A K-best Edge Tolling Scheme

Debojjal Bagchi, Keya Li, Qianqian Tong

CE392C Course Project

November 28, 2023

TEXAS
The University of Texas at Austin

Outline

- Introduction
- Methodology - Motivation
- Mathematical Formulation
- Algorithm
  - Convergence Criterion
  - Initial Solution
  - Improving Direction
  - Local Search
- Results
- Summary
- Future Directions

Introduction

- User Equilibrium (**User-driven**): Every used path between the same origin and destination has equal and minimal travel time
- System Optimal (**System-wide**): Total travel time for all travelers within the transportation network is minimized

(a) Individual Level

(b) Government Level

- SO strategies can help distribute traffic flows, reduce delay, and mitigate congestion
- Imposing tolls is one of methods to achieve SO at the premise of UE

Motivation

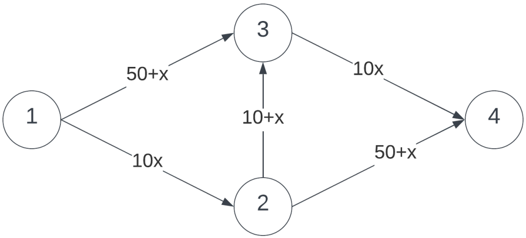


Figure: Braess Network (d^{14}=6)

- **UE solution**:
  $h^{[1,3,4]} = h^{[1,2,4]} = h^{[1,2,3,4]} = 2$
  $c^{[1,3,4]} = c^{[1,2,4]} = c^{[1,2,3,4]} = 92$
  TSTT $= 92*6 = 552$
- **SO solution**:
  $h^{[1,3,4]} = h^{[1,2,4]} = 3,\ h^{[1,2,3,4]} = 0$
  $c^{[1,3,4]} = c^{[1,2,4]} = 83$
  TSTT $= 498$
- **Toll construction**:
  One can add tolls of $t'_{ij}(x_{ij}) * x_{ij}$ to each link get SO even people choose routes as per UE.

But can we toll all links in real life?

Motivation

- **Literature Review** (Hearn et al, 2001)
  - MINSYS: Minimizing the total non-negative toll revenue collected
  - MINMAX: Minimizing the largest nonnegative toll collected
  - MINTB: Minimizing the number of toll booths

Motivation

- **Literature Review** (Hearn et al, 2001)
  - MINSYS: Minimizing the total non-negative toll revenue collected
  - MINMAX: Minimizing the largest nonnegative toll collected
  - MINTB: Minimizing the number of toll booths
- **Problem Statement**
  Given a number k, how to identify which k links be tolled at what toll so that the system is closest to System Optimum in terms of total system travel time

Motivation

- **Literature Review** (Hearn et al, 2001)
  - MINSYS: Minimizing the total non-negative toll revenue collected
  - MINMAX: Minimizing the largest nonnegative toll collected
  - MINTB: Minimizing the number of toll booths
- **Problem Statement**
  Given a number k, how to identify which k links be tolled at what toll so that the system is closest to System Optimum in terms of total system travel time
- **Project Objective**
  Minimizing total system travel time at equilibrium with only k number of constructed toll stations

Definition: Define **Tolled UE** as the UE solution when certain links are tolled.

Mathematical Formulation

Throughout this presentation, whenever we say "tolls", we refer to tolls in "time" value.
Formulation: [1]

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in E} x_{ij} \cdot t_{ij}(x_{ij}) && && (1) \\
\text{s.t.} \quad & x \in arg \min_{x\in X} \sum_{(i,j)\in A} \int_0^{x_{ij}} (t_{ij}(x) + \beta_{ij}) dx && && (2) \\
& 0 \leq \beta_{ij} \leq M y_{ij} && \forall ij \in E && (3) \\
& \sum_{ij} y_{ij} \leq k && && (4) \\
& y_{ij} \in \{0, 1\} && \forall ij \in E && (5)
\end{aligned}
$$

This is a bi-level problem with non-convex decision space.

[1] Equation 1 minimizes TSTT, Constraint 2 promises UE, β_{ij} is link toll, y_{ij} is dummy binary variable to identify which link is tolled, M is very large number

Algorithm

Hence, a heuristic based method is proposed. We need to define three things to develop an algorithm

- Convergence Measures
- Improvement Direction
- Initial Solution

Algorithm - Convergence Criterion

- We already have code for solving traffic assignment, replace the link performance functions $t(x)$ with $t(x) + xt'(x)$ and solve UE. This would give the SO solution. Compute the $TSTT_{UE}$ and $TSTT_{SO}$.
- Define tolled equilibrium by changing link performance functions $t(x)$ with $t(x) + \beta$ and solve UE. Call this solution UE_Tolled. We want to make $TSTT_{UE\_Tolled} - TSTT_{SO}$ close to zero.
- For a random solution of tolls β, Define Toll Gap (TG) as:

$$\mathtt{TG} = \frac{TSTT_{UE\_Tolled} - TSTT_{SO}}{TSTT_{UE} - TSTT_{SO}}$$

When there is no tolling $\mathtt{TG} = 1$, in best case scenario, $\mathtt{TG} = 0$

Sensitivity Analysis: The effect of change of tolls on each link can be found using sensitivity analysis.

Algorithm - Improving Direction

Sensitivity Analysis: The effect of change of tolls on each link can be found using sensitivity analysis.

Recall that we have derived the sensitivity analysis "formulaes" for changes in link performance functions.

- Compute the `link.slope` as derivative of the link performance function at current **tolled** eqm.
- Compute `link.constant` derivative of the link performance function w.r.t the parameter changed. This is just 1 or 0, based on whether a link is tolled or not. **(why?)**

Algorithm - Improving Direction

Sensitivity Analysis: The effect of change of tolls on each link can be found using sensitivity analysis.

Recall that we have derived the sensitivity analysis "formulaes" for changes in link performance functions.

- Compute the `link.slope` as derivative of the link performance function at current **tolled** eqm.
- Compute `link.constant` derivative of the link performance function w.r.t the parameter changed. This is just 1 or 0, based on whether a link is tolled or not. **(why?)**

Solving the UE with the these linear link performance function and zero demand gives the link sensitivities as "link flow solution"

Algorithm - Improving Direction (Contd.)

Now, Given a network, a `tolled_link` and corresponding `toll_value`, find if the “tolling scheme” improves TSTT:

Algorithm - Improving Direction (Contd.)

Now, Given a network, a `tolled_link` and corresponding `toll_value`, find if the "tolling scheme" improves TSTT:

This is the exact network design problem where there is no cost of building infrastructure!

Algorithm - Improving Direction (Contd.)

Now, Given a network, a `tolled_link` and corresponding `toll_value`, find if the "tolling scheme" improves TSTT:

This is the exact network design problem where there is no cost of building infrastructure!

- For a given `tolled_link` (ij) and corresponding `toll_value` (y_{ij}) find the `link_sensitivities` $(\frac{\partial x_{k\ell}}{\partial y_{ij}})$ on each link.
- Use the `link_sensitivities` to compute the grad_component using: $\frac{\partial f}{\partial y_{ij}} = \left\{ \sum_{(k,\ell)\in A} \left(\frac{\partial x_{k\ell}}{\partial y_{ij}} t_{k\ell}(x_{k\ell}, y_{k\ell}) + x_{k\ell} \frac{\partial t_{k\ell}(x_{k\ell}, y_{k\ell})}{\partial x_{k\ell}} \right) + x_{ij} \frac{\partial t_{ij}}{\partial y_{ij}} \right\}$

Algorithm - Improving Direction (Contd.)

Now, Given a network, a `tolled_link` and corresponding `toll_value`, find if the "tolling scheme" improves TSTT:

This is the exact network design problem where there is no cost of building infrastructure!

- For a given `tolled_link` (ij) and corresponding `toll_value` (y_{ij}) find the `link_sensitivities` ($\frac{\partial x_{k\ell}}{\partial y_{ij}}$) on each link.
- Use the `link_sensitivities` to compute the grad_component using:
  $\frac{\partial f}{\partial y_{ij}} = \left\{ \sum_{(k,\ell)\in A} \left(\frac{\partial x_{k\ell}}{\partial y_{ij}} t_{k\ell}(x_{k\ell}, y_{k\ell}) + x_{k\ell} \frac{\partial t_{k\ell}(x_{k\ell}, y_{k\ell})}{\partial x_{k\ell}} \right) + x_{ij} \frac{\partial t_{ij}}{\partial y_{ij}} \right\}$
  - If $\frac{\partial f}{\partial y_{ij}} \geq 0$, this tolling scheme does not improve TSTT
  - Else: This tolling scheme improves TSTT

Call this algorithm $\textsc{GradComp}(ij, y_{ij})$ that returns the $\frac{\partial f}{\partial y_{ij}}$ for a toll y_{ij} on link ij.

Algorithm - Local Search

Algorithm LocalSearch

```
Set Prospective_Links as all links
Set Tolled_Set as k random links and Set Toll_Values[link] as a toll of 1 unit on these links.
while TG > ε do
    for each link in Tolled_Set do
        if GradComp(link, Toll_Values[link]) < 0 then
            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 Tolled_Set
        else
            Remove link from Tolled_Set
            Remove link from Prospective_links
        If Prospective_links = {}, set one of the tolls as 0, and set Prospective_Links as all links
        Add a new link randomly from Prospective_links to Tolled_Set
```

Example

Initialisation

- Tolled Set : [(3, 4), (2, 4)]
- Toll Values : {(3, 4): 1, (2, 4): 1}
- TSTT = 553.994; TG = 1
- Prospective Links: [(1, 2), (1, 3), (2, 3), (3, 4), (2, 4)]

TEXAS
The University of Texas at Austin

Initialisation

- Tolled Set : [(3, 4), (2, 4)]
- Toll Values : {(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:

- Tolled Set : [(3, 4), (2, 4)]
- Toll Values : {(3, 4): 1, (2, 4): 1}
- Selected link and toll : (3, 4) and 1
- Sensitivities: [0.006, -0.006, -0.076 0.0839, -0.0839]
- Gradient Component: 0.839 (Positive)
- Tolled Set : [~~(3, 4)~~, (2, 4)]
- Prospective Links: [(1, 2), (1, 3), (2, 3), ~~(3, 4)~~, (2, 4)]
- Tolled Set : [(2, 4), (2, 3)]

Example

Iteration 2:

- Tolled Set : [(2, 4), (2, 3)]
- Toll Values : {(2, 4): 1, (2, 3): 1}
- Selected link and toll : (2, 3) and 1
- Sensitivities: [-0.07, 0.07, -0.15, 0.076, -0.076]
- Gradient Component: -4.30 (Negative)
- Add toll to (2,3) : 0.19 (toll becomes 1+0.19)
- TG: 0.80
- Tolled Set : [(2, 4), (2, 3)]

Example

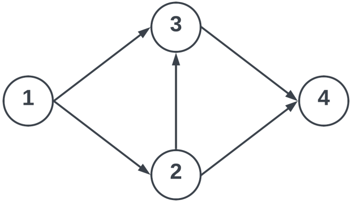
Iteration 3:

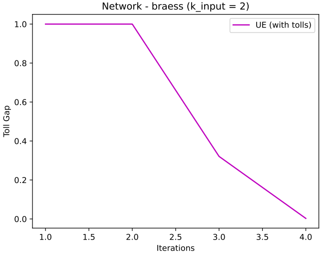
- Tolled Set : [(2, 4), (2, 3)]
- Toll Values : {(2, 4): 1, (2, 3): 1.19}
- Selected link and toll : (2, 3) and 1.19
- Sensitivities: [-0.07, 0.07, -0.15, 0.076, -0.076]
- Gradient Component: -4.33 (Negative)
- Add toll to (2,3) : 5.67 (toll becomes 1.19+5.67)
- TG: 0.31
- Tolled Set : [(2, 4), (2, 3)]

Next iteration (2,4) would be selected, at that point prospective links would be empty, the toll for (2,4) would be set as zero.

Results

Results





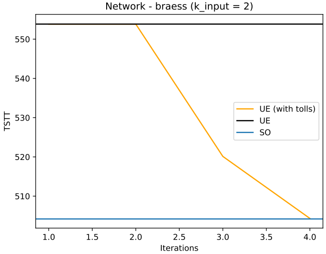
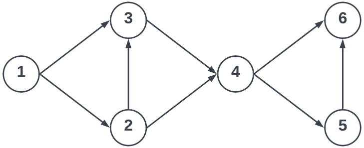
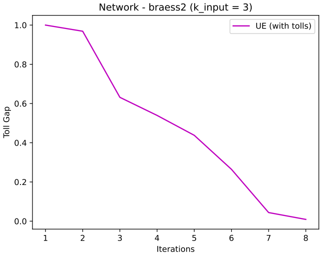


Figure: 1 OD Pair, k=2 (Tolled links are {(2,3):7.2 and (2,4):0})

Results





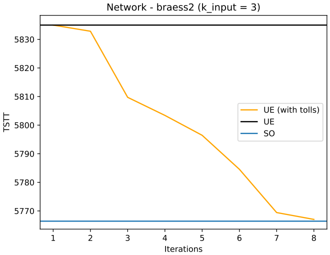


Figure: 2 OD Pair, k=3 (Tolls are (4, 6): 32.87, (2, 3): 26.16, (1, 2): 0 ; TG = 0.009)

Results (Summary)

A maximum run-time of 30s was allowed. 7 out of 9 test cases converged in less than 10s.

Name	Nodes	Edges	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

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

Summary

- We solved the problem of finding k best links in a network to be tolled to reduce total system travel time. We also found the toll values.
- We proposed an optimization program formulation for the problem and showed the problem has non convex constraint space.
- We proposed a simple heuristic that uses sensitivity analysis to find links to be tolled and find the toll values by slightly increasing the tolls on these links.
- We tested the heuristic on small networks and the heuristic converged in less than 10 iterations (7-8s)

Future Directions

- The tolls are set randomly from a uniform distribution, this can be improved using a simulated-annealing type technique.
- We check links one-by-one even though the effects are not cumulative, this should be addressed.
- We currently find the UE solution to the sensitivity problem by finding all possible paths and solving a system of linear equation, this process is too slow for slightly larger network like SiouxFalls. Techniques like MSA cant be used, a bush based method should be used.
- Given the heuristic nature of the solution, a heuristic to find the sensitivities can be implimented.

Thank You

Questions?