A k-best link tolling scheme

Debojjal Bagchi[1], Keya Li[1], and Qianqian Tong[1]

[1]Department of Civil, Architectural, and Environmental Engineering, University of Texas at Austin

March 19, 2024

Abstract

Tolling is an efficient technique for alleviating systematic traffic congestion. Commuters in general choose paths as per the principle of User Equilibrium (UE) which states that all used paths have equal and minimal travel time in each pair of origination-destination. However, from a system standpoint, the goal might be to minimize the total system travel time, as it reduces the average travel time of all users and has several system-level benefits. Adding tolls on certain links in a network increases the perceived travel time for each traveler as per their value of time thereby shifting commuters to be in a new user equilibrium. The tolls can hence be added in such a way that commuters choose paths that lead to system optima. An obvious way to do this is by tolling all links in the network. However, under the practical constraint of a limited number of toll booths, tolling all links is not viable. This project aims to heuristically find a k-best link tolling algorithm to achieve user equilibrium that minimizes total system travel time. To this end, a sensitivity analysis-based technique is adapted to solve the bi-level traffic assignment problem with non-convex constraint space. Network performances were explored in various scenarios, and conclusions were drawn from small-scale network examples, providing insights into larger-scale networks.

Keywords: tolling, system optima, network design, sensitivity analysis

1 Introduction

Ever-increasing traffic demands bring about multifaceted issues such as congestion and emissions, both from the user-driven and system-driven perspective. It is observed implementing tolling policies based on congestion mechanisms enables less average travel time on the network which in turn has several system-level benefits. Inspired by Wardropian principles, two perspectives of travel choice goals include User Equilibrium (UE) and System Optimum (SO) (Morandi, 2023). The UE and SO can be formally defined as follows.

Definition 1. ***User Equilibrium (UE):*** *A flow in the network such that every used path between the same origin and destination has equal and minimal travel time (Wardrop, 1952).*

Definition 2. ***System Optimum (SO):*** *A flow in the network that minimizes the total travel time for all travelers (Hearn et al., 2001).*

Since commuters are selfish, according to UE, traffic flows within a network would be distributed in a way that all used paths have equal and minimal travel time. Given the goal to reach system optima and minimize total system travel time, this project uses tolls as a viable option. When tolls are imposed, commuters would shift to a less-cost route that minimizes their perceived travel cost, and flows would be redistributed. The goal is to set the tolls in such a way that minimizes total system travel time to be as close to SO as possible. At this point, we formally define total system travel time and Tolled UE.

Definition 3. ***Total System Travel time (TSTT):*** *The total travel time for all travelers in the transportation network (Boyles et al., 2023; Hearn et al., 2001; Hearn and Ramana, 1998)*

Definition 4. ***Tolled User Equilibrium (Tolled UE):*** *A flow in the network such that users minimize their perceived travel cost, which is sum of the travel time and perceived travel time for toll paid.*

Table 1: Alternative toll pricing schemes

Toll Type	Objective	References
MINSYS	Minimize the total toll amount collected	Stefanello et al. (2017)
MINMAX	Minimize the maximum toll on a link	Hearn et al. (2001)
MINTB	Minimize the number of Toll booths	Shirazi and Aashtiani (2015)

Table 1 summarises three main objectives of pricing methods in the literature including minimizing the total (non-negative) toll revenue collected, minimizing the largest nonnegative toll collected, and minimizing the number of toll booths (MINSYS, MINMAX, MINTB respectively) in fixed demand networks (Hearn et al., 2001). All these pricing methods minimize different objectives given the flows are still at system optima. However, these pricing methods have no constraint on the number of toll booths that can be constructed. Specifically, the MINTB pricing strategy gives the minimum number of links that have to be tolled to be still at SO. However, in many practical networks, the output of the MINTB problem is significantly large (For instance 177 in the Stockholm network Hearn et al. (2001)). Due to limited infrastructure budgets, the tolling agency may only be able to construct a limited number of toll booths to achieve as close to minimal TSTT as possible. This project tries to address this significant gap in the literature. We now define our problem statement as follows.

Problem Definition (*k-best link tolling):* Given a network $G = (V, E)$ with V and E denoting the set of nodes and links and a scalar k denoting the number of links that can be tolled, the goal is to find which k links should be tolled and optimal tolls on those links such that the TSTT is minimized at tolled UE.

Proposition 1. *The k-best link tolling problem is NP-hard.*

Proof. Studies like Hearn et al. (2001) and Bai et al. (2004) which solve the MINTB pricing, find a minimum k^*, which is the number of links that must be tolled for the system to be in a system optima. One should observe that if $k \leq k^*$ then the system can never reach true system optimum with any amount of tolls on any k links. While for $k > k^*$, the goal of k-best link tolling and MINTB tolling are the same. Bai et al. (2010) has shown the MINTB problem to be NP-hard. The k-best link tolling problem is more generalized and hence is also NP-hard. □

Solving methods for the proposed models can be classified into two groups. Direct solvers include CPLEX (IBM, 2021), CVXPY (Diamond and Boyd, 2016), and other methods to solve problems of non-convex optimization. However, the k-best link tolling problem being NP-hard, heuristic algorithms should be used such as Local Search (Pirlot, 1996), Genetic Algorithm (Kumar et al., 2010), and Ant Colony optimization (Dorigo et al., 2006). The key to the success of the heuristics lies in determining the set of valid candidate links. Lee (2006) pointed out that the number of candidate links and link sets generated is critical for the performance of sensitivity analysis-based heuristics. As the number becomes larger, the method tends to sacrifice more efficiency and approach enumeration, while it can hardly provide precise search results if the number is too small.

In this project, we develop a local search-based heuristic that uses sensitivity analysis to find a search direction. To reach the system optimum as close as possible, we quantify the relationships between the objective function and link performance function after being tolled using theories from the Traffic Network Design Problem (TNDP) (Boyles et al., 2023). Zahedian et al. (2021) classified the problem of TNDP into three categories: building new links or expanding the capacity of links, determining the best location to add tolls to a network, and masking decisions based on three hierarchy processes. This project concerns about best locations of toll in terms of the whole system (Friesz, 1985).

The remainder of the report is organized as follows. Section 2 provides a mathematical formulation for the problem, Section 3 provides a local search heuristic algorithm, Section 4 provides examples of the working of proposed algorithm, Section 5 summarises the results, Section 6 adds some concluding remarks and Section 7 proposes some future directions on the work.

2 Mathematical Formulation

This section presents a new mathematical formulation for the best k-link tolling using appropriate decision variables. Table 3 summarizes the decision variables and notation.

Table 2: Mathematical formulation terminology

Notation	Description
x_{ij}	Flow on a link ij
x	Vector whose components are x_{ij}
y_{ij}	Binary decision variable indicating whether a link is tolled or not
β_{ij}	Toll on a link ij
β	Vector whose components are β_{ij}
t_{ij}	Link performance function on link ij (BPR Function)
X	Set of feasible assignments
k	The number of links to be tolled

The objective function in (1) minimizes the total system travel time in tolled user equilibrium. Constraint (2) ensures the flow is in user equilibrium after tolling. Constraint (3) ensures a non-negative tolling whenever a link is tolled. M is a large number. Constraint (4) allows at most k links to be tolled. Finally Constraint (5) and (6) allows the decision variables y_{ij} and β_{ij} to be binary and non-negative reals respectively.

$$\min \quad \sum_{(i,j)\in E} x_{ij} \cdot t_{ij}(x_{ij}) \tag{1}$$

$$\text{s.t.} \quad x \in arg\min_{x\in X} \sum_{(i,j)\in A} \int_0^{x_{ij}} (t_{ij}(x) + \beta_{ij})dx \tag{2}$$

$$0 \leq \beta_{ij} \leq M y_{ij} \qquad \forall ij \in E \tag{3}$$

$$\sum_{ij} y_{ij} \leq k \tag{4}$$

$$y_{ij} \in \{0, 1\} \qquad \forall ij \in E \tag{5}$$

$$\beta_{ij} \in \mathbb{R}^+ \qquad \forall ij \in E \tag{6}$$

The formulation however has a non convex decision space and is a bi-level optimisation problem. Specifically, Constraint (2) is an optimisation problem in itself, thereby making solution techniques computationally expensive. Further, the formulation does not have a unique solution. To see why, assume a network where a link has to be closed to reach system optima. Any toll value for that link above a threshold would have the same and minimal TSTT, which corresponds to a system optima, thereby giving multiple optima.

3 Heuristic Algorithm

As seen in Section 2, the mathematical formulations do not scale well for large-scale real-world networks with many links. This necessitates the use of heuristics to find near-optimal solutions in a reasonable time. One should note that the k-best link tolling problem is essentially a network design problem, hence sensitivity analysis techniques are expected to perform well. To this end, a local search-based heuristic is proposed. The procedure updates two sets TolledSet and TollValues until a convergence criterion is reached or the time budget is depleted. The set TolledSet contains the links which are tolled and the set TollValues contains the respective tolls. Subsection 3.1 describes the convergence criterion. The sensitivity analysis-based search direction is explained in Subsection 3.2. Finally, Section 3.3 summarizes the complete local search procedure.

3.1 Convergence Criterion

For the purpose of local search, the initial solution is set as the user equilibrium without any tolls. The TSTT at the initial solution is denoted as $TSTT_{UE}$. Throughout the local search process, adding tolls on links redirects commuters so that they form a new tolled user equilibrium. The TSTT at any tolled UE is denoted as $TSTT_{UE\_Tolled,\beta}$, where β is the toll vector. The minimal total system travel time possible in any network is at its system optima. Hence,

the TSTT reduces from $TSTT_{UE}$ to TSTT at system optima ($TSTT_{SO}$) throughout the iterations. Now we define, **Toll Gap** as a convergence criterion:

$$TG(\beta) = \frac{TSTT_{UE\_Tolled,\beta} - TSTT_{SO}}{TSTT_{UE} - TSTT_{SO}} \tag{7}$$

When there is no toll in the network, the overall performance wouldn't be improved, $TG = 1$; in the best case scenario, Tolled UE and SO will coincide, $TG = 0$. In our model, the value of TG is tracked and evaluated at each iteration. One should note, TG might not be 0 at an optimal solution. This is because, by tolling only a few links it is possible that TSTT can never reduced to reach system optima on large-scale networks. Even in that case, TG reduces over the iterations.

3.2 Search Direction

In this subsection, we describe the sensitivity analysis that helps to find a search direction. Given the sets TolledSet and TollValues, the goal is to find links that should be removed from the TolledSet. To do so, sensitivity analysis is used. A link ij is picked from the TolledSet and the toll value is perturbed by a small amount. The change in link flows of all links $kl \in E$ when a small change is done in tolling on link ij is computed. This turns out to be a network design problem with small changes in the link performance function. Following the methodology outlined in Boyles et al. (2023), a user equilibrium is solved with the following changes give the solution to the link sensitivities:

- Original link performance functions are replaced by affine link performance functions with slope equal to the derivative of the original link performance function at the current tolled user equilibrium solution, and intercept equal to the derivative of the link performance function with respect to the toll.
- The equilibrium bushed for each origin is fixed at the bushes for the current tolled equilibrium solution.
- All entries in the OD matrix (demands) are zero.

Once the values of $\frac{\partial x_{kl}}{\partial \beta_{ij}}$ are computed using the procedure outlined above, the change in TSTT due to the small change of tolls on link ij is computed. This is the ij^{th} component of the gradient of TSTT with respect to change in tolls on links. Following the derivations in Boyles et al. (2023) we have,

$$\frac{\partial TSTT}{\partial \beta_{ij}} = \sum_{kl \in E} \left(\frac{\partial x_{kl}}{\partial \beta_{ij}} t_{kl}(x_{kl}, \beta_{kl}) + x_{kl} \frac{\partial t_{kl}(x_{kl}, \beta_{kl})}{\partial x_{kl}}\right) + x_{ij} \frac{\partial t_{ij}}{\partial \beta_{ij}} \tag{8}$$

The sign of $\frac{\partial TSTT}{\partial \beta_{ij}}$ has the information on whether further tolling link ij is beneficial or not. If the sign is negative, adding a small amount of tolls would reduce the TSTT and hence tolling link ij is preferable. Otherwise, link tolling link ij is counterproductive and should be removed from the tolled set. Algorithm 1 (GradComp) summarises the entire procedure. It takes input a link ij and its tolls β_{ij} and returns $\frac{\partial TSTT}{\partial \beta_{ij}}$. Line 1 updates the link performance functions to be affine and sets the demands to be zero, Line 2 solves the user equilibrium to get link sensitivities and Line 3 computes the $\frac{\partial TSTT}{\partial \beta_{ij}}$.

Algorithm 1 GradComp

Input: ij : Tolled link, β_{ij} : Toll

1: Update link performance functions as described earlier and set demand as 0

2: Solve UE within new network and compute `Link_Sensitivities` $\leftarrow \frac{\partial x_{kl}}{\partial \beta_{ij}}$

3: `Grad_component` $\leftarrow \frac{\partial TSTT}{\partial \beta_{ij}} = \sum_{(kl \in A)} \left(\frac{\partial x_{kl}}{\partial \beta_{ij}} t_{kl}(x_{kl}, \beta_{kl}) + x_{kl} \frac{\partial t_{kl}(x_{kl}, \beta_{kl})}{\partial x_{kl}}\right) + x_{ij} \frac{\partial t_{ij}}{\partial \beta_{ij}}$

Output: `Grad_component`

3.3 Local Search

This subsection formalizes the entire local search procedure. Two sets TolledSet and TollValues are initialized as two random k links and a toll value[1] of 1 is given to each of them. Another set ProspectiveLinks is initialized with all links in the network. The set ProspectiveLinks contains the list of links that have not been evaluated for tolling locally. A link ij from TolledSet is picked in order. Two cases are possible. Either tolling link ij improves TSTT or not.

(a.) In case imposing a toll on link ij does not improve TSTT, the link ij is removed from TolledSet. Since link ij has been evaluated for tolling, it is removed from the set ProspectiveLinks. This reduces the cardinality of TolledSet by 1. Hence, a new link has to be added. To do so, a random link is picked from ProspectiveLinks and added to TolledSet.

(b.) Alternatively, imposing a toll on link ij improves TSTT. In that case, the toll value on link ij is slightly increased. A random number is drawn from the uniform distribution from 0 to U. The value of U is reduced over the iterations from 10 to 1 as closer to optimal solution less drastic changes should be done. The TSTT is computed after adding the tolls. If the TSTT is reduced the link is tolled with the additional amount, else the original toll is kept. This way of cumulatively adding tolls is per the fact sensitivity analysis only gives a local analysis.

If all the k links in the TolledSet have a negative component gradient, it means tolling all links in TolledSet improves TSTT. This could be a local minima as any iterations of the local search after this point would only adjust toll values and not change tolled links. To get out of such local minima, once a local minima is hit, a few iterations of local search are done till the toll values stabilize. Then new links are searched that have a gradient component lesser than the greatest negative gradient component of links in the tolled set. Once such a link is found, it replaces the original link in the tolled set, and the process is repeated once the toll values stabilize.

Further, once the ProspectiveSet is empty, it means all the links have been evaluated locally. The set ProspectiveSet is reset to have all links. Algorithm 2 (LocalSearch) presents the pseudocode for the entire local search procedure. Line 1–2 initializes the TolledSet, TollValues, and ProspectiveLinks sets. Line 3 initializes the variable `max_neg_grad_comp` capturing the least negative gradient component as 0. Line 4 starts a while loop until convergence criterion is hit or allowed time is depleted. Line 5–20 captures each iteration of the local search. Line 5 starts a for loop for each link in the TolledSet, Line 6 checks the gradient component, and Line 7 checks whether the gradient component is positive or negative until all links in the TolledSet have a negative gradient component. Thereafter the 'if' condition turns false for the least negative gradient component. Lines 7–14 try to add a small toll on a link if it should be tolled, else line 15–17 removes links that shouldn't be tolled from the TolledSet and ProspectiveLinks set. If a link was removed from TolledSet, Line 18–20 adds a random link from ProspectiveLinks to TolledSet. If ProspectiveLinks becomes empty, Line 19 resets it. Line 21–23 is an 'if' loop which only turns true if all the gradient components of links in the TolledSet become negative, hence a local minima. In that case, `max_neg_grad_comp` is adjusted such that a link with a more negative component of gradient will knock off the least negative component of gradient from the TolledSet. Furthermore, every time a local minima is hit the upper limit of the uniform distribution is reduced by 1 in Line 23.

[1]Toll values throughout this report is in time units. Once can get the equivalent toll in money units by multiplying the toll value by value of time.

Algorithm 2 LocalSearch

Input: G: Graph, k: Number of links to be tolled

Set ProspectiveLinks as all links
Set TolledSet as k random links and Set TollValues[link] as a toll of 1 unit on these links.
`max_neg_grad_comp` = 0
while $TG > \epsilon$ and allowed time is not depleted **do**
    for each `link` in TolledSet **do**
        δ = GradComp(`link`, TollValues[link])
        if $\delta <$ `max_neg_grad_comp` **then**
            Add δ to GradComp
            Add a toll randomly uniformly between 0 to U to `link` (Cumulative)
            Solve the Tolled UE
            if TG is reduced **then**
                Update `Toll_Values[link]`
            else
                Remove `link` from TolledSet
        else
            Remove `link` from TolledSet
            Remove `link` from ProspectiveLinks
        if a link was removed from TolledSet **then**
            If ProspectiveLinks = {}, set one of the tolls as 0, and set ProspectiveLinks as all links
            Add a new `link` randomly from ProspectiveLinks to TolledSet
    if |GradComps| = k and No improvement in TG for 10 iterations **then**
        `max_neg_grad_comp` = max(GradComps)
        Reduce U by 1. If $U < 0$, set $U = 1$

Output: `TolledSet`, `TollValue`

4 Example on the Braess Network

In this section, we provide an example of the working of the Algorithm described in Section 3. For this purpose, the Braess Network is used as shown in Figure 1(left). BPR functions were used as link performance functions. The values of the parameters of the BPR function is given in Appendix 1. Three iterations are shown in Table 3 when the input k=2.

Table 3: Iteration Process

Run	Results
Initial	TOLLEDSET: [(3,4), (2,4)] (randomly picked) TOLLVALUES: (3,4): 1, (2,4): 1 TSTT = 553.994, TG = 1 Prospective Links: [(1,2), (1,3), (2,3), (3,4), (2,4)]
Iteration 1(a)	TOLLEDSET: [(3,4), (2,4)] TOLLVALUES: (3,4): 1, (2,4): 1 Selected Link ij and Toll: (3,4), 1 Sensitivities: $[\frac{\partial x_{12}}{\partial \beta_{34}}, \frac{\partial x_{13}}{\partial \beta_{34}}, \frac{\partial x_{23}}{\partial \beta_{34}}, \frac{\partial x_{24}}{\partial \beta_{34}}, \frac{\partial x_{34}}{\partial \beta_{34}}]$ = [0.006, -0.006, -0.076, 0.0839, -0.0839] Gradient Component: 0.839 Positive, this link should be removed New TOLLEDSET: [~~(3,4)~~, (2,4)] New Prospective Links: [(1,2), (1,3), (2,3), ~~(3,4)~~, (2,4)] TG = 1
Iteration 1(b)	TOLLEDSET: [(2,4), (2,3)] TOLLVALUES: (2,4): 1, (2,3): 1 Selected Link ij and Toll: (2,3), 1 Sensitivities: $[\frac{\partial x_{12}}{\partial \beta_{23}}, \frac{\partial x_{13}}{\partial \beta_{23}}, \frac{\partial x_{23}}{\partial \beta_{23}}, \frac{\partial x_{24}}{\partial \beta_{23}}, \frac{\partial x_{34}}{\partial \beta_{23}}]$ = [-0.07, 0.07, -0.15, 0.076, -0.076] Gradient Component: -4.30 Negative, this link works Add Toll to link (2,3): 0.19 (Toll for link (2,3): 1.19) TG = 0.80
Iteration 2(a)	TOLLEDSET: [(2,4), (2,3)] TOLLVALUES: (2,4): 1, (2,3): 1.19 Selected Link ij and Toll: (2,4), 1 Sensitivities: $[\frac{\partial x_{12}}{\partial \beta_{24}}, \frac{\partial x_{13}}{\partial \beta_{24}}, \frac{\partial x_{23}}{\partial \beta_{24}}, \frac{\partial x_{24}}{\partial \beta_{24}}, \frac{\partial x_{34}}{\partial \beta_{24}}]$ = [-0.07, 0.07, 0.077, -0.084, 0.084] Gradient Component: 5.08 Positive, this link should be removed New TOLLEDSET: [~~(2,4)~~, (2,3)] New Prospective Links: [(1,2), (1,3), (2,3), ~~(3,4)~~, ~~(2,4)~~] TG = 0.31
Iteration n	**...**

5 Results

The algorithm was tested on two networks with different numbers of OD pairs and a number of links tolled. The networks used in the study are shown in Figures 1 and 2. BPR Functions were used as link performance functions for each network. The values of the parameters of the BPR function are provided in Appendix 1. The ϵ value was set as 0.01 for the experiments. Figures 1 and 2 also show how the value of Toll Gap and TSTT changed over the iterations.

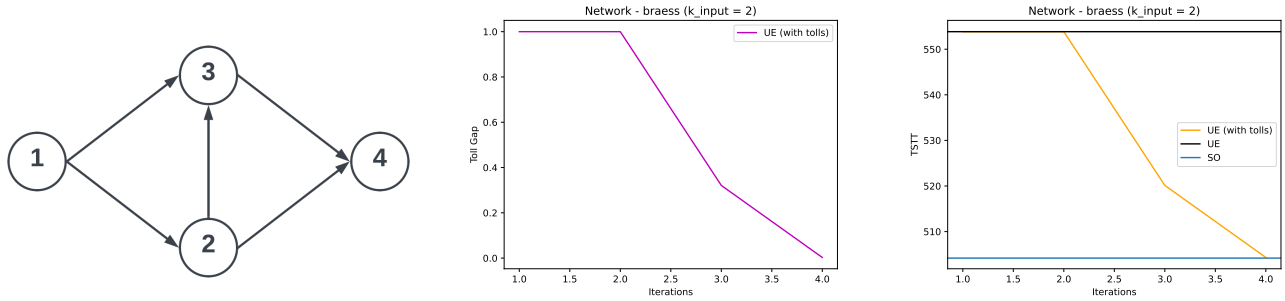

Figure 1: 1 OD Pair, $d^{14} = 6$, k=2 (Tolls are (2,3): 7.2, (1, 3): 0)

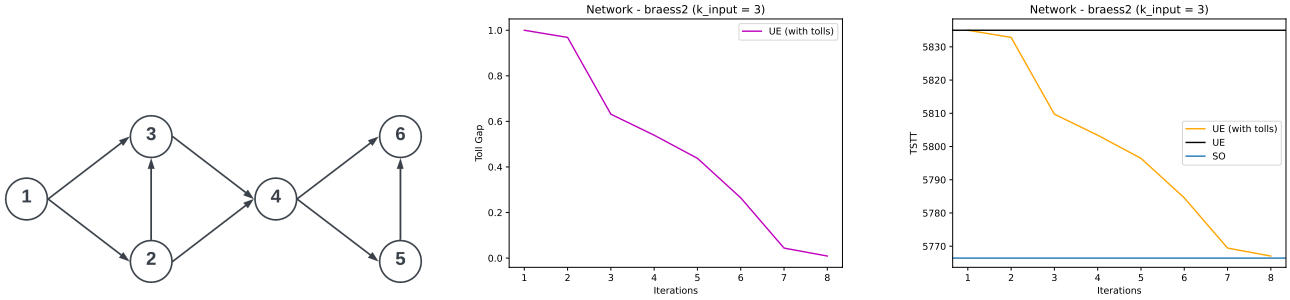

Figure 2: 2 OD Pair, $d^{14} = 6$, $d^{16} = 18$, k=3 (Tolls are (4,6): 32.87, (2, 3): 26.16, (1,2): 0)

A total of 9 experiments were run on the two networks with different number of nodes, links, OD paris and k value. The results are summarised in Table 4. The performance can be compared in the number of iterations, run-time, and TG. The algorithm was stopped if either the convergence criterion was reached or a time budget of 30s was depleted. 7 out of 9 test cases converged in less than 10s. Further, when $k = 1$, both the networks suffered from stalling. This can be explained by the fact that with smaller number of links to be tolled, it is difficult to reach system optima. With lesser values of k it becomes increasingly difficult to find the appropriate link and amount of toll. As the number of links allowed to be tolled increases, there are higher chances that the optimal links to be tolled would be in the TolledSet thereby decreasing the time to convergence.

Table 4: Test Cases Summary

Name	Nodes	links	OD Pairs	k	Its.	Time	TG
Braess	4	5	1	1	7	0.8	< 0.01
Braess	4	5	1	2	4	0.5	< 0.01
Braess	4	5	1	3	4	0.6	< 0.01
New Braess	6	8	1	1	120	30.7	0.17
New Braess	6	8	1	2	9	4.01	< 0.01
New Braess	6	8	1	3	10	6.02	< 0.01
New Braess	6	8	2	1	120	30.7	0.17
New Braess	6	8	2	2	9	4.54	<0.01
New Braess	6	8	2	3	9	5.65	< 0.01

Note: *Time*: Time in seconds to reach a TG of 0.01

6 Conclusions

In this project, we addressed the problem of identifying the k best links for tolling and determining toll amounts in the network to achieve minimum TSTT. It is practical in real-world scenarios where the tolling authority has a limited budget for building toll booths. To an extent, it bridges the gap of three main conventional methods for solving the toll problem.

The mathematical formulation was found to be a bi-level problem and the non-convex constraint space made it difficult to be solved by non convex programming methods. Inspired by network design problems in Farahani et al. (2013), we introduced sensitivity analysis to identify optimal links for tolling and its effects on the whole networks while also determining appropriate tolls through a gradual increase on these links. This effective heuristic was tested in two network structures with different OD pairs and k values. It demonstrated remarkable efficiency by converging in less than 10 iterations within a span of 7-8 seconds.

In conclusion, we developed a heuristic to effectively solve a non-convex constraint space and successfully tackled the bi-level challenge by adopting sensitivity analysis. Our approach to this kind of problem provides insights into solving bi-level transportation problems when UE has to be satisfied as a constraint. Besides, it can serve as a basis for further exploration and potential implementation in more complex and larger scenarios with constrained budgets for toll booth construction.

7 Future Direction

For potential advancements in our approach, we identify several key directions for improvement, aiming to ensure a faster and more sophisticated solution to similar problems in future work and extend it to large-scale networks.

First, tolls are currently assigned randomly from a uniform distribution, resulting in unstable and often inflated final toll values. While this method ensures TSTT is reduced and overall network performance at the system level, the toll values may be adversely affected by individual overpricing. Therefore, a simulated-annealing type technique is recommended to derive tolls. A strategic method can calculate toll values more accurately and close to the actual optimal value, benefiting both the tolling authority and individuals. For determining tolled links, we check links one by one despite their non-cumulative effects. Reevaluation is required on this front. Further, the algorithm proposed might significantly increase the number of iterations in large-scale networks, which is time-consuming. Addressing this discrepancy can streamline the identification of optimal tolled links.

In addition, in the existing method, to solve the sensitivity problem, we determined UE solution by enumerating all possible paths and solving a system of linear equations, which proves slow for slightly larger networks. It may even go wrong when there exist unused paths. A transition to a bush-based method is proposed for larger-scale networks and multiple OD demands. Given the heuristic nature of our solution, a heuristic to find sensitivities can also be implemented. It can align seamlessly with the original heuristic strategy we proposed, expediting the identification of tolled links and tolls.

The future directions discussed in this section would extend the applicability of our method in real-world scenarios and might be a significant contribution to existing literature.

References

Bai, L., D. W. Hearn, and S. Lawphongpanich (2004, September). Decomposition techniques for the minimum toll revenue problem. Networks 44(2), 142–150.

Bai, L., D. W. Hearn, and S. Lawphongpanich (2010). A heuristic method for the minimum toll booth problem. Journal of Global Optimization 48(4), 533–548.

Boyles, S. D., N. E. Lownes, and A. Unnikrishnan (2023). Transportation network analysis. Volume I, Version 0.92.

Diamond, S. and S. Boyd (2016). Cvxpy: A python-embedded modeling language for convex optimization. The Journal of Machine Learning Research 17(1), 2909–2913.

Dorigo, M., M. Birattari, and T. Stutzle (2006). Ant colony optimization. IEEE computational intelligence magazine 1(4), 28–39.

Farahani, R. Z., E. Miandoabchi, W. Y. Szeto, and H. Rashidi (2013). A review of urban transportation network design problems. European journal of operational research 229(2), 281–302.

Friesz, T. L. (1985). Transportation network equilibrium, design and aggregation: key developments and research opportunities. Transportation Research Part A: General 19(5-6), 413–427.

Hearn, D., M. Yidirim, M. Ramana, and L. Bai (2001). Computational methods for congestion toll pricing models. In ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No.01TH8585), Oakland, CA, USA, pp. 257–262. IEEE.

Hearn, D. W. and M. V. Ramana (1998). Solving congestion toll pricing models. Springer.

IBM (2021). IBM ILOG CPLEX. https://www.ibm.com/analytics/cplex-optimizer.

Kumar, M., D. M. Husain, N. Upreti, and D. Gupta (2010). Genetic algorithm: Review and application. Available at SSRN 3529843.

Lee, Y.-J. (2006). Transit network sensitivity analysis. Journal of Public Transportation 9(1), 21–52.

Morandi, V. (2023). Bridging the user equilibrium and the system optimum in static traffic assignment: a review. 4OR, 1–31.

Pirlot, M. (1996). General local search methods. European journal of operational research 92(3), 493–511.

Shirazi, M. and H. Z. Aashtiani (2015). Solving the minimum toll revenue problem in real transportation networks. Optimization Letters 9, 1187–1197.

Stefanello, F., L. S. Buriol, M. J. Hirsch, P. M. Pardalos, T. Querido, M. G. Resende, and M. Ritt (2017). On the minimization of traffic congestion in road networks with tolls. Annals of Operations Research 249, 119–139.

Wardrop, J. G. (1952). Road paper. some theoretical aspects of road traffic research. Proceedings of the institution of civil engineers 1(3), 325–362.

Zahedian, S., A. Nohekhan, and K. F. Sadabadi (2021, September). Dynamic toll prediction using historical data on toll roads: Case Study of the I-66 Inner Beltway. Transportation Engineering 5, 100084.

Appendix 1

Values of parameters of the BPR function in Braess Network are shown in Table 5.

Table 5: Parameters of the BPR function in Braess Network

Init node	Term node	Capacity	Length	Free Flow time	Alpha	Beta	Speed limit	Toll	Type
1	2	1	1	1	10	1	60	0	0
1	3	1	1	50	0.02	1	60	0	0
2	3	1	1	10	0.1	1	60	0	0
2	4	1	1	50	0.02	1	60	0	0
3	4	1	1	1	10	1	60	0	0

Note: FIRST THRU NODE:1

Values of parameters of the BPR function in New Braess Network are shown in Table 6.

Table 6: Parameters of the BPR function in New Braess Network

Init node	Term node	Capacity	Length	Free Flow time	Alpha	Beta	Speed limit	Toll	Type
1	2	1	1	1	10	1	60	0	0
1	3	1	1	50	0.02	1	60	0	0
2	3	1	1	10	0.1	1	60	0	0
2	4	1	1	50	0.02	1	60	0	0
3	4	1	1	1	10	1	60	0	0
4	6	1	1	1	10	1	60	0	0
4	5	1	1	50	0.02	1	60	0	0
5	6	1	1	10	0.1	1	60	0	0

Note: FIRST THRU NODE:1