheme on this model.

A line of work by Sharon et al. [2, 3] introduced ∆-Tolling, a simple adaptive pricing scheme that updates tolls using travel time observations and two parameters $R$ and $\beta$. They showed formally that ∆-Tolling achieves system optimum in a macroscopic traffic model using the Bureau of Public Roads type latency functions [25] if $\beta$ is chosen correctly. Although they did not have similar theoretical results for the hydrodynamic model, they conducted empirical studies on the ∆-Tolling scheme in this model, and showed that it achieved good system performance compared to no tolls. Mirzaei et al. [4] continued this line of research by proposing $E\Delta T$, a generalization of ∆-Tolling where $R$ and $\beta$ parameters are allowed to vary in each road link. This work empirically showed that $E\Delta T$ performed better than ∆-Tolling, where the parameters were tuned using finite-difference gradient. This work continues to use $E\Delta T$, but differs from previous works in that it explores different methods for parameter tuning.

7.2 Black Box Optimization

Next, we discuss optimization strategies and why we chose certain methods. Additionally, we contrast our surrogate model approach to a previous work that combined CMA-ES with support vector machines.
7.2.1 Techniques

Black-box optimization techniques are algorithms that optimize an objective function $f : \Theta \rightarrow \mathbb{R}$. The algorithm can query the value of $f$ for a given set of parameters $\theta$, but does not make any assumptions about the form of $f$. This implies we cannot directly compute certain properties of the objective function such as the gradient, although they can still be approximated by evaluating several nearby parameters.

There are several broad categories of optimization techniques, which can roughly be categorized by what information is used in the optimization. The simplest category optimizes using the function values directly. These techniques working by perturbing parameters and comparing the new result against previous ones to decide if the method should decrease its step size or terminate. An example of this is the Nelder-Mead method [6]. Another category of optimization techniques uses the gradient of the objective function. An example of this is gradient descent, which moves the current parameters in the direction of the steepest decrease. A third category of techniques utilize the Hessian, the matrix of second-order partial derivatives. Given the exact Hessian, one can use the Newton’s method for optimization. This method did not apply to our problem as we do not have a closed-form equation for the objective function. If the Hessian can only be approximated, one can instead use quasi-Newtonian methods, which approximate the Hessian using the gradient and emulate Newton’s method of optimization. An example of a quasi-Newtonian method is BFGS [26]. We did not test this method as it involves computing and inverting a $D \times D$ matrix, which may be expensive given the high dimensionality of the E$\Delta$T optimization problem. Furthermore, each iteration requires both a gradient approximation and an additional line search, so this method would require more function evaluations than just gradient descent.

Finally, there are also evolutionary strategies. These strategies maintain a population of candidates and use their function evaluations to refine the quality of candidates. In each iteration, the candidates are all evaluated and the next generation is generated as some function of the best-performing candidates. An example of an evolutionary strategy is CMA-ES, which we used in this thesis.

7.2.2 Surrogate Models

Loshchilov et al. [27] explores the use of rank-based support vector machines as surrogate models within a CMA-ES optimization. In this paper, the surrogate model is used as a method of pre-screening candidates. This approach was tested on several closed-form functions and shown to achieve a peak speedup of approximately 4x when the problem dimensional was around 8-10. In contrast, in this thesis we attempted to use surrogate models on unknown functions with dimensionality ranging from 76 to 1,259. Also, we tried a more aggressive strategy of directly replacing the objective function, rather than pre-screening.

8 Conclusions and Future Work

In this work, we have evaluated several black-box optimization techniques and their extensions in the context of optimizing Enhanced $\Delta$-Tolling parameters. Empirical studies showed that CMA-ES was able to consistently achieve the best performance for E$\Delta$T across different network settings. Furthermore, we showed that a hybrid method combining CMA-ES and FD has potential for improving the robustness of the optimization with minimal sacrifices to optimality.

In our data efficiency study, we showed that we can greatly reduce the number of function evaluations while still achieving respectable system performance by tuning the population size of CMA-ES. Our study of optimization with surrogate models did improve upon normal optimization, which we hypothesize is because the models were not able to accurately approximate the objective function, especially in later generations.

For future work, we believe that some experiments discussed in this paper warrant additional investigation. Firstly, some initial experiments with the static tolling baseline suggests that this technique may yield even lower system travel time than E$\Delta$T on some traffic scenarios. Additional investigation is needed to understand if static tolls can replace or enhance the E$\Delta$T approach. Secondly, the hybrid optimization approach seems promising, and additional parameter tuning may yield better results. Finally, population tuning has been shown to be an effective strategy for improve data efficiency, but the full extent of its benefits has not been explored. Experimenting with even lower populations will allow us to gain a better understanding of the trade-off between data efficiency and optimality.
Acknowledgments

I would like to thank Dr. Peter Stone, Dr. Guni Sharon and Josiah Hanna for providing the opportunity to work on this research and their continuous guidance, Dr. Stephen Boyles and the transportation research group at UT Austin for their input throughout this project, and my fellow Turing Scholars for attending my practice thesis defense and giving me feedback.
References


