Μελέτη Περιπτώσεων στη Λήψη Αποφάσεων

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

BY NC ND

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης

ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σημείωμα Αδειοδότησης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Πατρών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Algorithms for Transport Optimization
Theory and Practice

Christos Zaroliagis

zaro@ceid.upatras.gr

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

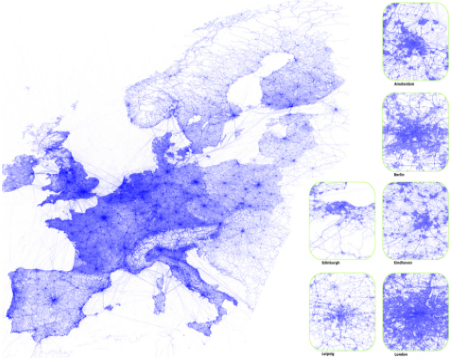
Dept. of Computer Engineering & Informatics
University of Patras, Greece

CTI "DIOPHANTUS"
computer technology institute & press

Computer Technology Institute & Press
"Diophantus"

Transport Optimization Problems

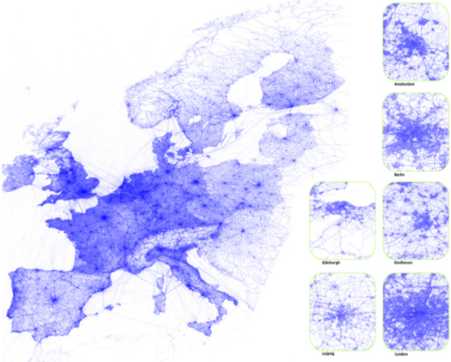
Public transportation networks



Road networks

Transport Optimization Problems

Public transportation networks

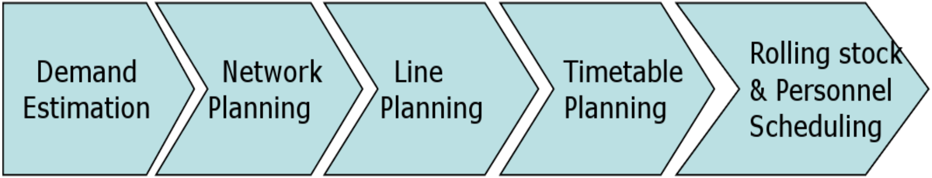


Road networks

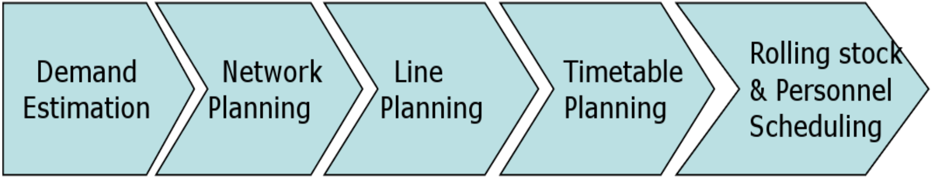
Common characteristic: large/huge scale

Outline

Public Transportation Planning

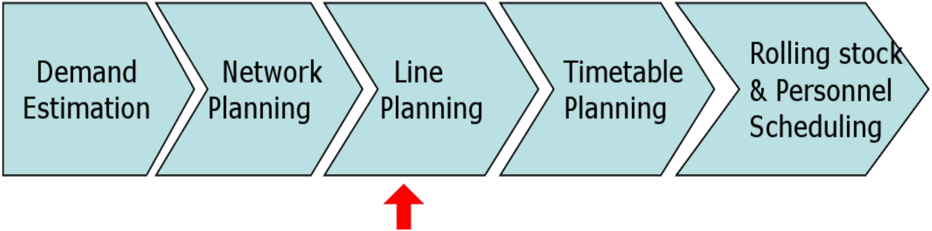


Public Transportation Planning



- **This talk:** Railways

Public Transportation Planning



- **This talk:** Railways
- **Line Planning**
  - Determine the set of train lines (routes) along with their frequencies
  - Typically, a line pool is provided

Line Planning Problem (I)

- **Railway Network Infrastructure**
  governed by a **network operator** (**NOP**) & represented as a digraph $G = (V, L)$
  - $V \longleftrightarrow$ stations or junctions of rail tracks
  - $L \longleftrightarrow$ direct connections or (track) links between nodes
    $\forall \ell \in L$, $\exists$ **capacity** $c_\ell > 0$ [# trains per day]
  - **Line pool**: set of **lines** (origin-destination paths) in G

Line Planning Problem (I)

- **Railway Network Infrastructure**
  governed by a **network operator** (**NOP**) & represented as a digraph $G = (V, L)$
  - $V \longleftrightarrow$ stations or junctions of rail tracks
  - $L \longleftrightarrow$ direct connections or (track) links between nodes
    $\forall \ell \in L$, $\exists$ **capacity** $c_\ell > 0$ [# trains per day]
  - **Line pool**: set of **lines** (origin-destination paths) in G
- **Line Operators (LOPs)** P
  Request usage of lines, at varying **frequencies**, in order to serve their customers

Line Planning Problem (I)

- **Railway Network Infrastructure**
  governed by a **network operator** (**NOP**) & represented as a digraph $G = (V, L)$
  - $V \longleftrightarrow$ stations or junctions of rail tracks
  - $L \longleftrightarrow$ direct connections or (track) links between nodes
    $\forall \ell \in L, \exists$ **capacity** $c_\ell > 0$ [# trains per day]
  - **Line pool**: set of **lines** (origin-destination paths) in G
- **Line Operators (LOPs)** P
  Request usage of lines, at varying **frequencies**, in order to serve their customers
- **Goal**
  Find a **line concept** (feasible allocation of lines to LOPs along with proper frequencies) so as to optimize a system-wise welfare function

Line Planning Problem (II)

- Cost-Oriented Approach: optimize the performance of **NOP**
  - Minimize cost (minimize total / max train travel time)
  - Maximize profit (maximize throughput)
  - ...

Eg, [Claessens-van Dijk-Zwaneveld (1996); Goossens-Hoesel-Kroon (2004)]

Line Planning Problem (II)

- Cost-Oriented Approach: optimize the performance of **NOP**
  - Minimize cost (minimize total / max train travel time)
  - Maximize profit (maximize throughput)
  - ...

Eg, [Claessens-van Dijk-Zwaneveld (1996); Goossens-Hoesel-Kroon (2004)]

- Customer-Oriented Approach: maximize the clients' aggregate level of satisfaction
  - Maximize travelers with direct connections
  - Minimize their total / max number of changes
  - Minimize the traveling time of customers
  - Minimize aggregate payments
  - ...

Eg, [Schöbel-Scholl (2005); Bussieck (1998); Bussieck-Lindner-Lübbecke (2004)]

Robust Line Planning (I)

Robust Line Planning (I)

- Provide line concepts that are robust to fluctuations of the input parameters
  - Disruptions (e.g., delays) to daily operations
  - Temporal unavailability of tracks due to delays/accidents
  - Fluctuating customer demands
  - ...

Robust Line Planning (I)

- Provide line concepts that are robust to fluctuations of the input parameters
  - Disruptions (e.g., delays) to daily operations
  - Temporal unavailability of tracks due to delays/accidents
  - Fluctuating customer demands
  - ...
- Optimization Approach to Robustness (typical representatives):
  - Stochastic programming models: flexible but too large in size; requires apriori knowledge of probability distributions
  - (Classical) robust optimization models: may lead to very conservative solutions

Robust Line Planning (I)

- Provide line concepts that are robust to fluctuations of the input parameters
  - Disruptions (e.g., delays) to daily operations
  - Temporal unavailability of tracks due to delays/accidents
  - Fluctuating customer demands
  - ...
- Optimization Approach to Robustness (typical representatives):
  - Stochastic programming models: flexible but too large in size; requires apriori knowledge of probability distributions
  - (Classical) robust optimization models: may lead to very conservative solutions
  - [Bertsimas-Sim (2004)] : feasibility is guaranteed if # of affected constraints is limited

Robust Line Planning (I)

- Provide line concepts that are robust to fluctuations of the input parameters
  - Disruptions (e.g., delays) to daily operations
  - Temporal unavailability of tracks due to delays/accidents
  - Fluctuating customer demands
  - ...
- Optimization Approach to Robustness (typical representatives):
  - Stochastic programming models: flexible but too large in size; requires apriori knowledge of probability distributions
  - (Classical) robust optimization models: may lead to very conservative solutions
  - [Bertsimas-Sim (2004)] : feasibility is guaranteed if # of affected constraints is limited
  - [Fischetti-Monaci (2009)] : light robustness

Robust Line Planning (I)

- Provide line concepts that are robust to fluctuations of the input parameters
  - Disruptions (e.g., delays) to daily operations
  - Temporal unavailability of tracks due to delays/accidents
  - Fluctuating customer demands
  - ...
- Optimization Approach to Robustness (typical representatives):
  - Stochastic programming models: flexible but too large in size; requires apriori knowledge of probability distributions
  - (Classical) robust optimization models: may lead to very conservative solutions
  - [Bertsimas-Sim (2004)] : feasibility is guaranteed if # of affected constraints is limited
  - [Fischetti-Monaci (2009)] : light robustness
  - [Liebchen-Lübbecke-Möhring-Stiller (2009)] : recoverable robustness

Robust Line Planning (II)

Robust Line Planning (II)

- Game-theoretic Approach to Robustness: participating entities react selfishly to the fluctuations of the input parameters
  - [Schöbel-Schwarze (2006)] : use game dynamics of a non-atomic network congestion game as a robust scheme to deal with delays
  - [Aghassi-Bertsimas (2005)] : robust version (fluctuations in feasibility constraints) of a strategic game is as difficult as the nominal game

Robust Line Planning (III)

Robust Line Planning (III)

- Previous optimization & game-theoretic approaches
  - Powerful set of methods to deal with predictable and/or statically described level of uncertainty in constraints
  - Centralized solution approaches

Robust Line Planning (III)

- Previous optimization & game-theoretic approaches
  - Powerful set of methods to deal with predictable and/or statically described level of uncertainty in constraints
  - Centralized solution approaches

What if uncertainty is neither predictable/quantifiable nor statically describable ?

Robust Line Planning – A Different Perspective

- **Motivation:** regulations for competition – free railway market

Robust Line Planning – A Different Perspective

- **Motivation:** regulations for competition – free railway market
- **LOP:** commercial entity trying to ...
  - ... make profit out of the usage of the infrastructure
  - ... unwilling to reveal its true incentives to the other competitors, or to NOP

Robust Line Planning – A Different Perspective

- **Motivation:** regulations for competition – free railway market
- **LOP:** commercial entity trying to ...
  - ... make profit out of the usage of the infrastructure
  - ... unwilling to reveal its true incentives to the other competitors, or to NOP
- **NOP:** governmental entity, aiming to ...
  - ... maximize the unknown aggregate level of satisfaction for the **LOP**s (**socially optimal** solution)
  - ... ensure fairness in cost sharing

Robust Line Planning – A Different Perspective

- **Motivation:** regulations for competition – free railway market
- **LOP:** commercial entity trying to ...
  - ... make profit out of the usage of the infrastructure
  - ... unwilling to reveal its true incentives to the other competitors, or to NOP
- **NOP:** governmental entity, aiming to ...
  - ... maximize the unknown aggregate level of satisfaction for the **LOP**s (**socially optimal** solution)
  - ... ensure fairness in cost sharing

Our Notion of Robustness

Tolerance to **LOP**s' unknown and/or dynamically changing incentives causing elasticity of frequency requests

Our Approach: A Railway Market (I)

- Each **LOP** $p \in P$ ...
  - ... has a private **utility function** of its assigned frequency $U_p : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}_{\geq 0}$
  - ... has a unique (or multiple) fixed line(s) that interest her (public information)
  - ... competes against the other **LOP**s for the total **frequency** committed to her along her line(s)

Our Approach: A Railway Market (I)

- Each **LOP** $p \in P$ ...
  - ... has a private **utility function** of its assigned frequency $U_p : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}_{\geq 0}$
  - ... has a unique (or multiple) fixed line(s) that interest her (public information)
  - ... competes against the other **LOP**s for the total **frequency** committed to her along her line(s)

- **NOP** uses a **mechanism** ...
  - ... a feasible frequency allocation rule

    and
  - ... an anonymous resource pricing scheme

  aiming to maximize the aggregate level of satisfaction for the **LOP**s

Our Approach: A Railway Market (I)

- Each **LOP** $p \in P$ ...
  - ... has a private **utility function** of its assigned frequency $U_p : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}_{\geq 0}$
  - ... has a unique (or multiple) fixed line(s) that interest her (public information)
  - ... competes against the other **LOP**s for the total **frequency** committed to her along her line(s)

- **NOP** uses a **mechanism** ...
  - ... a feasible frequency allocation rule

    and
  - ... an anonymous resource pricing scheme

  aiming to maximize the aggregate level of satisfaction for the **LOP**s

ASSUMPTION 1 (economy of scale)

For every **LOP** $p \in P$, U_p is strictly increasing and strictly concave

Our Approach: A Railway Market (II)

Reality of an emerging (Pan-European) Railway Market:

- Huge instances to be handled globally by a central authority (**NOP**)

Our Approach: A Railway Market (II)

Reality of an emerging (Pan-European) Railway Market:

- Huge instances to be handled globally by a central authority (**NOP**)
- Real-time changes of
  (i) The network infrastructure
  (ii) **LOP** preferences

Our Approach: A Railway Market (II)

Reality of an emerging (Pan-European) Railway Market:

- Huge instances to be handled globally by a central authority (**NOP**)
- Real-time changes of
  - (i) The network infrastructure
  - (ii) **LOP** preferences
- Instead of using a (static, global) **mechanism** that aims to maximize the aggregate level of satisfaction for the **LOP**s

Our Approach: A Railway Market (II)

Reality of an emerging (Pan-European) Railway Market:

- Huge instances to be handled globally by a central authority (**NOP**)
- Real-time changes of
  - (i) The network infrastructure
  - (ii) **LOP** preferences

Instead of using a (static, centralized) **mechanism** that aims to maximize the aggregate level of satisfaction for the **LOP**s

- Devise a dynamic, decentralized **mechanism** that
  - assures global convergence to the (unknown, possibly changing over time) social optimum
  - is based (as much as possible) on local information

Cases Studied [Bessas, Kontogiannis & Z (2009; 2011)]

[SP] Single Line Pool

- A unique line (path) per **LOP**

[MP] Multiple Line Pools

- A polynomial number of different line pools representing non-overlapping usage of the infrastructure, due to ...
  - ... varying customer traffic (rush-hour morning pool, late morning pool, rush-hour afternoon pool, night pool, etc)
  - ... maintenance
  - ... dependencies between types of lines (a high-speed line affects the choice of lines for other trains)

[MPSU] Multiple line Pools – Single Utility:
One utility function per **LOP**, for the aggregate frequency over all pools

[MPMU] Multiple line Pools – Multiple Utilities:
Different utility functions per pool for each **LOP**

New Contributions [Bessas, Kontogiannis & Z (2009; 2011)]

New Contributions [Bessas, Kontogiannis & Z (2009; 2011)]

- Globally convergent (continuous) decentralized mechanism (dynamic resource pricing and **LOP** bidding scheme) for
  - **[SP]** – adaptation of the **proportionally fair pricing** scheme [Kelly (1997)]
  - **[MPSU]** and **[MPMU]**

[SP]: Single line Pool **[MPSU]:** Multiple line Pools – Single Utility **[MPMU]:** Multiple line Pools – Multiple Utilities

New Contributions [Bessas, Kontogiannis & Z (2009; 2011)]

- Globally convergent (continuous) decentralized mechanism (dynamic resource pricing and **LOP** bidding scheme) for
  - **[SP]** – adaptation of the **proportionally fair pricing** scheme [Kelly (1997)]
  - **[MPSU]** and **[MPMU]**

- Experimental study on discrete variants of the globally convergent mechanisms for **[SP]** and **[MPMU]** on synthetic and real-world data
  - **1st Experiment:** global convergence to social optimum, starting from an arbitrary initial state
    Experiments indicated independence from number of pools, but sensitivity to the shape of the utility functions
  - **2nd Experiment:** convergence to optimality, recovering from small disruptions to a previous social optimum
    Experiments indicated very fast (re-)convergence to optimum

[SP]: Single line Pool **[MPSU]:** Multiple line Pools – Single Utility **[MPMU]:** Multiple line Pools – Multiple Utilities

New Contributions [Bessas, Kontogiannis & Z (2009; 2011)] – In this Talk

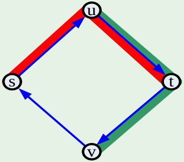
- Globally convergent (continuous) decentralized mechanism (dynamic resource pricing and **LOP** bidding scheme) for
  - **[SP]** – adaptation of the **proportionally fair pricing** scheme [Kelly (1997)]
  - **[MPSU]** and **[MPMU]**

- Experimental study on discrete variants of the globally convergent mechanisms for [SP] and **[MPMU]** on synthetic and real-world data
  - **1st Experiment:** global convergence to social optimum, starting from an arbitrary initial state
    Experiments indicated independence from number of pools, but sensitivity to the shape of the utility functions
  - **2nd Experiment:** convergence to optimality, recovering from small disruptions to a previous social optimum
    Experiments indicated very fast (re-)convergence to optimum

[SP]: Single line Pool **[MPSU]:** Multiple line Pools – Single Utility **[MPMU]:** Multiple line Pools – Multiple Utilities

The Optimization Problem

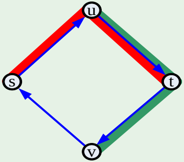
- Line Pool: **routing matrix** $R \in \{0,1\}^{|L|\times|P|}$ (one line per **LOP**)
  - Column $\leftrightarrow$ **LOP** $p \in P$
  - Row $\leftrightarrow$ specific resource (edge) $\ell \in L$



$$R = \begin{array}{c|c|c|} & \mathbf{p} & \mathbf{q} \\ \mathbf{su} & 1 & 0 \\ \mathbf{ut} & 1 & 1 \\ \mathbf{vs} & 0 & 0 \\ \mathbf{tv} & 0 & 1 \end{array}$$

The Optimization Problem

- Line Pool: **routing matrix** $R \in \{0,1\}^{|L|\times|P|}$ (one line per **LOP**)
  - Column $\leftrightarrow$ **LOP** $p \in P$
  - Row $\leftrightarrow$ specific resource (edge) $\ell \in L$



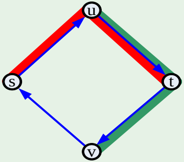
$R =$

	p	**q**
su	1	0
ut	1	1
vs	0	0
tv	0	1

- **Capacity vector** $\mathbf{c} \in (\mathbb{R}_{\geq 0})^{|L|}$: frequency upper bounds of edges

The Optimization Problem

- Line Pool: **routing matrix** $R \in \{0,1\}^{|L| \times |P|}$ (one line per **LOP**)
  - Column $\leftrightarrow$ **LOP** $p \in P$
  - Row $\leftrightarrow$ specific resource (edge) $\ell \in L$

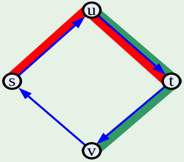


$$R = \begin{array}{c|c|c|} & \mathbf{p} & \mathbf{q} \\ \hline \mathbf{su} & 1 & 0 \\ \hline \mathbf{ut} & 1 & 1 \\ \hline \mathbf{vs} & 0 & 0 \\ \hline \mathbf{tv} & 0 & 1 \\ \hline \end{array}$$

- **Capacity vector** $\mathbf{c} \in (\mathbb{R}_{\geq 0})^{|L|}$: frequency upper bounds of edges
- x_p: path frequency granted to **LOP** p along her line

The Optimization Problem

- Line Pool: **routing matrix** $R \in \{0,1\}^{|L|\times|P|}$ (one line per **LOP**)
  - Column $\leftrightarrow$ **LOP** $p \in P$
  - Row $\leftrightarrow$ specific resource (edge) $\ell \in L$



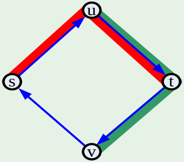
$$R = \begin{array}{c} \\ \mathbf{su} \\ \mathbf{ut} \\ \mathbf{vs} \\ \mathbf{tv} \end{array} \begin{array}{cc} \mathbf{p} & \mathbf{q} \\ \hline 1 & 0 \\ 1 & 1 \\ 0 & 0 \\ 0 & 1 \end{array}$$

- **Capacity vector** $\mathbf{c} \in (\mathbb{R}_{\geq 0})^{|L|}$: frequency upper bounds of edges
- x_p: path frequency granted to **LOP** p along her line
- Goal: find the (unique) optimal solution of the convex program

$$\boxed{\text{SOCIAL}} \qquad \max \left\{ \sum_{p \in P} U_p(x_p) \; : \; R\mathbf{x} \leq \mathbf{c}; \;\; \mathbf{x} \geq \mathbf{0} \right\}$$

The Optimization Problem

- Line Pool: **routing matrix** $R \in \{0,1\}^{|L| \times |P|}$ (one line per **LOP**)
  - Column $\leftrightarrow$ **LOP** $p \in P$
  - Row $\leftrightarrow$ specific resource (edge) $\ell \in L$



$$R = \begin{array}{c|c|c|} & \mathbf{p} & \mathbf{q} \\ \hline \mathbf{su} & 1 & 0 \\ \hline \mathbf{ut} & 1 & 1 \\ \hline \mathbf{vs} & 0 & 0 \\ \hline \mathbf{tv} & 0 & 1 \\ \hline \end{array}$$

- **Capacity vector** $\mathbf{c} \in (\mathbb{R}_{\geq 0})^{|L|}$: frequency upper bounds of edges
- x_p: path frequency granted to **LOP** p along her line
- Goal: find the (unique) optimal solution of the convex program

$$\boxed{\text{SOCIAL}} \quad \max \left\{ \sum_{p \in P} U_p(x_p) \ : \ R\mathbf{x} \leq \mathbf{c}; \ \mathbf{x} \geq \mathbf{0} \right\}$$

Where is the problem?

Difficulties in Solving $\boxed{\text{SOCIAL}}$

$$\boxed{\text{SOCIAL}} \qquad \max\left\{ \sum_{p \in P} U_p(x_p) \,:\, R\mathbf{x} \le \mathbf{c};\; \mathbf{x} \ge \mathbf{0} \right\}$$

Difficulties in Solving $\boxed{\text{SOCIAL}}$

$$\boxed{\text{SOCIAL}} \qquad \max\left\{\sum_{p\in P} U_p(x_p) : R\mathbf{x} \le \mathbf{c};\ \mathbf{x} \ge \mathbf{0}\right\}$$

Reluctance of **LOP**s to reveal their private utilities to either **NOP** or their competitors

$\Rightarrow$ **Ignorance** of the exact shape of the objective function

Difficulties in Solving $\boxed{\text{SOCIAL}}$

$$\boxed{\text{SOCIAL}} \quad \max\left\{\sum_{p\in P} U_p(x_p) : R\mathbf{x} \le \mathbf{c};\ \mathbf{x} \ge \mathbf{0}\right\}$$

Reluctance of **LOP**s to reveal their private utilities to either **NOP** or their competitors

$\Rightarrow$ **Ignorance** of the exact shape of the objective function

Huge scale makes centralized computations inefficient

An Alternative Description of $\boxed{\text{SOCIAL}}$

$$\boxed{\text{SOCIAL}} \quad \max\left\{\sum_{p \in P} U_p(x_p) : R\mathbf{x} \leq \mathbf{c};\ \mathbf{x} \geq \mathbf{0}\right\}$$

- $\hat{\mathbf{x}} \in OPT(SOCIAL) \Rightarrow \exists$ vector of Lagrange Multipliers $\hat{\lambda} = (\hat{\lambda}_\ell)_{\ell \in L}$, satisfying the Karush-Kuhn-Tucker conditions:

An Alternative Description of $\boxed{\text{SOCIAL}}$

$$\boxed{\text{SOCIAL}} \quad \max\left\{\sum_{p\in P} U_p(x_p) : R\mathbf{x} \le \mathbf{c};\ \mathbf{x} \ge \mathbf{0}\right\}$$

- $\hat{\mathbf{x}} \in OPT(SOCIAL) \Rightarrow \exists$ vector of Lagrange Multipliers $\hat{\lambda} = (\hat{\lambda}_\ell)_{\ell\in L}$, satisfying the Karush-Kuhn-Tucker conditions:

$\boxed{\text{KKT-SOCIAL}}$

$$\begin{array}{rcl}
U'_p(\hat{x}_p) & = & \hat{\lambda}^T \cdot R_{\star,p}, \ \ \forall p \in P, \\
\hat{\lambda}_\ell\,(c_\ell - R_{\ell,\star} \cdot \hat{\mathbf{x}}) & = & 0, \ \ \forall \ell \in L, \\
R_{\ell,\star} \cdot \hat{\mathbf{x}} & \le & c_\ell, \ \ \forall \ell \in L, \\
\hat{\lambda}, \hat{\mathbf{x}} & \ge & \mathbf{0}
\end{array}$$

Economic Interpretation of Lagrange Multipliers

Assuming knowledge of the optimal vector of Lagrange multipliers $\hat{\lambda}$

Economic Interpretation of Lagrange Multipliers

Assuming knowledge of the optimal vector of Lagrange multipliers $\hat{\lambda}$

- **NOP** announces pricing scheme:

Each resource $\ell \in L$ charges a per-unit-of-frequency price equal to $\hat{\lambda}_\ell$

Economic Interpretation of Lagrange Multipliers

Assuming knowledge of the optimal vector of Lagrange multipliers $\hat{\lambda}$

- **NOP** announces pricing scheme:

Each resource $\ell \in L$ charges a per-unit-of-frequency price equal to $\hat{\lambda}_\ell$

- Each **LOP** $p \in P$, granted line frequency $x_p \geq 0$, pays **usage cost**:

$$C_p(x_p) = \hat{\mu}_p \cdot x_p$$

where $\hat{\mu}_p \equiv \sum_{\ell \in L : R_{\ell,p}=1} \hat{\lambda}_\ell = \hat{\lambda}^T R_{\star,p}$ is the total per-unit price of p along her line $R_{\star,p}$.

Exploiting the Selfishness of **LOP**s

Each selfish **LOP** is interested in solving:

$$\boxed{\text{USER-I}} \qquad \max\left\{U_p(x_p) - \hat{\mu}_p x_p : x_p \geq 0\right\}$$

Exploiting the Selfishness of **LOP**s

Each selfish **LOP** is interested in solving:

$$\boxed{\text{USER-I}} \qquad \max\left\{U_p(x_p) - \hat{\mu}_p x_p \,:\, x_p \geq 0\right\}$$

ASSUMPTION 2

LOPs control negligible fractions of frequency and are **price takers** (accept announced prices as constant)

Exploiting the Selfishness of **LOP**s

Each selfish **LOP** is interested in solving:

$$\boxed{\text{USER-I}} \qquad \max\left\{U_p(x_p) - \hat{\mu}_p x_p \, : \, x_p \geq 0\right\}$$

ASSUMPTION 2

LOPs control negligible fractions of frequency and are **price takers** (accept announced prices as constant)

The selfish solution $\tilde{x}_p \geq 0$ of $\boxed{\text{USER-I}}$ satisfies

$$U_p'(\tilde{x}_p) = \hat{\mu}_p = \hat{\lambda}^T \cdot R_{\star,p}$$

$\Rightarrow$ the vector of selfish frequencies $\tilde{\mathbf{x}}$ satisfies the first (hard) set of equalities of $\boxed{\text{KKT-SOCIAL}}$

Exploiting the Selfishness of **LOP**s

Each selfish **LOP** is interested in solving:

$$\boxed{\text{USER-I}} \qquad \max \left\{ U_p(x_p) - \hat{\mu}_p x_p \,:\, x_p \geq 0 \right\}$$

ASSUMPTION 2

LOPs control negligible fractions of frequency and are **price takers** (accept announced prices as constant)

- The selfish solution $\tilde{x}_p \geq 0$ of $\boxed{\text{USER-I}}$ satisfies

$$U_p'(\tilde{x}_p) = \hat{\mu}_p = \hat{\lambda}^T \cdot R_{\star,p}$$

$\Rightarrow$ the vector of selfish frequencies $\tilde{\mathbf{x}}$ satisfies the first (hard) set of equalities of $\boxed{\text{KKT-SOCIAL}}$

- The optimal vector $\hat{\lambda}$ of Lagrange multipliers is also not known

Dynamic Pricing Scheme

Iteratively:

1. Each **LOP** $p \in P$ (rather than requesting a frequency x_p) announces a **bid** $w_p \geq 0$ for buying frequency

Dynamic Pricing Scheme

Iteratively:

1. Each **LOP** $p \in P$ (rather than requesting a frequency x_p) announces a **bid** $w_p \geq 0$ for buying frequency

2. **NOP** considers the following program, with strictly concave pseudo–utilities

$$\boxed{\text{NETWORK}} \quad \max \left\{ \sum_{p \in P} \overbrace{U_p(x_p)}^{w_p \cdot \log(x_p)} : R\mathbf{x} \leq \mathbf{c};\ \mathbf{x} \geq \mathbf{0} \right\}$$

whose **optimal** Lagrange Multipliers vector $\bar{\lambda}$ determines the per-unit-prices of the resources

Dynamic Pricing Scheme

Iteratively:

1. Each **LOP** $p \in P$ (rather than requesting a frequency x_p) announces a **bid** $w_p \geq 0$ for buying frequency

2. **NOP** considers the following program, with strictly concave pseudo–utilities

$$\boxed{\text{NETWORK}} \quad \max \left\{ \sum_{p \in P} \overbrace{U_p(x_p)}^{w_p \cdot \log(x_p)} : R\mathbf{x} \leq \mathbf{c};\ \mathbf{x} \geq \mathbf{0} \right\}$$

whose **optimal** Lagrange Multipliers vector $\bar{\lambda}$ determines the per-unit-prices of the resources

3. Allocation of frequencies to **LOP**s: $\forall p \in P,\ \bar{x}_p = \frac{w_p}{\bar{\mu}_p}$
$\bar{\mu}_p \equiv \sum_{\ell \in L : R_{\ell,p}=1} \bar{\lambda}_\ell = \bar{\lambda}^T \cdot R_{\star,p}$ is the total price of p committing a unit of traffic along her line $R_{\star,p}$

An Alternative Description of $\boxed{\text{NETWORK}}$

$$\boxed{\text{NETWORK}} \quad \max\left\{\sum_{p\in P} w_p \cdot \log(x_p) \,:\, R\mathbf{x} \le \mathbf{c};\ \mathbf{x} \ge \mathbf{0}\right\}$$

$\boxed{\text{KKT-NETWORK}}$

$$\begin{aligned}
\frac{w_p}{\bar{x}_p} &= \bar{\lambda}^T \cdot R_{\star,p}, \quad \forall p \in P, \\
\bar{\lambda}_\ell \left(c_\ell - R_{\ell,\star} \cdot \bar{\mathbf{x}}\right) &= 0, \quad \forall \ell \in L, \\
R_{\ell,\star} \cdot \bar{\mathbf{x}} &\le c_\ell, \quad \forall \ell \in L, \\
\bar{\lambda}, \bar{\mathbf{x}} &\ge \mathbf{0}
\end{aligned}$$

What remains?

The only difference between $\boxed{\text{KKT-NETWORK}}$ and $\boxed{\text{KKT-SOCIAL}}$ is the first condition:

$$\boxed{\text{KKT-NETWORK}} \qquad \frac{w_p}{\bar{x}_p} = \bar{\lambda}^T \cdot R_{\star,p}, \quad \forall p \in P$$

vs.

$$\boxed{\text{KKT-SOCIAL}} \qquad U'_p\left(\hat{x}_p\right) = \hat{\lambda}^T \cdot R_{\star,p}, \quad \forall p \in P$$

What remains?

The only difference between $\boxed{\text{KKT-NETWORK}}$ and $\boxed{\text{KKT-SOCIAL}}$ is the first condition:

$$\boxed{\text{KKT-NETWORK}} \qquad \frac{w_p}{\bar{x}_p} = \bar{\lambda}^T \cdot R_{\star,p}, \quad \forall p \in P$$

vs.

$$\boxed{\text{KKT-SOCIAL}} \qquad U'_p\left(\hat{x}_p\right) = \hat{\lambda}^T \cdot R_{\star,p}, \quad \forall p \in P$$

Prove that the optimal solution $(\bar{\mathbf{x}}, \bar{\lambda})$ of $\boxed{\text{KKT-NETWORK}}$ satisfies

$$\forall p \in P, \quad U'_p(\bar{x}_p) = \frac{w_p}{\bar{x}_p}$$

Exploiting (again) the Selfishness of **LOPs**

At each time $t \geq 0$, **LOP** $p \in P$ is interested in solving:

$$\boxed{\text{USER-II}} \qquad \max \left\{ U_p(\underbrace{w_p/\mu_p(t)}_{=x_p(t)}) - w_p \; : \; w_p \geq 0 \right\}$$

Exploiting (again) the Selfishness of **LOPs**

At each time $t \geq 0$, **LOP** $p \in P$ is interested in solving:

$$\boxed{\text{USER-II}} \quad \max \left\{ U_p(\underbrace{w_p/\mu_p(t)}_{=x_p(t)}) - w_p \; : \; w_p \geq 0 \right\}$$

- Given the price taking property, the selfish solution $\tilde{w}_p(t)$ satisfies:

$$(*) \;\; \forall p \in P, \; \frac{1}{\mu_p(t)} \cdot U'_p\left(\frac{\tilde{w}_p(t)}{\mu_p(t)}\right) = 1 \quad \Leftrightarrow \quad U'_p\left(\tilde{x}_p(t)\right) = \frac{\tilde{w}_p(t)}{\tilde{x}_p(t)}$$

Exploiting (again) the Selfishness of **LOP**s

At each time $t \geq 0$, **LOP** $p \in P$ is interested in solving:

$$\boxed{\text{USER-II}} \qquad \max\left\{ U_p(\underbrace{w_p/\mu_p(t)}_{=x_p(t)}) - w_p \,:\, w_p \geq 0 \right\}$$

- Given the price taking property, the selfish solution $\tilde{w}_p(t)$ satisfies:

$$(*) \quad \forall p \in P, \; \frac{1}{\mu_p(t)} \cdot U'_p\left(\frac{\tilde{w}_p(t)}{\mu_p(t)}\right) = 1 \quad \Leftrightarrow \quad U'_p\left(\tilde{x}_p(t)\right) = \frac{\tilde{w}_p(t)}{\tilde{x}_p(t)}$$

At equilibrium we have: $\boxed{\text{KKT-NETWORK}}$ = $\boxed{\text{KKT-SOCIAL}}$!!!

Single line Pool – Recap

- At equilibrium $\boxed{\text{KKT-NETWORK}} = \boxed{\text{KKT-SOCIAL}}$
- Crucial point: set the “right” resource prices and the “right” bids will follow

Single line Pool – Recap

- At equilibrium $\boxed{\text{KKT-NETWORK}} = \boxed{\text{KKT-SOCIAL}}$
- Crucial point: set the “right” resource prices and the “right” bids will follow
- Avoid solving globally $\boxed{\text{NETWORK}}$ (although, in principle we could)

At every time step $t \geq 0$:

1. Every resource $\ell \in L$ updates its per-unit-of-frequency (anonymous) price according to

$$\dot{\lambda}_\ell(t) = \begin{cases} \max\{y_\ell(t) - c_\ell, 0\}, & \text{if } \lambda_\ell(t) = 0\,, \\ (y_\ell(t) - c_\ell), & \text{if } \lambda_\ell(t) > 0\,. \end{cases}$$

where $y_\ell(t) \equiv \sum_{p \in R : R_{\ell,p}=1} x_p(t) = R_{\ell,\star} \cdot \mathbf{x(t)}$ is the cumulative frequency committed at edge $\ell \in L$ at time t

At every time step $t \geq 0$:

1. Every resource $\ell \in L$ updates its per-unit-of-frequency (anonymous) price according to

$$\dot{\lambda}_\ell(t) = \begin{cases} \max\{y_\ell(t) - c_\ell, 0\}, & \text{if } \lambda_\ell(t) = 0\,, \\ (y_\ell(t) - c_\ell), & \text{if } \lambda_\ell(t) > 0\,. \end{cases}$$

   where $y_\ell(t) \equiv \sum_{p \in R : R_{\ell,p}=1} x_p(t) = R_{\ell,\star} \cdot \mathbf{x}(\mathbf{t})$ is the cumulative frequency committed at edge $\ell \in L$ at time t

2. Each **LOP** announces her current bid $w_p(t)$ for buying frequency over her own line, as a solution to $\boxed{\text{USER-II}}$

At every time step $t \geq 0$:

1. Every resource $\ell \in L$ updates its per-unit-of-frequency (anonymous) price according to

$$\dot{\lambda}_\ell(t) = \begin{cases} \max\{y_\ell(t) - c_\ell, 0\}, & \text{if } \lambda_\ell(t) = 0\,, \\ (y_\ell(t) - c_\ell), & \text{if } \lambda_\ell(t) > 0\,. \end{cases}$$

   where $y_\ell(t) \equiv \sum_{p \in R : R_{\ell,p}=1} x_p(t) = R_{\ell,\star} \cdot \mathbf{x}(\mathbf{t})$ is the cumulative frequency committed at edge $\ell \in L$ at time t

2. Each **LOP** announces her current bid $w_p(t)$ for buying frequency over her own line, as a solution to $\boxed{\text{USER-II}}$

3. Each **LOP** $p \in P$ receives a per-unit-of-frequency price $\mu_p(t) \equiv \sum_{\ell \in L : R_{\ell,p}=1} \lambda_\ell(t) = \lambda(\mathbf{t})^T \cdot R_{\star,p}$
   and thus a frequency $x_p(t) = \frac{w_p(t)}{\mu_p(t)}$, at time t

How to Prove Convergence?

Via a Lyapunov Function argument (plus full rank of R) we can prove convergence to the optimal solution $(\bar{\mathbf{x}}, \bar{\lambda}) = (\hat{\mathbf{x}}, \hat{\lambda})$ of both $\boxed{\text{NETWORK}}$ and $\boxed{\text{SOCIAL}}$

Multiple Line Pools

- The **NOP** can ...
  - periodically exploit a set K of line pools
  - determine how to divide the usage of the network among the different pools
- Each line pool operates in disjoint time intervals (time division multiplexing)
- Every **LOP** p ...
  - can claim different lines from different line pools
  - has a different utility function $U_{p,k}$ per line pool k

Multiple Line Pools (set K)

- Pool $k \in K$: routing matrix $R(k) \in \{0,1\}^{|L| \times |P|}$ (one line per **LOP** per pool)

Multiple Line Pools (set K)

- Pool $k \in K$: routing matrix $R(k) \in \{0, 1\}^{|L| \times |P|}$ (one line per **LOP** per pool)
- Capacity vector $\mathbf{c} \in (\mathbb{R}_{\geq 0})^{|L|}$: max frequency over whole time period

Multiple Line Pools (set K)

- Pool $k \in K$: routing matrix $R(k) \in \{0,1\}^{|L| \times |P|}$ (one line per **LOP** per pool)
- Capacity vector $\mathbf{c} \in (\mathbb{R}_{\geq 0})^{|L|}$: max frequency over whole time period
- $x_{p,k}$: frequency granted to **LOP** p along her line within pool k

Multiple Line Pools (set K)

- Pool $k \in K$: routing matrix $R(k) \in \{0,1\}^{|L| \times |P|}$ (one line per **LOP** per pool)
- Capacity vector $\mathbf{c} \in (\mathbb{R}_{\geq 0})^{|L|}$: max frequency over whole time period
- $x_{p,k}$: frequency granted to **LOP** p along her line within pool k
- $f_k, k \in K$: proportion consumed (from the capacity of each edge) by pool k over the whole time period (determined by **NOP**)

Multiple Line Pools (set K)

- Pool $k \in K$: routing matrix $R(k) \in \{0,1\}^{|L|\times|P|}$ (one line per **LOP** per pool)
- Capacity vector $\mathbf{c} \in (\mathbb{R}_{\geq 0})^{|L|}$: max frequency over whole time period
- $x_{p,k}$: frequency granted to **LOP** p along her line within pool k
- $f_k, k \in K$: proportion consumed (from the capacity of each edge) by pool k over the whole time period (determined by **NOP**)
- Find the (unique) optimal solution of the convex program:

MULTI-SOCIAL-2 (MSC2)

$$
\begin{aligned}
\max \quad & \sum_{p\in P} U_p(\mathbf{x}_p) = \sum_{p\in P}\sum_{k\in K} U_{p,k}(x_{p,k}) \\
\text{s.t. } \forall(\ell,k) \in L\times K, \quad & \sum_{p\in P} R_{\ell,p}(k)\cdot x_{p,k} \leq c_{\ell,k}\cdot f_k \\
& \sum_{k\in K} f_k \leq 1;\ \mathbf{x},\mathbf{f} \geq \mathbf{0}
\end{aligned}
$$

An Alternative Description of $\boxed{\text{MSC2}}$

- $(\hat{\mathbf{x}}, \hat{\mathbf{f}}) \in OPT(MSC2) \Rightarrow \exists$ vector of Lagrange Multipliers $\left(\hat{\boldsymbol{\Lambda}} = (\hat{\Lambda}_{\ell,k})_{\ell \in L, k \in K}, \hat{\zeta}\right)$, satisfying the Karush-Kuhn-Tucker conditions:

An Alternative Description of $\boxed{MSC2}$

- $(\hat{\mathbf{x}}, \hat{\mathbf{f}}) \in OPT(MSC2) \Rightarrow \exists$ vector of Lagrange Multipliers $\left(\hat{\boldsymbol{\Lambda}} = (\hat{\Lambda}_{\ell,k})_{\ell \in L, k \in K}, \hat{\zeta}\right)$, satisfying the Karush-Kuhn-Tucker conditions:

KKT-MSC2 — $\Lambda_{\ell,k}$: per-unit-of-frequency price

$$
\begin{aligned}
U'_{p,k}(\hat{x}_{p,k}) = \sum_{\ell \in L} \hat{\Lambda}_{\ell,k} \cdot R_{\ell,p}(k) &\equiv \mu_{p,k}(\hat{\boldsymbol{\Lambda}}),\ (p,k) \in P \times K \\
\sum_{\ell \in L} \hat{\Lambda}_{\ell,k} \cdot c_\ell &= \hat{\zeta},\ k \in K \\
\hat{\Lambda}_{\ell,k} \left[\sum_{p \in P} R_{\ell,p}(k) \cdot \hat{x}_{p,k} - c_\ell \hat{f}_k\right] &= 0,\ (\ell,k) \in L \times K \\
\hat{\zeta} \cdot \left(\sum_{k \in K} \hat{f}_k - 1\right) &= 0 \\
\sum_{p \in P} R(k)_{\ell,p} \cdot \hat{x}_{p,k} &\leq c_\ell \cdot \hat{f}_k,\ (\ell,k) \in L \times K \\
\sum_{k \in K} \hat{f}_k &\leq 1 \\
\hat{\mathbf{x}}, \hat{\mathbf{f}}, \hat{\boldsymbol{\Lambda}}, \hat{\zeta} &\geq \mathbf{0}
\end{aligned}
$$

An Alternative Description of $\boxed{\text{MSC2}}$

- $(\hat{\mathbf{x}}, \hat{\mathbf{f}}) \in OPT(MSC2) \Rightarrow \exists$ vector of Lagrange Multipliers $\left(\hat{\mathbf{\Lambda}} = (\hat{\Lambda}_{\ell,k})_{\ell \in L, k \in K}, \hat{\zeta}\right)$, satisfying the Karush-Kuhn-Tucker conditions:

KKT-MSC2

$\Lambda_{\ell,k}$: per-unit-of-frequency price

Per-unit cost of **LOP** p in pool k

$$
\begin{array}{rcl}
U'_{p,k}(\hat{x}_{p,k}) = \sum_{\ell \in L} \hat{\Lambda}_{\ell,k} \cdot R_{\ell,p}(k) & \equiv & \mu_{p,k}(\hat{\mathbf{\Lambda}}),\ (p,k) \in P \times K \\
\sum_{\ell \in L} \hat{\Lambda}_{\ell,k} \cdot c_\ell & = & \hat{\zeta},\ k \in K \\
\hat{\Lambda}_{\ell,k} \left[\sum_{p \in P} R_{\ell,p}(k) \cdot \hat{x}_{p,k} - c_\ell \hat{f}_k \right] & = & 0,\ (\ell,k) \in L \times K \\
\hat{\zeta} \cdot \left(\sum_{k \in K} \hat{f}_k - 1 \right) & = & 0 \\
\sum_{p \in P} R(k)_{\ell,p} \cdot \hat{x}_{p,k} & \leq & c_\ell \cdot \hat{f}_k,\ (\ell,k) \in L \times K \\
\sum_{k \in K} \hat{f}_k & \leq & 1 \\
\hat{\mathbf{x}}, \hat{\mathbf{f}}, \hat{\mathbf{\Lambda}}, \hat{\zeta} & \geq & \mathbf{0}
\end{array}
$$

An Alternative Description of $\boxed{\text{MSC2}}$

- $(\hat{\mathbf{x}}, \hat{\mathbf{f}}) \in OPT(MSC2) \Rightarrow \exists$ vector of Lagrange Multipliers $\left(\hat{\boldsymbol{\Lambda}} = (\hat{\Lambda}_{\ell,k})_{\ell\in L, k\in K}, \hat{\zeta}\right)$, satisfying the Karush-Kuhn-Tucker conditions:

KKT-MSC2

$\Lambda_{\ell,k}$: per-unit-of-frequency price

All pools have same aggregate cost

$$
\begin{aligned}
U'_{p,k}(\hat{x}_{p,k}) = \sum_{\ell\in L} \hat{\Lambda}_{\ell,k} \cdot R_{\ell,p}(k) \quad &\equiv \quad \mu_{p,k}(\hat{\boldsymbol{\Lambda}}),\ (p,k) \in P \times K \\
\sum_{\ell\in L} \hat{\Lambda}_{\ell,k} \cdot c_\ell \quad &= \quad \hat{\zeta},\ k \in K \\
\hat{\Lambda}_{\ell,k} \left[\sum_{p\in P} R_{\ell,p}(k) \cdot \hat{x}_{p,k} - c_\ell \hat{f}_k\right] \quad &= \quad 0,\ (\ell,k) \in L \times K \\
\hat{\zeta} \cdot \left(\sum_{k\in K} \hat{f}_k - 1\right) \quad &= \quad 0 \\
\sum_{p\in P} R(k)_{\ell,p} \cdot \hat{x}_{p,k} \quad &\leq \quad c_\ell \cdot \hat{f}_k,\ (\ell,k) \in L \times K \\
\sum_{k\in K} \hat{f}_k \quad &\leq \quad 1 \\
\hat{\mathbf{x}}, \hat{\mathbf{f}}, \hat{\boldsymbol{\Lambda}}, \hat{\zeta} \quad &\geq \quad \mathbf{0}
\end{aligned}
$$

An Alternative Description of $\boxed{\text{MSC2}}$

- $(\hat{\mathbf{x}}, \hat{\mathbf{f}}) \in OPT(MSC2) \Rightarrow \exists$ vector of Lagrange Multipliers $\left(\hat{\boldsymbol{\Lambda}} = (\hat{\Lambda}_{\ell,k})_{\ell \in L, k \in K}, \hat{\zeta}\right)$, satisfying the Karush-Kuhn-Tucker conditions:

KKT-MSC2

$\Lambda_{\ell,k}$: per-unit-of-frequency price

Network is totally distributed among pools

$$
\begin{aligned}
U'_{p,k}(\hat{x}_{p,k}) = \textstyle\sum_{\ell \in L} \hat{\Lambda}_{\ell,k} \cdot R_{\ell,p}(k) \quad &\equiv \quad \mu_{p,k}(\hat{\boldsymbol{\Lambda}}),\ (p,k) \in P \times K \\
\textstyle\sum_{\ell \in L} \hat{\Lambda}_{\ell,k} \cdot c_\ell \quad &= \quad \hat{\zeta},\ k \in K \\
\hat{\Lambda}_{\ell,k} \left[\textstyle\sum_{p \in P} R_{\ell,p}(k) \cdot \hat{x}_{p,k} - c_\ell \hat{f}_k \right] \quad &= \quad 0,\ (\ell,k) \in L \times K \\
\hat{\zeta} \cdot \left(\textstyle\sum_{k \in K} \hat{f}_k - 1 \right) \quad &= \quad 0 \\
\textstyle\sum_{p \in P} R(k)_{\ell,p} \cdot \hat{x}_{p,k} \quad &\leq \quad c_\ell \cdot \hat{f}_k,\ (\ell,k) \in L \times K \\
\textstyle\sum_{k \in K} \hat{f}_k \quad &\leq \quad 1 \\
\hat{\mathbf{x}}, \hat{\mathbf{f}}, \hat{\boldsymbol{\Lambda}}, \hat{\zeta} \quad &\geq \quad \mathbf{0}
\end{aligned}
$$

Pricing Scheme

1. Each **LOP** $p \in P$ announces a **bid** $w_{p,k} \geq 0$ for buying frequency in pool $k \in K$

Pricing Scheme

1. Each **LOP** $p \in P$ announces a **bid** $w_{p,k} \geq 0$ for buying frequency in pool $k \in K$
2. **NOP** considers the following program, with strictly concave pseudo–utilities, whose **optimal** Lagrange Multipliers vector $\bar{\mathbf{\Lambda}}$ determines the per-unit-prices of the resources in the pools

MNET2

$$\max. \quad \sum_{p \in P} \sum_{k \in K} \overbrace{U_{p,k}(x_{p,k})}^{w_{p,k} \cdot \log(x_{p,k})}$$

$$\text{s.t.} \quad \forall (\ell, k) \in L \times K, \ \sum_{p \in P} R(k)_{\ell,p} \cdot x_{p,k} \leq c_{\ell,k} \cdot f_k ; \sum_{k \in K} f_k \leq 1; \ \mathbf{f}, \mathbf{x} \geq \mathbf{0}$$

Pricing Scheme

1. Each **LOP** $p \in P$ announces a **bid** $w_{p,k} \geq 0$ for buying frequency in pool $k \in K$

2. **NOP** considers the following program, with strictly concave pseudo–utilities, whose **optimal** Lagrange Multipliers vector $\bar{\mathbf{\Lambda}}$ determines the per-unit-prices of the resources in the pools

MNET2

$$\max. \quad \sum_{p \in P} \sum_{k \in K} \overbrace{U_{p,k}(x_{p,k})}^{w_{p,k} \cdot \log(x_{p,k})}$$

$$\text{s.t.} \quad \forall (\ell, k) \in L \times K, \ \sum_{p \in P} R(k)_{\ell,p} \cdot x_{p,k} \leq c_{\ell,k} \cdot f_k ; \sum_{k \in K} f_k \leq 1; \ \mathbf{f}, \mathbf{x} \geq \mathbf{0}$$

3. Allocation of frequencies to **LOPs**: $\forall p \in P, \forall k \in K, \ \bar{x}_{p,k} = \frac{w_{p,k}}{\bar{\mu}_{p,k}}$
$\bar{\mu}_{p,k} \equiv \sum_{\ell \in L} \bar{\Lambda}_{\ell,k} \cdot R_{\ell,p}(k)$ is the total price of p for committing a unit of traffic along her line in pool $k \in K$

An Alternative Description of $\boxed{\text{MNET2}}$

- $(\bar{\mathbf{x}}, \bar{\mathbf{f}}) \in OPT(MNET2) \Rightarrow \exists$ vector of Lagrange Multipliers $\left(\bar{\boldsymbol{\Lambda}} = (\bar{\Lambda}_{\ell,k})_{\ell \in L, k \in K}, \bar{\zeta}\right)$, satisfying the Karush-Kuhn-Tucker conditions:

An Alternative Description of $\boxed{\text{MNET2}}$

- $(\bar{\mathbf{x}}, \bar{\mathbf{f}}) \in OPT(MNET2) \Rightarrow \exists$ vector of Lagrange Multipliers $\left(\bar{\boldsymbol{\Lambda}} = (\bar{\Lambda}_{\ell,k})_{\ell\in L, k\in K}, \bar{\zeta}\right)$, satisfying the Karush-Kuhn-Tucker conditions:

KKT-MNET2

$$
\begin{aligned}
\overbrace{U'_{p,k}(\bar{x}_{p,k})}^{\frac{w_{p,k}}{\bar{x}_{p,k}}} = \textstyle\sum_{\ell\in L} \bar{\Lambda}_{\ell,k} \cdot R_{\ell,p}(k) \quad &\equiv \quad \bar{\mu}_{p,k}, \ (p,k) \in P \times K \\
\textstyle\sum_{\ell\in L} \bar{\Lambda}_{\ell,k} \cdot c_\ell \quad &= \quad \bar{\zeta}, \ k \in K \\
\bar{\Lambda}_{\ell,k} \left[\textstyle\sum_{p\in P} R_{\ell,p}(k) \cdot \bar{x}_{p,k} - c_\ell \bar{f}_k\right] \quad &= \quad 0, \ (\ell, k) \in L \times K \\
\bar{\zeta} \cdot \left(\textstyle\sum_{k\in K} \bar{f}_k - 1\right) \quad &= \quad 0 \\
\textstyle\sum_{p\in P} R(k)_{\ell,p} \cdot \bar{x}_{p,k} \quad &\leq \quad c_\ell \cdot \bar{f}_k, \ (\ell, k) \in L \times K \\
\textstyle\sum_{k\in K} \bar{f}_k \quad &\leq \quad 1 \\
\bar{\mathbf{x}}, \bar{\mathbf{f}}, \bar{\boldsymbol{\Lambda}}, \bar{\zeta} \quad &\geq \quad \mathbf{0}
\end{aligned}
$$

An Alternative Description of $\boxed{\text{MNET2}}$

- $(\bar{\mathbf{x}}, \bar{\mathbf{f}}) \in OPT(MNET2) \Rightarrow \exists$ vector of Lagrange Multipliers $\left(\bar{\boldsymbol{\Lambda}} = (\bar{\Lambda}_{\ell,k})_{\ell \in L, k \in K}, \bar{\zeta}\right)$, satisfying the Karush-Kuhn-Tucker conditions:

KKT-MNET2 — The only difference with KKT-MSC2

$$
\begin{array}{rcl}
\overbrace{U'_{p,k}(\bar{x}_{p,k})}^{\frac{w_{p,k}}{\bar{x}_{p,k}}} = \sum_{\ell \in L} \bar{\Lambda}_{\ell,k} \cdot R_{\ell,p}(k) & \equiv & \bar{\mu}_{p,k}, \ (p,k) \in P \times K \\
\sum_{\ell \in L} \bar{\Lambda}_{\ell,k} \cdot c_\ell & = & \bar{\zeta}, \ k \in K \\
\bar{\Lambda}_{\ell,k} \left[\sum_{p \in P} R_{\ell,p}(k) \cdot \bar{x}_{p,k} - c_\ell \bar{f}_k \right] & = & 0, \ (\ell, k) \in L \times K \\
\bar{\zeta} \cdot \left(\sum_{k \in K} \bar{f}_k - 1 \right) & = & 0 \\
\sum_{p \in P} R(k)_{\ell,p} \cdot \bar{x}_{p,k} & \leq & c_\ell \cdot \bar{f}_k, \ (\ell,k) \in L \times K \\
\sum_{k \in K} \bar{f}_k & \leq & 1 \\
\bar{\mathbf{x}}, \bar{\mathbf{f}}, \bar{\boldsymbol{\Lambda}}, \bar{\zeta} & \geq & \mathbf{0}
\end{array}
$$

Multiple Line Pools

- Selfishness of **LOP**s $\Rightarrow$ at equilibrium $\boxed{\text{KKT-MS2}} = \boxed{\text{KKT-MNET2}}$

Multiple Line Pools

- Selfishness of **LOPs** $\Rightarrow$ at equilibrium $\boxed{\text{KKT-MS2}} = \boxed{\text{KKT-MNET2}}$

KEY PROPERTIES

1. The NOP completely divides the infrastructure among the pools
2. For any fixed $\mathbf{f}$ (that completely divides the infrastructure among the pools) the optimal value of $\boxed{\text{KKT-MSC2}}$ depends exclusively on the optimal $\bar{\boldsymbol{\Lambda}}$

Multiple Line Pools

- Selfishness of **LOPs** $\Rightarrow$ at equilibrium $\boxed{\text{KKT-MS2}} = \boxed{\text{KKT-MNET2}}$

KEY PROPERTIES

1. The NOP completely divides the infrastructure among the pools
2. For any fixed $\mathbf{f}$ (that completely divides the infrastructure among the pools) the optimal value of $\boxed{\text{KKT-MSC2}}$ depends exclusively on the optimal $\bar{\boldsymbol{\Lambda}}$

- KEY PROPERTIES $\Rightarrow$ dynamic (decentralized) scheme for solving $\boxed{\text{KKT-MNET2}}$

Dynamic Scheme for solving MNET2

At every time step $t \geq 0$:

1. Resource price updates (by the resources, per pool, continuously):

$$\forall (\ell, k) \in L \times K,\ \dot{\Lambda}_{\ell,k}(t) = \begin{cases} \max\left\{y_{\ell,k}(t) - c_\ell f_k, 0\right\}, & \text{if } \Lambda_{\ell,k}(t) = 0 \\ \left[y_{\ell,k}(t) - c_\ell f_k\right], & \text{if } \Lambda_{\ell,k}(t) > 0 \end{cases}$$

Dynamic Scheme for solving MNET2

At every time step $t \geq 0$:

1. Resource price updates (by the resources, per pool, continuously):

$$\forall(\ell, k) \in L \times K,\ \dot{\Lambda}_{\ell,k}(t) = \begin{cases} \max\{y_{\ell,k}(t) - c_\ell f_k, 0\}, & \text{if } \Lambda_{\ell,k}(t) = 0 \\ [y_{\ell,k}(t) - c_\ell f_k], & \text{if } \Lambda_{\ell,k}(t) > 0 \end{cases}$$

2. **LOP** bid updates (only when resource prices have stabilized):

$$\forall p \in P,\ w_p(t) \in \arg\max_{\mathbf{w}_p \geq \mathbf{0}} \left\{ \sum_{k \in K} \left(U_{p,k}\left(\frac{w_{p,k}}{\bar{\mu}_{p,k}}\right) - w_{p,k} \right) \right\}$$

Dynamic Scheme for solving MNET2

At every time step $t \geq 0$:

1. Resource price updates (by the resources, per pool, continuously):

$$\forall(\ell, k) \in L \times K,\ \dot{\Lambda}_{\ell,k}(t) = \begin{cases} \max\{y_{\ell,k}(t) - c_\ell f_k, 0\}, & \text{if } \Lambda_{\ell,k}(t) = 0 \\ [y_{\ell,k}(t) - c_\ell f_k], & \text{if } \Lambda_{\ell,k}(t) > 0 \end{cases}$$

2. **LOP** bid updates (only when resource prices have stabilized):

$$\forall p \in P,\ w_p(t) \in \arg\max_{\mathbf{w}_p \geq \mathbf{0}} \left\{ \sum_{k \in K} \left(U_{p,k}\left(\frac{w_{p,k}}{\bar{\mu}_{p,k}}\right) - w_{p,k} \right) \right\}$$

3. Allocation of path frequencies: $\forall p \in P, \mathbf{x}_p(t) = \left(\frac{\bar{w}_{p,k}(t)}{\bar{\mu}_{p,k}(t)}\right)_{k \in K}$

Dynamic Scheme for solving MNET2

At every time step $t \geq 0$:

1. Resource price updates (by the resources, per pool, continuously):

$$\forall (\ell, k) \in L \times K,\ \dot{\Lambda}_{\ell,k}(t) = \begin{cases} \max\left\{y_{\ell,k}(t) - c_\ell f_k, 0\right\}, & \text{if } \Lambda_{\ell,k}(t) = 0 \\ \left[y_{\ell,k}(t) - c_\ell f_k\right], & \text{if } \Lambda_{\ell,k}(t) > 0 \end{cases}$$

2. **LOP** bid updates (only when resource prices have stabilized):

$$\forall p \in P,\ w_p(t) \in \arg\max_{\mathbf{w}_p \geq \mathbf{0}} \left\{ \sum_{k \in K} \left(U_{p,k}\left(\frac{w_{p,k}}{\bar{\mu}_{p,k}}\right) - w_{p,k} \right) \right\}$$

3. Allocation of path frequencies: $\forall p \in P, \mathbf{x}_p(t) = \left(\frac{\bar{w}_{p,k}(t)}{\bar{\mu}_{p,k}(t)}\right)_{k \in K}$

4. Capacity Proportion updates (by the **NOP**, only when resource prices and **LOP bids** have stabilized):

$$\zeta(t) = \frac{1}{|K|} \sum_{k \in K} \mathbf{c}^T \cdot \Lambda_{\star,k}(t)$$

$$\forall k \in K,\ \dot{f}_k(t) = \phi(t) \cdot \max\left\{0, \mathbf{c}^T \cdot \Lambda_{\star,k}(t) - \zeta(t)\right\}$$

Experimental Study – Synthetic Data

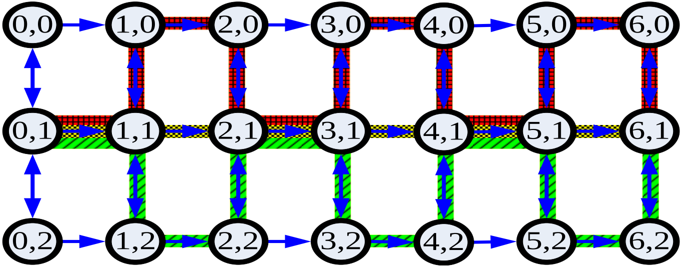
- grid graphs $n \times p$, $n \in \{3, 7\}$, $p \in [120, 3600]$

Experimental Study – Synthetic Data

- grid graphs $n \times p$, $n \in \{3, 7\}$, $p \in [120, 3600]$
- $c_\ell \in [10, 110)$ randomly chosen

Experimental Study – Synthetic Data

- grid graphs $n \times p$, $n \in \{3, 7\}$, $p \in [120, 3600]$
- $c_\ell \in [10, 110)$ randomly chosen
- $|K| \in [2, 4]$; 3 types of LOPs



Lines (paths): deterministic & random

Experimental Study – Real Data

- Two parts of the German railway network; $c_\ell \in [8, 16]$
  - R1: 280 nodes, 354 edges, $|\text{total lines}| \in [100, 400]$
  - R2: 296 nodes, 393 edges, $|\text{total lines}| \in [100, 1000]$

Experimental Study – Real Data

- Two parts of the German railway network; $c_\ell \in [8, 16]$
  - R1: 280 nodes, 354 edges, $|\text{total lines}| \in [100, 400]$
  - R2: 296 nodes, 393 edges, $|\text{total lines}| \in [100, 1000]$
- Per instance
  - $|K| = 2$
  - about 10% difference in lines between the pools

1st Experiment: Convergence to OPT for [MPMU]

- Scenarios considered
  - S1: $U_{p,1}(x_{p,1}) = 10^4 \sqrt{x_{p,1}}$ and $U_{p,2}(x_{p,2}) = 10^4 \sqrt{x_{p,2}}, \ \forall p \in P$.
  - S2: $U_{p,1}(x_{p,1}) = \frac{3}{4} \cdot 10^4 \cdot \sqrt{x_{p,1}}$ and $U_{p,2}(x_{p,2}) = \frac{4}{5} \cdot 10^4 \cdot \sqrt{x_{p,2}}, \ \forall p \in P$.
  - S3: $U_{p,1}(x_{p,1}) = 10^4 \cdot \sqrt{x_{p,1}}$ and $U_{p,2}(x_{p,2}) = \frac{1}{2} \cdot 10^4 \cdot \sqrt{x_{p,2}}, \ \forall p \in P$.
  - S4: $U_{p,1}(x_{p,1}) = 10^4 \cdot \sqrt{x_{p,1}}$ and $U_{p,2}(x_{p,2}) = \frac{1}{4} \cdot 10^4 \cdot \sqrt{x_{p,2}}, \ \forall p \in P$.

- Measured quantity: number of updates in the vector **f** of capacity proportions (= # [SP] instances need to be solved)

Results on **[MPMU]** Convergence

updates of **f** in $R1$ with two line pools, for all four scenarios

#Lines	S1	S2	S3	S4
100	9	33	127	178
200	12	33	127	178
300	19	29	128	178

Similar results for $R2$

Results on **[MPMU]** Convergence

\# updates of **f** in *R*1 with two line pools, for all four scenarios

#Lines	S1	S2	S3	S4
100	9	33	127	178
200	12	33	127	178
300	19	29	128	178

Similar results for *R*2

Bottom Line for **[MPMU]** Convergence

\# updates for convergence to OPT largely depends on the exact parameters of the utility functions, and not really on the number of pools

2nd Experiment: Disruptions in **[MPMU]**

- The system is currently at optimality
- How fast can it re-converge to optimality after a disruption ?

2nd Experiment: Disruptions in **[MPMU]**

- The system is currently at optimality
- How fast can it re-converge to optimality after a disruption ?
- **Disruption:** Change (track breakdown, or improvement) in the capacities of some edges
- **Disruption Scenarios:**
  - D1: Reducing the capacity of a certain number of edges (chosen among the congested ones)
  - D2: Increasing the capacity of a certain number of edges (chosen among the congested ones)
  - D3: Reducing the capacity of a certain number of edges, while increasing the capacity of an equal number of a different set of edges (chosen among the congested ones)
- Change in capacity of a disrupted edge: $\pm 10\%$ or $\pm 50\%$

Disruptions in the **[MPMU]** Case (I)

- Two pools considered (random for grid-networks, with 10% difference from each other in $R1$)
- Measured quantity: number of updates in the **LOP**s' bid vectors
- Starting from previous OPT, no update in vector $\mathbf{f}$ of capacity proportions occurred

Disruptions in the **[MPMU]** Case (II)

updates of **w** to recover optimality in $7 \times p$ grid-networks, starting from a previous optimal state

Disruptions	p	D1	D2	D3
10%	120	0	0	0
	180	0	0	0
	240	0	0	0
	300	0	0	0
	360	0	0	0
50%	120	0	2	1
	180	0	2	0
	240	0	0	0
	300	0	1	2
	360	0	2	2

updates of **w** to recover optimality in $R1$, starting from a previous optimal state

Disruption	#Lines	D1	D2	D3
10%	100	0	0	0
	200	0	0	0
	300	0	0	0
50%	100	0	0	0
	200	0	0	0
	300	0	0	0
90%	100	0	3	0
	200	0	2	2
	300	0	0	0

Disruptions in the **[MPMU]** Case (II)

Disruptions	p	D1	D2	D3
10%	120	0	0	0
	180	0	0	0
	240	0	0	0
	300	0	0	0
	360	0	0	0
50%	120	0	2	1
	180	0	2	0
	240	0	0	0
	300	0	1	2
	360	0	2	2

updates of **w** to recover optimality in $7 \times p$ grid-networks, starting from a previous optimal state

Disruption	#Lines	D1	D2	D3
10%	100	0	0	0
	200	0	0	0
	300	0	0	0
50%	100	0	0	0
	200	0	0	0
	300	0	0	0
90%	100	0	3	0
	200	0	2	2
	300	0	0	0

updates of **w** to recover optimality in $R1$, starting from a previous optimal state

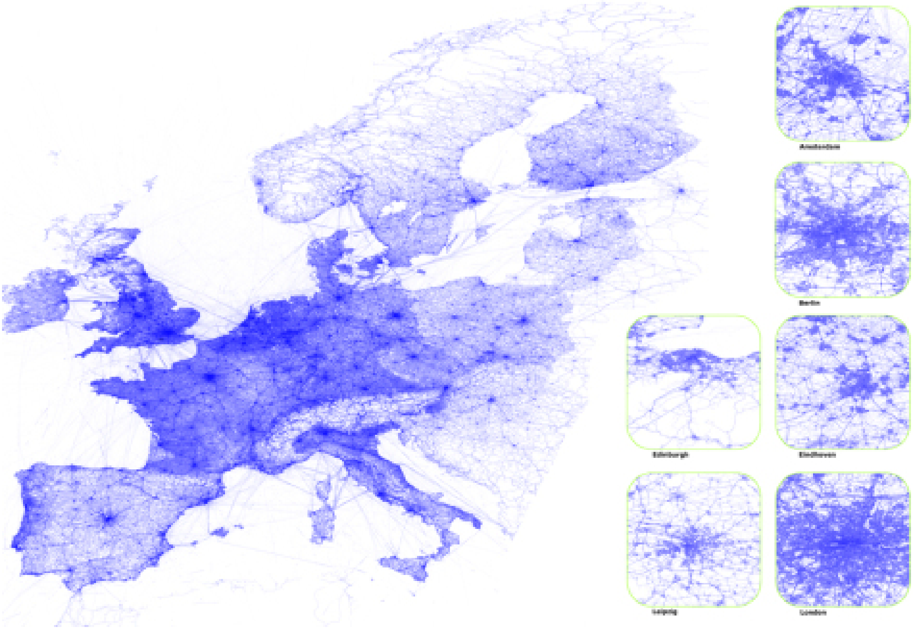
Bottom Line for disruptions in [MPMU]

Very rarely there is a need (for only a few) bid updates, after disruptions

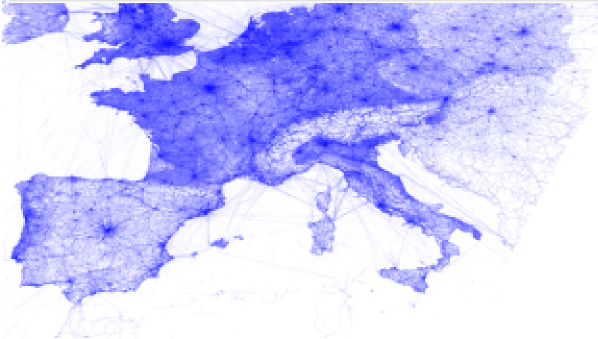
Conclusion

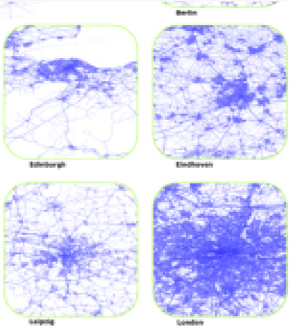
- Incentive-compatible robust solutions for line planning (**[SP]**,**[MPMU]**)
  - Robustness against unknown incentives
  - Recoverability to (unknown) social optimum via dynamic, decentralized mechanism
- Experiments indicated
  - **Convergence** (starting from arbitrary initial state): **independent of # pools**, but sensitive to utility functions
  - **Very fast re-convergence to optimum** in case of disruptions (starting from an optimal state)

Outline



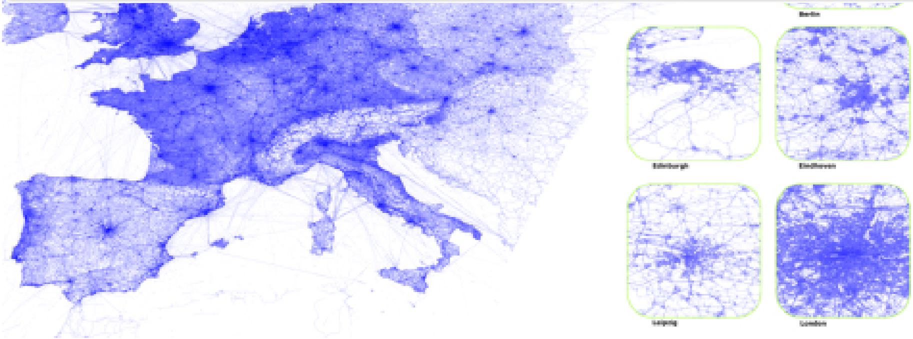
Raw traffic (speed probe) data





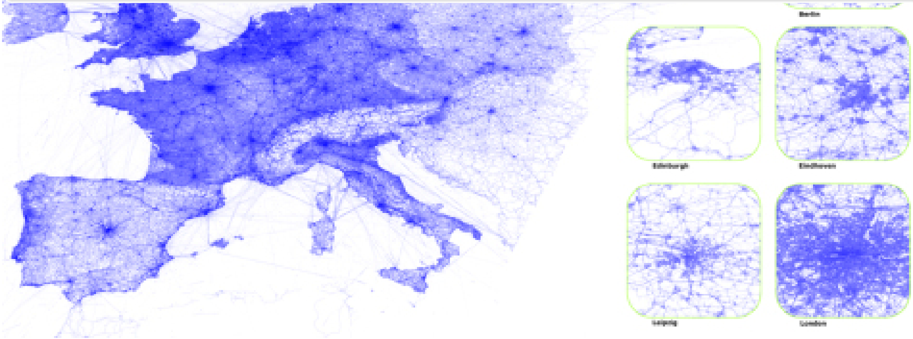
Raw traffic (speed probe) data

- **70 Million** contributing users



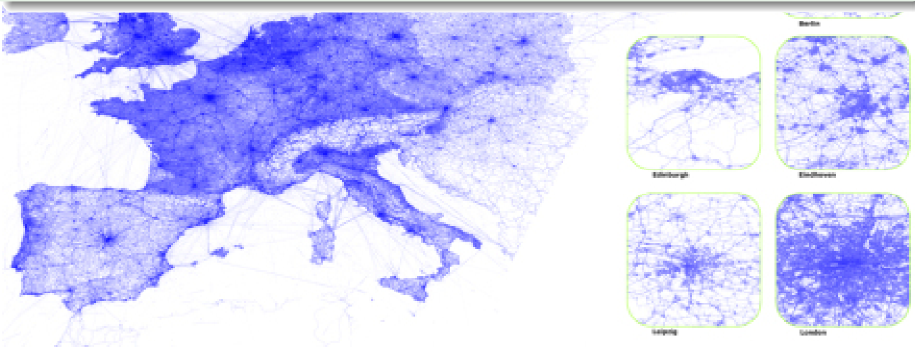
Raw traffic (speed probe) data

- **70 Million** contributing users
- **4 Billion** measurements per day



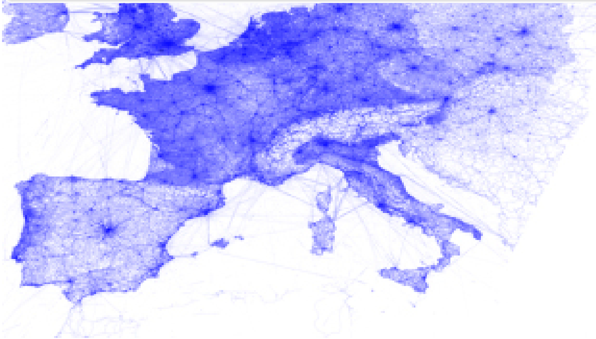
Raw traffic (speed probe) data

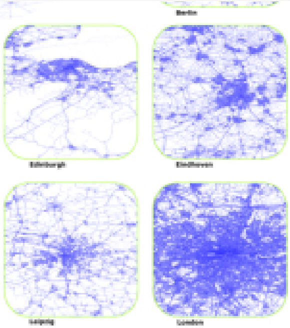
- **70 Million** contributing users
- **4 Billion** measurements per day
- **5 Trillion** measurements over 140 Billion Km



Raw traffic (speed probe) data

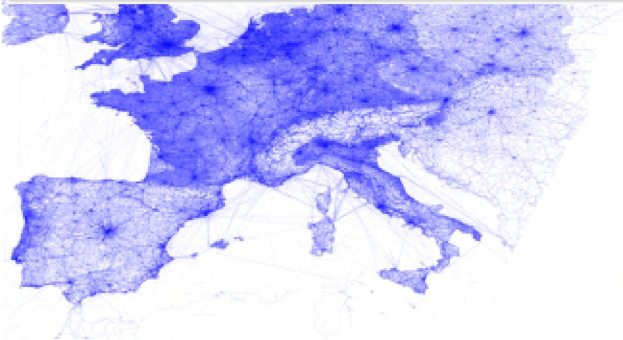
- **70 Million** contributing users
- **4 Billion** measurements per day
- **5 Trillion** measurements over 140 Billion Km
- every road segment measured 2000 times on average

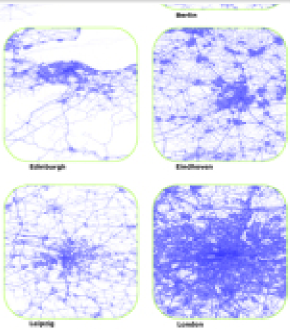




Raw traffic (speed probe) data

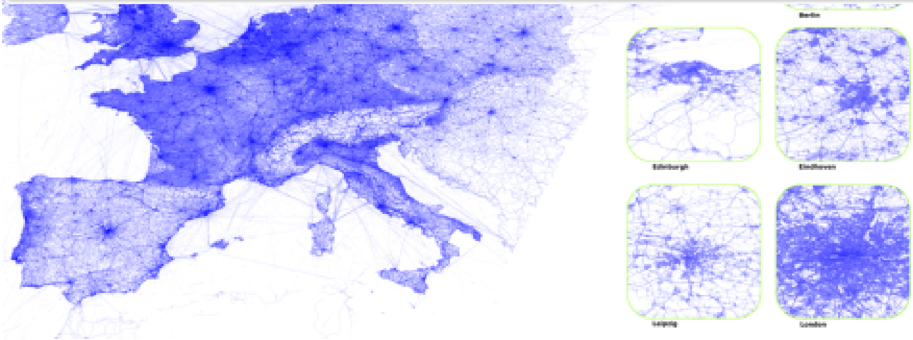
- **70 Million** contributing users
- **4 Billion** measurements per day
- **5 Trillion** measurements over 140 Billion Km
- every road segment measured 2000 times on average
- measured speeds in 5-min intervals





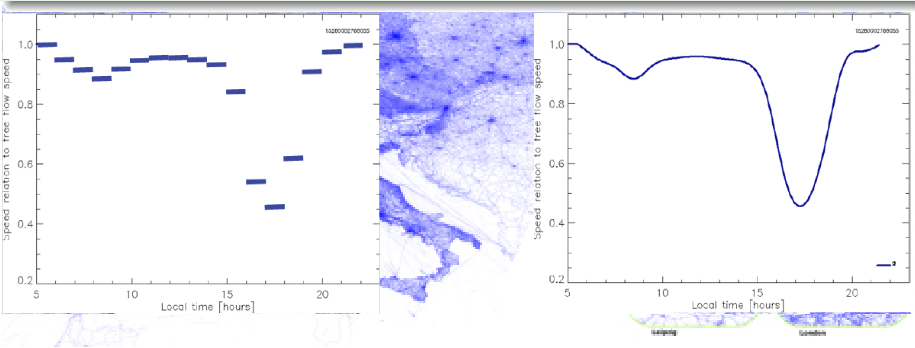
Raw traffic (speed probe) data

- **70 Million** contributing users
- **4 Billion** measurements per day
- **5 Trillion** measurements over 140 Billion Km
- every road segment measured 2000 times on average
- measured speeds in 5-min intervals



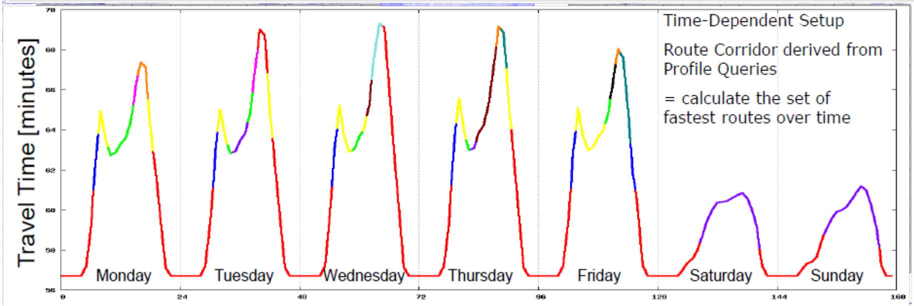
Raw traffic (speed probe) data

- **70 Million** contributing users
- **4 Billion** measurements per day
- **5 Trillion** measurements over 140 Billion Km
- every road segment measured 2000 times on average
- measured speeds in 5-min intervals



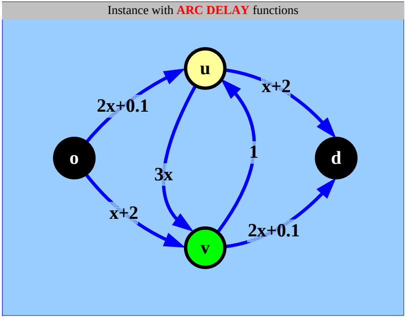
Raw traffic (speed probe) data

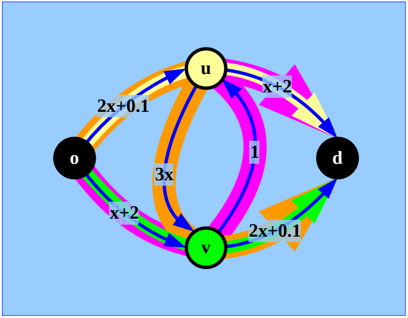
- **70 Million** contributing users
- **4 Billion** measurements per day
- **5 Trillion** measurements over 140 Billion Km
- every road segment measured 2000 times on average
- measured speeds in 5-min intervals



Main Issue: time-dependence

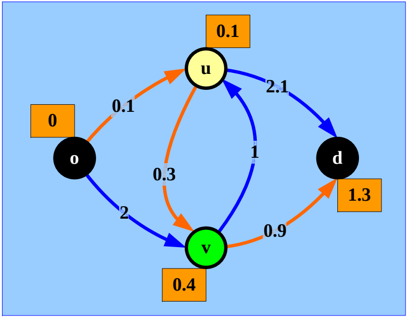
Time-Dependent Shortest Paths

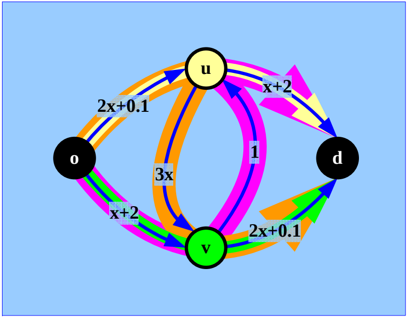




Q1 How would you commute **as fast as possible** from *o* to *d*, for a given departure time (from *o*)?

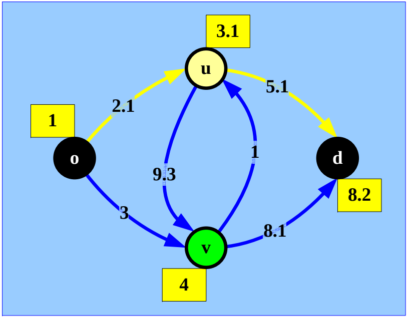
Time-Dependent Shortest Paths

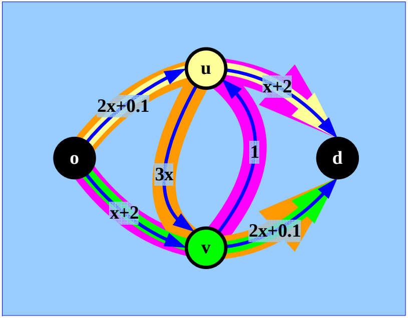




Q1 How would you commute **as fast as possible** from *o* to *d*, for a given departure time (from *o*)? Eg: $\boxed{t_o = 0}$

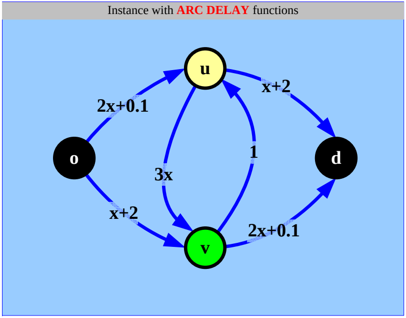
Time-Dependent Shortest Paths

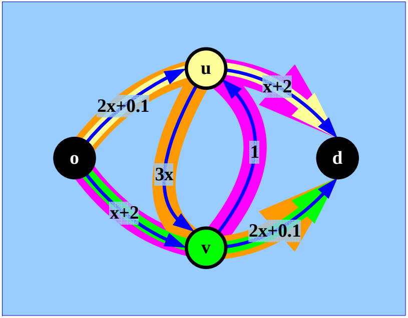




Q1 How would you commute **as fast as possible** from *o* to *d*, for a given departure time (from *o*)? Eg: $\boxed{t_o = 1}$

Time-Dependent Shortest Paths

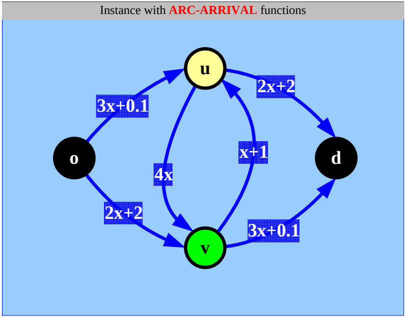


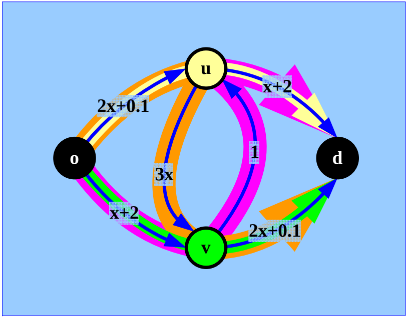


Q1 How would you commute **as fast as possible** from *o* to *d*, for a given departure time (from *o*)?

Q2 What if you are not sure about the departure time?

Time-Dependent Shortest Paths

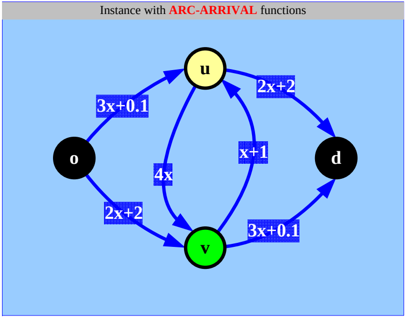


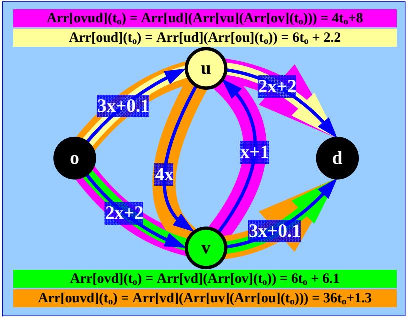


Q1 How would you commute **as fast as possible** from *o* to *d*, for a given departure time (from *o*)?

Q2 What if you are not sure about the departure time?

Time-Dependent Shortest Paths

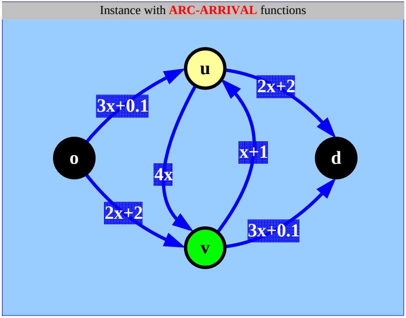


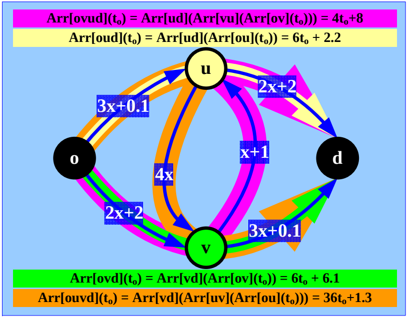


Q1 How would you commute **as fast as possible** from *o* to *d*, for a given departure time (from *o*)?

Q2 What if you are not sure about the departure time?

Time-Dependent Shortest Paths





Q1 How would you commute **as fast as possible** from *o* to *d*, for a given departure time (from *o*)?

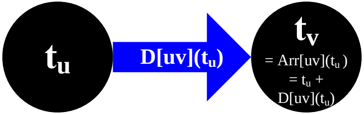
Q2 What if you are not sure about the departure time?

A

$$\text{shortest } od\text{–path} = \begin{cases} \textbf{orange path}, \text{ if} & t_o \in [0, 0.03] \\ \textbf{yellow path}, \text{ if} & t_o \in [0.03, 2.9] \\ \textbf{purple path}, \text{ if} & t_o \in [2.9, +\infty) \end{cases}$$

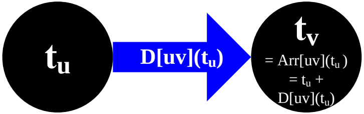
Time-Dependent Shortest Paths

- Directed graph $G = (V, A)$, $n = |V|$
- **Arc travel-time (arc-delay)** function $D[uv](t)$
- **Arc arrival** function $Arr[uv](t)$



Time-Dependent Shortest Paths

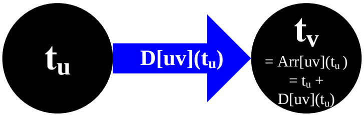
- Directed graph $G = (V, A)$, $n = |V|$
- **Arc travel-time (arc-delay)** function $D[uv](t)$
- **Arc arrival** function $Arr[uv](t)$



- $P_{o,d}$: od-paths; $p = (a_1, \ldots, a_k) \in P_{o,d}$
- **Path arrival / travel-time** functions
  $Arr[p](t_0) = Arr[a_k] \circ \cdots \circ Arr[a_1](t_0)$ (composition)
  $D[p](t_0) = Arr[p](t_0) - t_0$

Time-Dependent Shortest Paths

- Directed graph $G = (V, A)$, $n = |V|$
- **Arc travel-time (arc-delay)** function $D[uv](t)$
- **Arc arrival** function $Arr[uv](t)$

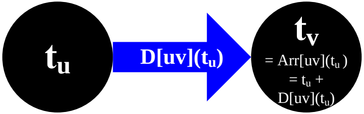


- $P_{o,d}$: od-paths; $p = (a_1, \ldots, a_k) \in P_{o,d}$
- **Path arrival / travel-time** functions
  $Arr[p](t_0) = Arr[a_k] \circ \cdots \circ Arr[a_1](t_0)$ (composition)
  $D[p](t_0) = Arr[p](t_0) - t_0$

- **Earliest-arrival / Shortest-travel-time** functions
  $Arr[o,d](t_0) = \min_{p \in P_{o,d}} \{ Arr[p](t_0) \}$
  $D[o,d](t_0) = Arr[o,d](t_0) - t_0$

Time-Dependent Shortest Paths

- Directed graph $G = (V, A)$, $n = |V|$
- **Arc travel-time (arc-delay)** function $D[uv](t)$
- **Arc arrival** function $Arr[uv](t)$



- $P_{o,d}$: od-paths; $p = (a_1, \ldots, a_k) \in P_{o,d}$
- **Path arrival / travel-time** functions
  $Arr[p](t_0) = Arr[a_k] \circ \cdots \circ Arr[a_1](t_0)$ (composition)
  $D[p](t_0) = Arr[p](t_0) - t_0$

- **Earliest-arrival / Shortest-travel-time** functions
  $Arr[o,d](t_0) = \min_{p \in P_{o,d}} \{ Arr[p](t_0) \}$
  $D[o,d](t_0) = Arr[o,d](t_0) - t_0$

Goals

1. For departure-time t_o from o, determine $t_d = Arr[o,d](t_o)$
2. Provide a **succinct representation** of $Arr[o,d]$ (or $D[o,d]$)

FIFO vs non-FIFO Arc Delays

- **FIFO Arc-Delays:** slopes of arc-delay functions ≥ -1
  $\equiv$ non-decreasing arc-arrival functions

FIFO vs non-FIFO Arc Delays

- **FIFO Arc-Delays:** slopes of arc-delay functions ≥ -1
  $\equiv$ non-decreasing arc-arrival functions

- **Non-FIFO Arc-Delays**
  - **Forbidden waiting:** $\nexists$ subpath optimality; NP-hard [Orda-Rom (1990)]
  - **Unrestricted waiting:** $\equiv$ FIFO (arbitrary waiting) [Dreyfus (1969)]

Complexity of Time-Dependent Shortest Path

FIFO, piecewise-linear arc-delay functions; K: total $\#$ number of breakpoints

- Given $od-$pair and departure time t_o from o: **time-dependent** Dijkstra [Dreyfus (1969), Orda-Rom (1990)]

Complexity of Time-Dependent Shortest Path

FIFO, piecewise-linear arc-delay functions; K: total $\#$ number of breakpoints

- Given od–pair and departure time t_o from o: **time-dependent** Dijkstra [Dreyfus (1969), Orda-Rom (1990)]
- Time-dependent shortest path heuristics: only empirical evidence [e.g., Delling & Wagner 2009; Batz etal, 2009]

Complexity of Time-Dependent Shortest Path

FIFO, piecewise-linear arc-delay functions; K: total $\#$ number of breakpoints

- Given od–pair and departure time t_o from o: **time-dependent** Dijkstra [Dreyfus (1969), Orda-Rom (1990)]
- Time-dependent shortest path heuristics: only empirical evidence [e.g., Delling & Wagner 2009; Batz etal, 2009]
- Complexity of computing **succinct representations** ??

Complexity of Time-Dependent Shortest Path

FIFO, piecewise-linear arc-delay functions; K: total $\#$ number of breakpoints

- Given od–pair and departure time t_o from o: **time-dependent** Dijkstra [Dreyfus (1969), Orda-Rom (1990)]
- Time-dependent shortest path heuristics: only empirical evidence [e.g., Delling & Wagner 2009; Batz etal, 2009]
- Complexity of computing **succinct representations** ??
  - Open till recently ...

Complexity of Time-Dependent Shortest Path

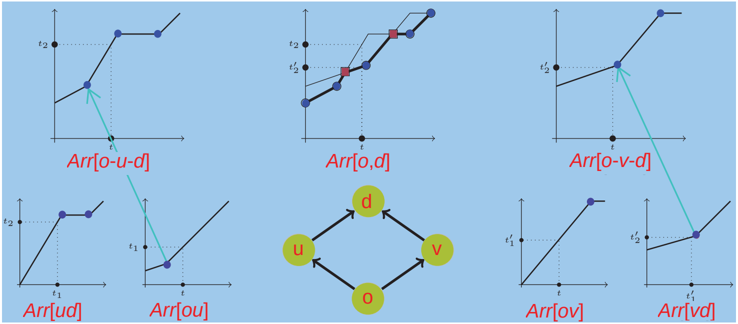
FIFO, piecewise-linear arc-delay functions; K: total $\#$ number of breakpoints

- Given od–pair and departure time t_o from o: **time-dependent** Dijkstra [Dreyfus (1969), Orda-Rom (1990)]
- Time-dependent shortest path heuristics: only empirical evidence [e.g., Delling & Wagner 2009; Batz etal, 2009]
- Complexity of computing **succinct representations** ??
  - Open till recently ...
  - $Arr[o, d]$: $O((K+1) \cdot n^{\Theta(\log(n))})$ space [Foschini-Hershberger-Suri (2011)]

Complexity of Time-Dependent Shortest Path

FIFO, piecewise-linear arc-delay functions; K: total $\#$ number of breakpoints

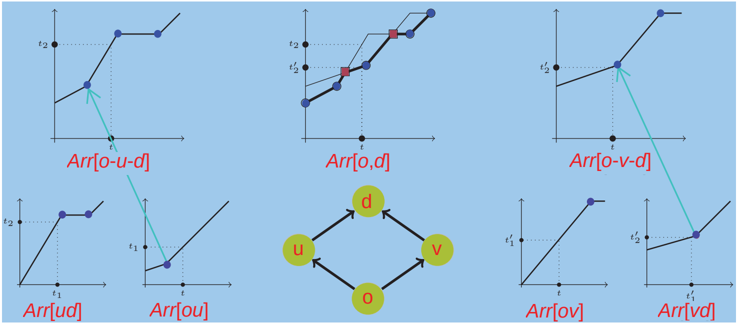
- Given $od-$pair and departure time t_o from o: **time-dependent** Dijkstra [Dreyfus (1969), Orda-Rom (1990)]
- Time-dependent shortest path heuristics: only empirical evidence [e.g., Delling & Wagner 2009; Batz etal, 2009]
- Complexity of computing **succinct representations** ??
  - Open till recently ...
  - $Arr[o, d]$: $O((K+1) \cdot n^{\Theta(\log(n))})$ space [Foschini-Hershberger-Suri (2011)]



- **Primitive Breakpoint (PB):** Departure-time b_{xy} from x at which $Arr[xy]$ changes slope

Exact Succinct Representation

Why so high complexity ?



- **Primitive Breakpoint (PB):** Departure-time b_{xy} from x at which $Arr[xy]$ changes slope
- **Minimization Breakpoint (MB):** Departure-time b_x from origin o such that $Arr[o, x]$ changes slope due to **min** operator at x

Complexity of Time-Dependent Shortest Path

FIFO, piecewise-linear arc-delay functions; K: total $\#$ number of breakpoints

- Given *od*–pair and departure time t_o from o: **time-dependent** Dijkstra [Dreyfus (1969), Orda-Rom (1990)]
- Time-dependent shortest path heuristics: only empirical evidence [e.g., Delling & Wagner 2009; Batz etal, 2009]
- Complexity of computing **succinct representations** ??
  - Open till recently ...
  - $Arr[o,d]$: $O((K+1)\cdot n^{\Theta(\log(n))})$ space [Foschini-Hershberger-Suri (2011)]
  - $D[o,d]$: $O(K+1)$ space for point-to-point $(1+\varepsilon)$–approximation [Dehne-Omran-Sack (2010), Foschini-Hershberger-Suri (2011)]

Complexity of Time-Dependent Shortest Path

FIFO, piecewise-linear arc-delay functions; K: total $\#$ number of breakpoints

- **Question 1**: $\exists$ data structure (**distance oracle**) that
  - requires **reasonable space** ?
  - allows answering **distance queries** efficiently ?

Complexity of Time-Dependent Shortest Path

FIFO, piecewise-linear arc-delay functions; K: total $\#$ number of breakpoints

- **Question 1**: $\exists$ data structure (**distance oracle**) that
  - requires **reasonable space** ?
  - allows answering **distance queries** efficiently ?
- **Trivial solution I:** Precompute all $(1+\varepsilon)-$approximate distance summaries for every *od*-pair
  - $\mathrm{O}\big(n^2(K+1)\big)$ space
  - $\mathrm{O}(\log\log(K))$ query time
  - $(1+\varepsilon)-$stretch

Complexity of Time-Dependent Shortest Path

FIFO, piecewise-linear arc-delay functions; K: total $\#$ number of breakpoints

- **Question 1**: $\exists$ data structure (**distance oracle**) that
  - requires **reasonable space** ?
  - allows answering **distance queries** efficiently ?
- **Trivial solution I:** Precompute all $(1+\varepsilon)-$approximate distance summaries for every *od*-pair
  - $\mathrm{O}\big(n^2(K+1)\big)$ space
  - $\mathrm{O}(\log\log(K))$ query time
  - $(1+\varepsilon)-$stretch
- **Trivial solution II:** No preprocessing, respond to queries with TD-Dijkstra
  - $\mathrm{O}(n+m+K)$ space
  - $\mathrm{O}([m+n\log(n)]\times\log\log(K))$ query time
  - $1-$stretch

Complexity of Time-Dependent Shortest Path

FIFO, piecewise-linear arc-delay functions; K: total # number of breakpoints

- **Question 1**: $\exists$ data structure (**distance oracle**) that
  - requires **reasonable space** ?
  - allows answering **distance queries** efficiently ?
- **Trivial solution I:** Precompute all $(1+\varepsilon)-$approximate distance summaries for every *od*-pair
  - ☹ $\mathrm{O}\big(n^2(K+1)\big)$ space
  - ☺ $\mathrm{O}(\log\log(K))$ query time
  - ☺ $(1+\varepsilon)-$stretch
- **Trivial solution II:** No preprocessing, respond to queries with TD-Dijkstra
  - ☺ $\mathrm{O}(n+m+K)$ space
  - ☹ $\mathrm{O}([m+n\log(n)]\times\log\log(K))$ query time
  - ☺ $1-$stretch
- **Question 2**: can we do better ?
  - **subquadratic** space & **sublinear** query time
  - $\exists$ smooth tradeoff among space / query time / stretch ?

Our Contribution [Kontogiannis & Z, 2014]

1. Efficient time-dependent distance oracle:
   subquadratic space and time preprocessing, **sublinear** query time

Our Contribution [Kontogiannis & Z, 2014]

1. Efficient time-dependent distance oracle:
   subquadratic space and time preprocessing, **sublinear** query time

2. $(1+\varepsilon)$–approximate algorithm for computing one-to-all distances in $O(K+1)$ space (same complexity with P2P approximation algorithm by [Foschini-Hershberger-Suri (2011)])
   - Bisection-based approach
   - Closed form for max absolute error

Our Contribution [Kontogiannis & Z, 2014]

1. Efficient time-dependent distance oracle:
   subquadratic space and time preprocessing, **sublinear** query time

2. $(1+\varepsilon)$–approximate algorithm for computing one-to-all distances in $O(K+1)$ space (same complexity with P2P approximation algorithm by [Foschini-Hershberger-Suri (2011)])
   - Bisection-based approach
   - Closed form for max absolute error

3. Preprocessing: choose a set L of **landmarks** and $\forall(\ell, v) \in L \times V$, compute $(1+\varepsilon)$–approximate **distance summaries** $\Delta[\ell, v](t)$ ($D[\ell, v](t) \leq \Delta[\ell, v](t) \leq (1+\epsilon) \cdot D[\ell, v](t)$)

Our Contribution [Kontogiannis & Z, 2014]

1. Efficient time-dependent distance oracle:
   subquadratic space and time preprocessing, **sublinear** query time

2. $(1+\varepsilon)$–approximate algorithm for computing one-to-all distances in $O(K+1)$ space (same complexity with P2P approximation algorithm by [Foschini-Hershberger-Suri (2011)])
   - Bisection-based approach
   - Closed form for max absolute error

3. Preprocessing: choose a set L of **landmarks** and $\forall(\ell, v) \in L \times V$, compute $(1+\varepsilon)$–approximate **distance summaries** $\Delta[\ell, v](t)$ $(D[\ell, v](t) \leq \Delta[\ell, v](t) \leq (1+\epsilon) \cdot D[\ell, v](t))$

4. Answer arbitrary queries (o, d, t_o) using two **query algorithms** (FCA/RQA) that return $O(1)$ / $(1+\sigma)$-approximate distance values

Assumptions

$\boxed{\text{Q}}$ Static & undirected world $\longrightarrow$ **time-dependent** & **directed** world ?

Assumptions

$\boxed{\text{Q}}$ Static & undirected world $\longrightarrow$ **time-dependent** & **directed** world ?

ASSUMPTION 1 (bounded travel time slopes)

Slopes of $D[o,d] \in [-1, \Lambda_{\max}]$, for some constant $\Lambda_{\max} > 0$

Assumptions

$\boxed{\text{Q}}$ Static & undirected world $\longrightarrow$ **time-dependent** & **directed** world ?

ASSUMPTION 1 (bounded travel time slopes)

Slopes of $D[o, d] \in [-1, \Lambda_{max}]$, for some constant $\Lambda_{max} > 0$

ASSUMPTION 2 (bounded opposite trips)

$\exists \zeta \geq 1 : \forall (o, d) \in V \times V,\ \forall t \in [0, T],\ D[o, d](t) \leq \zeta \cdot D[d, o](t_o)$

Assumptions

$\boxed{\text{Q}}$ Static & undirected world $\longrightarrow$ **time-dependent** & **directed** world ?

ASSUMPTION 1 (bounded travel time slopes)

Slopes of $D[o, d] \in [-1, \Lambda_{max}]$, for some constant $\Lambda_{max} > 0$

ASSUMPTION 2 (bounded opposite trips)

$\exists \zeta \geq 1 : \forall (o, d) \in V \times V, \ \forall t \in [0, T], D[o, d](t) \leq \zeta \cdot D[d, o](t_o)$

Assumptions

Q Static & undirected world $\longrightarrow$ **time-dependent** & **directed** world ?

ASSUMPTION 1 (bounded travel time slopes)

Slopes of $D[o,d] \in [-1, \Lambda_{\max}]$, for some constant $\Lambda_{\max} > 0$

ASSUMPTION 2 (bounded opposite trips)

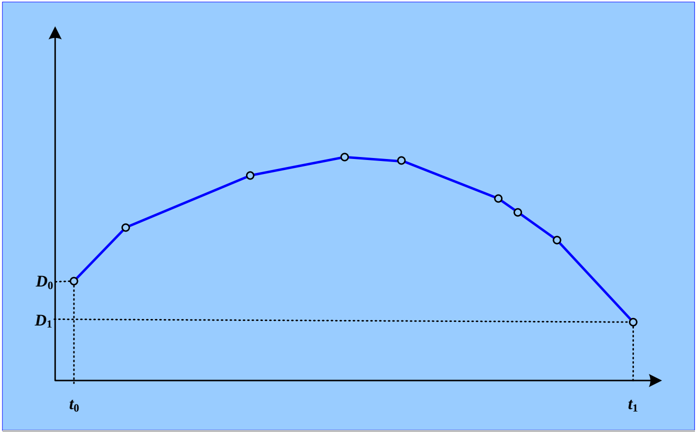
$\exists \zeta \geq 1 : \forall (o,d) \in V \times V,\ \forall t \in [0,T],\ D[o,d](t) \leq \zeta \cdot D[d,o](t_o)$

Experimental Analysis

Data Set	Type (source)	n	m	$\Lambda_{\max}$	ζ
Berlin	real-world (TomTom)	480 K	1135 K	0.185	1.54
W. Europe	benchmark (PTV)	18010 K	42188 K	6.186	1.18

Approximating Distance Functions via Bisection

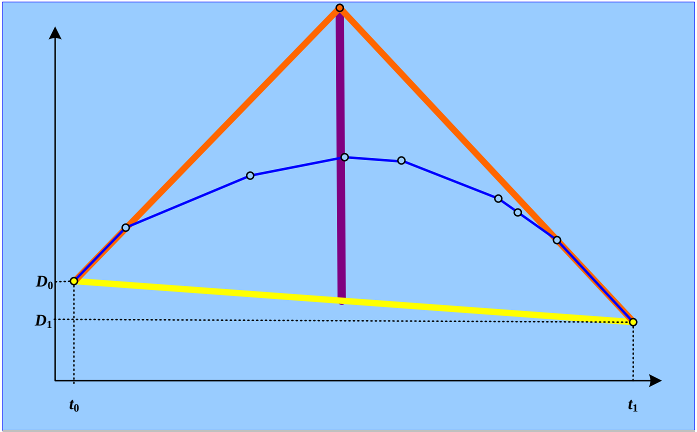
sample simultaneously all distance values from o, at mid-points of time intervals, until required approximation guarantee is achieved $\forall$ destinations



Example of Bisection Execution : INPUT = **UNKNOWN BLUE** function

Approximating Distance Functions via Bisection

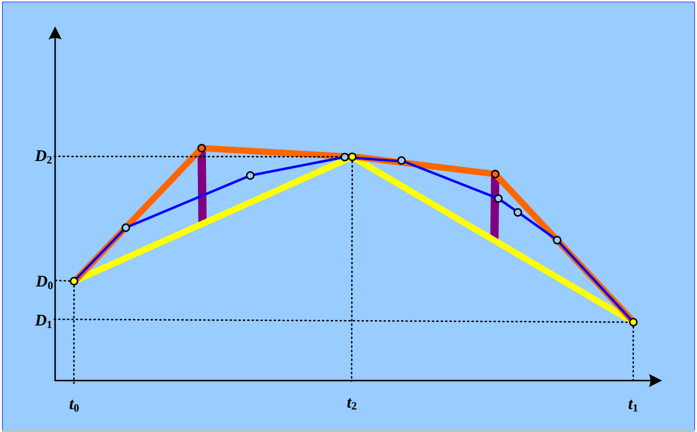
sample simultaneously all distance values from *o*, at mid-points of time intervals, until required approximation guarantee is achieved ∀ destinations



Example of Bisection Execution : **ORANGE** = Upper Bound, **YELLOW** = Lower Bound

Approximating Distance Functions via Bisection

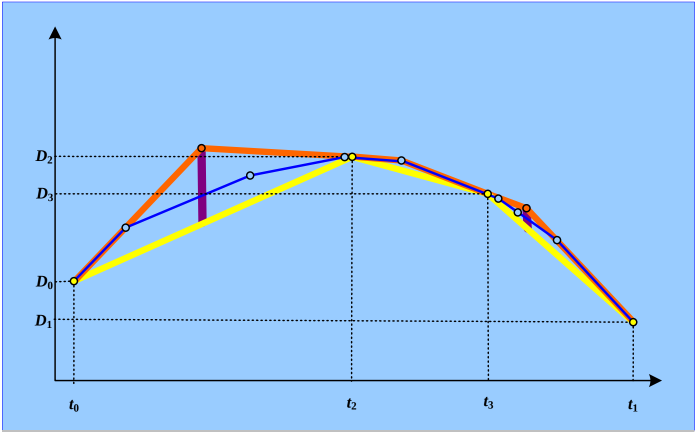
sample simultaneously all distance values from *o*, at mid-points of time intervals, until required approximation guarantee is achieved ∀ destinations



Example of Bisection Execution : Level-1 Recursion

Approximating Distance Functions via Bisection

sample simultaneously all distance values from o, at mid-points of time intervals, until required approximation guarantee is achieved $\forall$ destinations

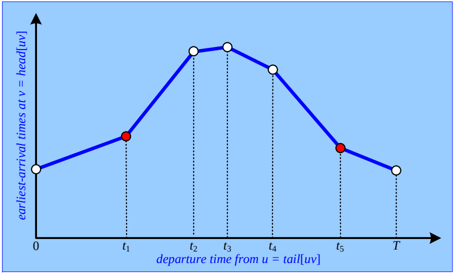


Example of Bisection Execution : Level-2 Recursion

Approximating Distance Functions via Bisection

For continuous, pwl arc-delays

1. Run Reverse TD-Dijkstra to project each concavity-spoiling PB to a primitive image (PI) of origin *o*
2. For each pair of consecutive PIs at *o*, run Bisection for the corresponding departure-times interval



3. Return the concatenation of approximate distance summaries

Landmark Selection and Preprocessing

K^*: total $\#$ number of concavity-spoiling breakpoints; $K^* < K$

- Landmark selection: $\forall v \in V$, $\Pr[v \in L] = \rho \in (0, 1)$
  [correctness is independent of the landmark selection]
- Preprocessing: $\forall \ell \in L$, compute $(1+\varepsilon)$–approximate distance functions $\Delta[\ell, v]$ to all $v \in V$

Landmark Selection and Preprocessing

K^*: total $\#$ number of concavity-spoiling breakpoints; $K^* < K$

- Landmark selection: $\forall v \in V$, $\Pr[v \in L] = \rho \in (0, 1)$
  [correctness is independent of the landmark selection]
- Preprocessing: $\forall \ell \in L$, compute $(1+\varepsilon)$–approximate distance functions $\Delta[\ell, v]$ to all $v \in V$

Preprocessing complexity

Landmark Selection and Preprocessing

K^*: total $\#$ number of concavity-spoiling breakpoints; $K^* < K$

- Landmark selection: $\forall v \in V$, $\Pr[v \in L] = \rho \in (0,1)$
  [correctness is independent of the landmark selection]
- Preprocessing: $\forall \ell \in L$, compute $(1+\varepsilon)$–approximate distance functions $\Delta[\ell, v]$ to all $v \in V$

Preprocessing complexity

- Space – asymptotically optimal

$$\boxed{\mathrm{O}\left((K^*+1)\cdot|L|\cdot n\cdot\frac{1}{\varepsilon}\cdot\max_{(\ell,v)\in L\times V}\left\{\log\left(\frac{D_{\max}[\ell,v](0,T)}{D_{\min}[\ell,v](0,T)}\right)\right\}\right)}$$

Landmark Selection and Preprocessing

K^*: total $\#$ number of concavity-spoiling breakpoints; $K^* < K$

- Landmark selection: $\forall v \in V$, $\Pr[v \in L] = \rho \in (0, 1)$
  [correctness is independent of the landmark selection]
- Preprocessing: $\forall \ell \in L$, compute $(1+\varepsilon)$–approximate distance functions $\Delta[\ell, v]$ to all $v \in V$

Preprocessing complexity

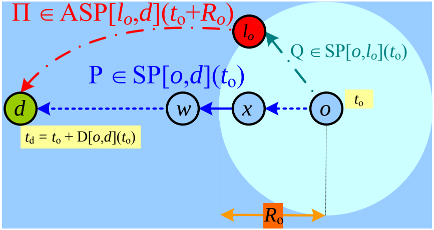
- Space – asymptotically optimal

$$\boxed{\mathrm{O}\left((K^* + 1) \cdot |L| \cdot n \cdot \frac{1}{\varepsilon} \cdot \max_{(\ell,v)\in L\times V} \left\{\log\left(\frac{D_{\max}[\ell,v](0,T)}{D_{\min}[\ell,v](0,T)}\right)\right\}\right)}$$

- Time (in number of TDSP-Probes)

$$\boxed{\mathrm{O}\left((K^* + 1) \cdot |L| \cdot \max_{(\ell,v)} \left\{\log\left(\frac{T\cdot(\Lambda_{\max}+1)}{\varepsilon D_{\min}[\ell,v](0,T)}\right)\right\} \cdot \frac{1}{\varepsilon} \max_{(\ell,v)} \left\{\log\left(\frac{D_{\max}[\ell,v](0,T)}{D_{\min}[\ell,v](0,T)}\right)\right\}\right)}$$

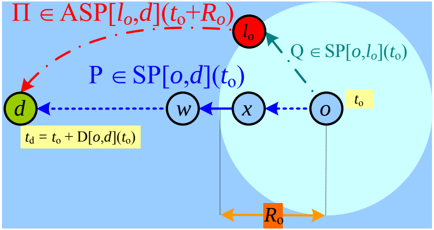
FCA: A constant-approximation query algorithm



Forward Constant Approximation

1. Grow TD-Dijkstra ball $B(o, t_o)$ until closest landmark ℓ_o or d is settled
2. **return** $sol_o = D[o, \ell_o](t_o) + \Delta[\ell_o, d](t_o + D[o, \ell_o](t_o))$

FCA: A constant-approximation query algorithm

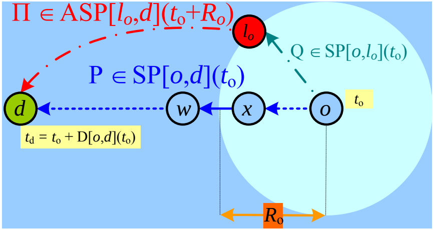


Forward Constant Approximation

1. Grow TD-Dijkstra ball $B(o, t_o)$ until closest landmark ℓ_o or d is settled
2. **return** $sol_o = D[o, \ell_o](t_o) + \Delta[\ell_o, d](t_o + D[o, \ell_o](t_o))$

FCA complexity

FCA: A constant-approximation query algorithm



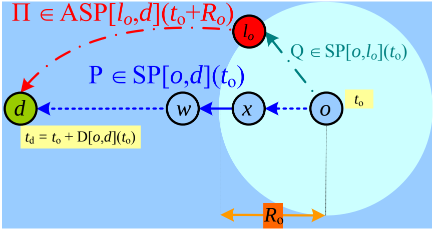
Forward Constant Approximation

1. Grow TD-Dijkstra ball $B(o, t_o)$ until closest landmark ℓ_o or d is settled
2. **return** $sol_o = D[o, \ell_o](t_o) + \Delta[\ell_o, d](t_o + D[o, \ell_o](t_o))$

FCA complexity

- Approximation guarantee: $\leq (1 + \epsilon + \psi) \cdot D[o, d](t_o)$
  $\psi = 1 + \Lambda_{max}(1 + \epsilon)(1 + 2\zeta + \Lambda_{max}\zeta) + (1 + \epsilon)\zeta$

FCA: A constant-approximation query algorithm



Forward Constant Approximation

1. Grow TD-Dijkstra ball $B(o, t_o)$ until closest landmark ℓ_o or d is settled
2. **return** $sol_o = D[o, \ell_o](t_o) + \Delta[\ell_o, d](t_o + D[o, \ell_o](t_o))$

FCA complexity

- Approximation guarantee: $\leq (1 + \epsilon + \psi) \cdot D[o, d](t_o)$
  $\psi = 1 + \Lambda_{\max}(1 + \epsilon)(1 + 2\zeta + \Lambda_{\max}\zeta) + (1 + \epsilon)\zeta$
- Query-time: $\mathrm{O}\left(\frac{1}{\rho} \cdot \ln\left(\frac{1}{\rho}\right) \log\log(K_{\max})\right)$

RQA: Boosting the Approximation Guarantee

Recursive Query Approximation

1. **while** recursion budget R not exhausted **do**
2. Grow TD-Dijkstra ball $B(w_i, t_i)$ until closest landmark ℓ_i is settled
3. $sol_i = D[o, w_i](t_o) + D[w_i, \ell_i](t_i) + \Delta[\ell_i, d](t_i + D[w_i, \ell_i](t_i))$
4. Run RQA at each boundary node of $B(w_i, t_i)$ with budget $R - 1$
5. **endwhile**
6. **return** best solution found

RQA: Boosting the Approximation Guarantee

Recursive Query Approximation

while recursion budget R not exhausted **do**

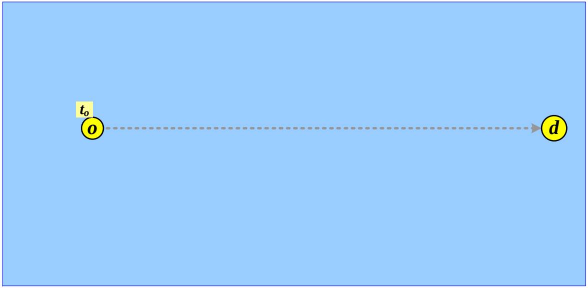
  Grow TD-Dijkstra ball $B(w_i, t_i)$ until closest landmark ℓ_i is settled

  $sol_i = D[o, w_i](t_o) + D[w_i, \ell_i](t_i) + \Delta[\ell_i, d](t_i + D[w_i, \ell_i](t_i))$

  Run RQA at each boundary node of $B(w_i, t_i)$ with budget $R - 1$

endwhile

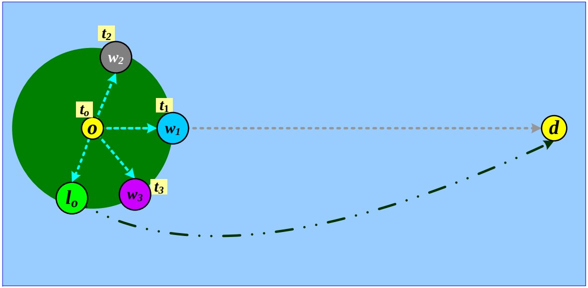
return best solution found



RQA: Boosting the Approximation Guarantee

Recursive Query Approximation

1. **while** recursion budget R not exhausted **do**
2. Grow TD-Dijkstra ball $B(w_i, t_i)$ until closest landmark ℓ_i is settled
3. $sol_i = D[o, w_i](t_o) + D[w_i, \ell_i](t_i) + \Delta[\ell_i, d](t_i + D[w_i, \ell_i](t_i))$
4. Run RQA at each boundary node of $B(w_i, t_i)$ with budget $R - 1$
5. **endwhile**
6. **return** best solution found

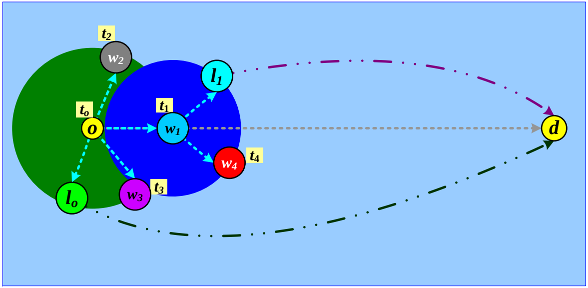


- Growing level-0 ball...

RQA: Boosting the Approximation Guarantee

Recursive Query Approximation

1. **while** recursion budget R not exhausted **do**
2. Grow TD-Dijkstra ball $B(w_i, t_i)$ until closest landmark ℓ_i is settled
3. $sol_i = D[o, w_i](t_o) + D[w_i, \ell_i](t_i) + \Delta[\ell_i, d](t_i + D[w_i, \ell_i](t_i))$
4. Run RQA at each boundary node of $B(w_i, t_i)$ with budget $R - 1$
5. **endwhile**
6. **return** best solution found

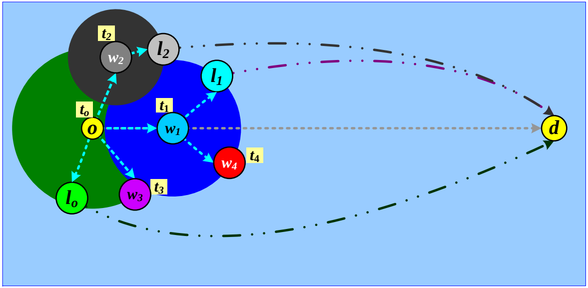


- Growing level-0 ball...
- Growing level-1 balls...

RQA: Boosting the Approximation Guarantee

Recursive Query Approximation

1.	**while** recursion budget R not exhausted **do**
2.	Grow TD-Dijkstra ball $B(w_i, t_i)$ until closest landmark ℓ_i is settled
3.	$sol_i = D[o, w_i](t_o) + D[w_i, \ell_i](t_i) + \Delta[\ell_i, d](t_i + D[w_i, \ell_i](t_i))$
4.	Run RQA at each boundary node of $B(w_i, t_i)$ with budget $R - 1$
5.	**endwhile**
6.	**return** best solution found

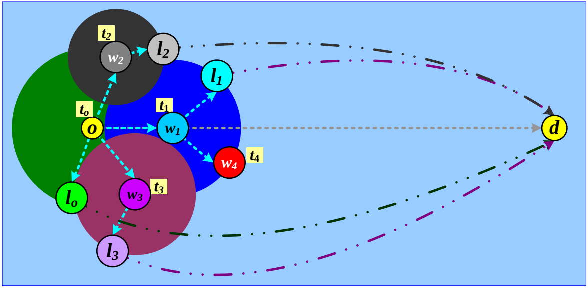


- Growing level-0 ball...
- Growing level-1 balls...

RQA: Boosting the Approximation Guarantee

Recursive Query Approximation

while recursion budget R not exhausted **do**
    Grow TD-Dijkstra ball $B(w_i, t_i)$ until closest landmark ℓ_i is settled
    $sol_i = D[o, w_i](t_o) + D[w_i, \ell_i](t_i) + \Delta[\ell_i, d](t_i + D[w_i, \ell_i](t_i))$
    Run RQA at each boundary node of $B(w_i, t_i)$ with budget $R - 1$
endwhile
return best solution found

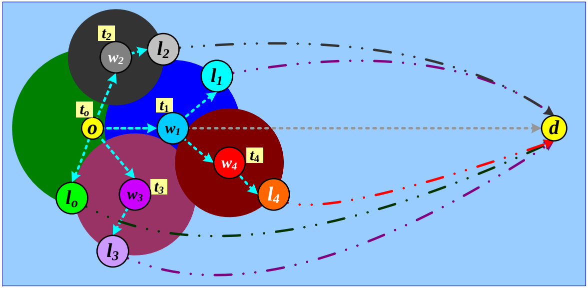


- Growing level-0 ball...
- Growing level-1 balls...

RQA: Boosting the Approximation Guarantee

Recursive Query Approximation

1. **while** recursion budget R not exhausted **do**
2. Grow TD-Dijkstra ball $B(w_i, t_i)$ until closest landmark ℓ_i is settled
3. $sol_i = D[o, w_i](t_o) + D[w_i, \ell_i](t_i) + \Delta[\ell_i, d](t_i + D[w_i, \ell_i](t_i))$
4. Run RQA at each boundary node of $B(w_i, t_i)$ with budget $R - 1$
5. **endwhile**
6. **return** best solution found

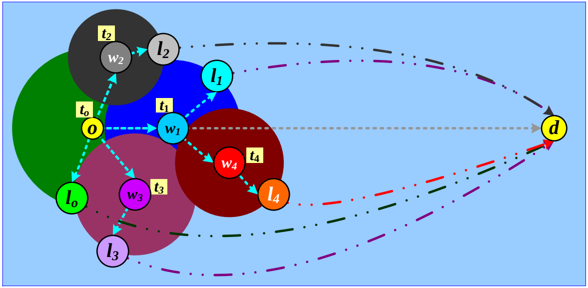


- Growing level-0 ball...
- Growing level-1 balls...
- Growing level-2 balls...

RQA: Boosting the Approximation Guarantee

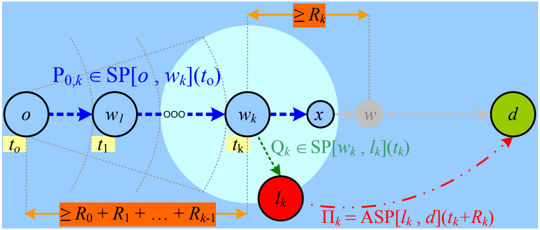
Recursive Query Approximation

1. **while** recursion budget R not exhausted **do**
2. Grow TD-Dijkstra ball $B(w_i, t_i)$ until closest landmark ℓ_i is settled
3. $sol_i = D[o, w_i](t_o) + D[w_i, \ell_i](t_i) + \Delta[\ell_i, d](t_i + D[w_i, \ell_i](t_i))$
4. Run RQA at each boundary node of $B(w_i, t_i)$ with budget $R - 1$
5. **endwhile**
6. **return** best solution found

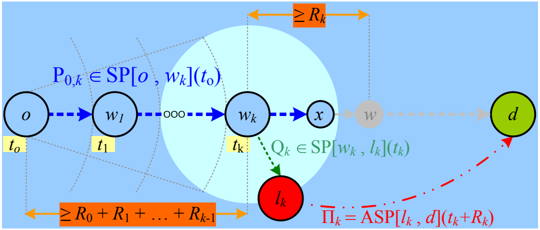


- Growing level-0 ball...
- Growing level-1 balls...
- Growing level-2 balls...
- ...

RQA: Boosting the Approximation Guarantee

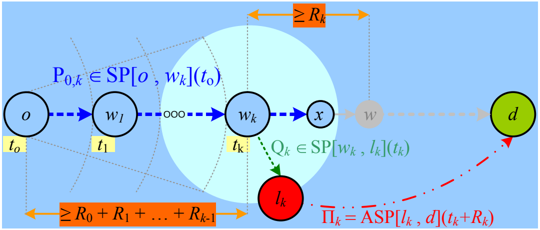


RQA: Boosting the Approximation Guarantee



1. One of the discovered approximate *od*–paths has all its ball centers at nodes of the (unknown) shortest od-path

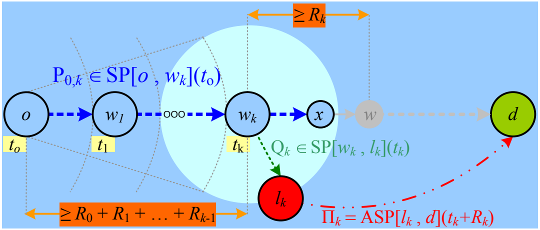
RQA: Boosting the Approximation Guarantee



1. One of the discovered approximate *od*–paths has all its ball centers at nodes of the (unknown) shortest od-path
2. Optimal **prefix subpaths** improve approximation guarantee:

$$\boxed{\forall\beta > 1,\ \forall\lambda \in (0,1),\ \lambda \cdot OPT + (1-\lambda)\cdot\beta\cdot OPT < \beta \cdot OPT}$$

RQA: Boosting the Approximation Guarantee

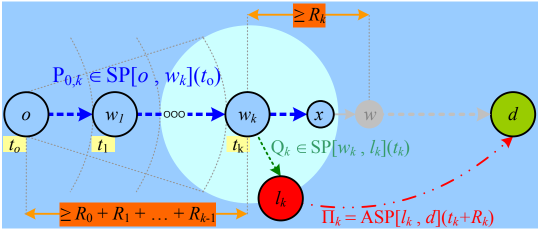


1. One of the discovered approximate *od*−paths has all its ball centers at nodes of the (unknown) shortest od-path
2. Optimal **prefix subpaths** improve approximation guarantee:

$$\boxed{\forall\beta > 1,\ \forall\lambda \in (0,1),\ \lambda \cdot OPT + (1-\lambda) \cdot \beta \cdot OPT < \beta \cdot OPT}$$

3. Approximation guarantee for **suffix subpath** to destination depends on last ball radius

RQA: Boosting the Approximation Guarantee



1. One of the discovered approximate *od*−paths has all its ball centers at nodes of the (unknown) shortest od-path
2. Optimal **prefix subpaths** improve approximation guarantee:

$$\boxed{\forall\beta > 1,\ \forall\lambda \in (0,1),\ \lambda \cdot OPT + (1-\lambda)\cdot\beta\cdot OPT < \beta \cdot OPT}$$

3. Approximation guarantee for **suffix subpath** to destination depends on last ball radius
4. $R = O(1)$ suffices to ensure guarantee close to $1+\varepsilon$

RQA: Boosting the Approximation Guarantee

RQA Complexity

RQA: Boosting the Approximation Guarantee

RQA Complexity

- Approximation guarantee: $1+\sigma = 1+\varepsilon \cdot \frac{(1+\varepsilon/\psi)^{R+1}}{(1+\varepsilon/\psi)^{R+1}-1}$

RQA: Boosting the Approximation Guarantee

RQA Complexity

- Approximation guarantee: $1 + \sigma = 1 + \varepsilon \cdot \frac{(1+\varepsilon/\psi)^{R+1}}{(1+\varepsilon/\psi)^{R+1}-1}$
- Query-time: $\mathrm{O}\left(\left(\frac{1}{\rho}\right)^{R+1} \cdot \ln\left(\frac{1}{\rho}\right) \log\log(K_{\max})\right)$

Summary of Complexity Bounds

Preprocessed	Preproc. Space	Preproc. Time	Query Time
All-To-All	$\mathrm{O}((K^* + 1)n^2)$	$\mathrm{O}\left(n^2 \log(n) \cdot \log\log(K_{\max}) \cdot (K^* + 1)\right)$	$\mathrm{O}(\log\log(K^*))$
Nothing	$\mathrm{O}(n + m + K)$	$\mathrm{O}(1)$	$\mathrm{O}\left(n \log(n) \cdot \log\log(K_{\max})\right)$
Landmarks-To-All **[This work]**	$\mathrm{O}(\rho n^2 (K^* + 1))$	$\mathrm{O}\left(\rho n^2 \log(n) \cdot \log\log(K_{\max}) \cdot (K^* + 1)\right)$	$\mathrm{O}\left(\left(\frac{1}{\rho}\right)^{R+1} \cdot \log\left(\frac{1}{\rho}\right) \cdot \log\log(K_{\max})\right)$

- $m = O(n)$; $K_{\max}$: max number of breakpoints in an arc-delay function
- K^*: total $\#$ number of concavity-spoiling breakpoints
- $K^* < K$ (total $\#$ number of breakpoints)

Summary of Complexity Bounds

Preprocessed	Preproc. Space	Preproc. Time	Query Time
All-To-All	$\mathrm{O}((K^* + 1)n^2)$	$\mathrm{O}\left(\begin{array}{c} n^2 \log(n) \\ \cdot \log\log(K_{\max}) \\ \cdot(K^* + 1) \end{array}\right)$	$\mathrm{O}(\log\log(K^*))$
Nothing	$\mathrm{O}(n + m + K)$	$\mathrm{O}(1)$	$\mathrm{O}\left(\begin{array}{c} n\log(n)\cdot \\ \log\log(K_{\max}) \end{array}\right)$
Landmarks-To-All **[This work]**	$\mathrm{O}(\rho n^2 (K^* + 1))$	$\mathrm{O}\left(\begin{array}{c} \rho n^2 \log(n) \\ \cdot \log\log(K_{\max}) \\ \cdot(K^* + 1) \end{array}\right)$	$\mathrm{O}\left(\begin{array}{c} \left(\frac{1}{\rho}\right)^{R+1} \cdot \log\left(\frac{1}{\rho}\right) \\ \cdot \log\log(K_{\max}) \end{array}\right)$

- $m = O(n)$; $K_{\max}$: max number of breakpoints in an arc-delay function ($K_{\max} \in \mathrm{O}(1)$)
- K^*: total $\#$ number of concavity-spoiling breakpoints
- $K^* < K$ (total $\#$ number of breakpoints); $K^* \in \mathrm{O}(\mathrm{polylog}(n))$
- $\rho = n^{-\alpha}$, $0 < \alpha < \frac{1}{R+1}$

Summary of Complexity Bounds

Preprocessed	Preproc. Space	Preproc. Time	Query Time
All-To-All	$\tilde{O}(n^2)$	$\tilde{O}(n^2 \log(n))$	$\mathrm{O}(\log\log\log(n))$
Nothing	$\mathrm{O}(n+m+K)$	$\mathrm{O}(1)$	$\mathrm{O}(n\log(n))$
Landmarks-To-All **[This work]**	$\tilde{O}(n^{2-\alpha})$	$\tilde{O}(n^{2-\alpha})$	$\tilde{O}\left(n^{(R+1)\alpha}\right)$

- $m = O(n)$; $K_{\max}$: max number of breakpoints in an arc-delay function ($K_{\max} \in \mathrm{O}(1)$)
- K^*: total $\#$ number of concavity-spoiling breakpoints
- $K^* < K$ (total $\#$ number of breakpoints); $K^* \in \mathrm{O}(\mathrm{polylog}(n))$
- $\rho = n^{-\alpha}$, $0 < \alpha < \frac{1}{R+1}$

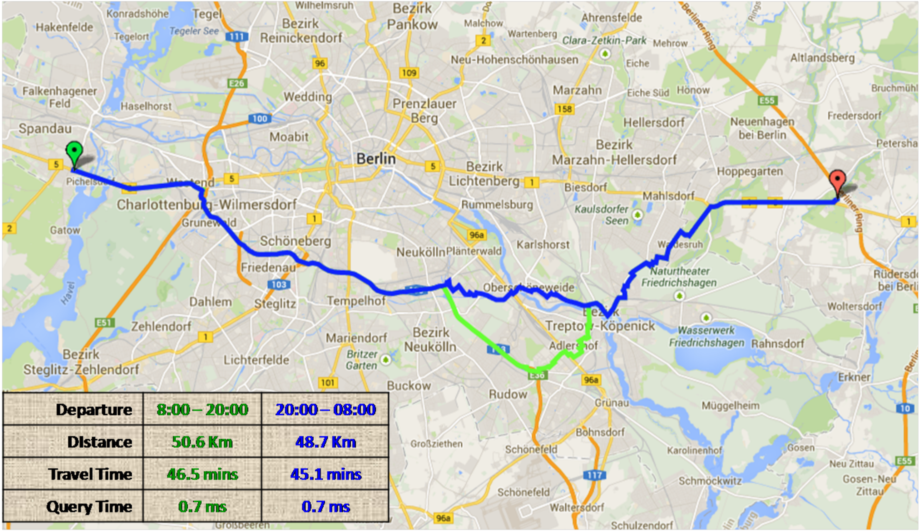
Distance Oracle: Practical Issues

- Berlin data set: $n = 480000$, $m = 1135000$
- Time resolution: 10.3 msec

Landmarks		**FCA**		**RQA**		
Method	Number	ms	σ (%)	ms	σ (%)	**TD-Dijkstra** (ms)
METIS	1061	0.381	2.201	2.349	0.483	77.424
METIS	2063	0.152	1.115	0.700	0.314	77.424
Random	1000	0.195	1.634	1.692	0.575	77.424
Random	2000	0.107	1.065	0.771	0.445	77.424
KAHIP	1053	0.362	2.165	2.015	0.382	77.424
KAHIP	2015	0.148	1.405	0.655	0.298	77.424

- Speedup (over TDD) > 723
- Query time of previous time-dependent heuristics $\in [1, 1.5]$ ms

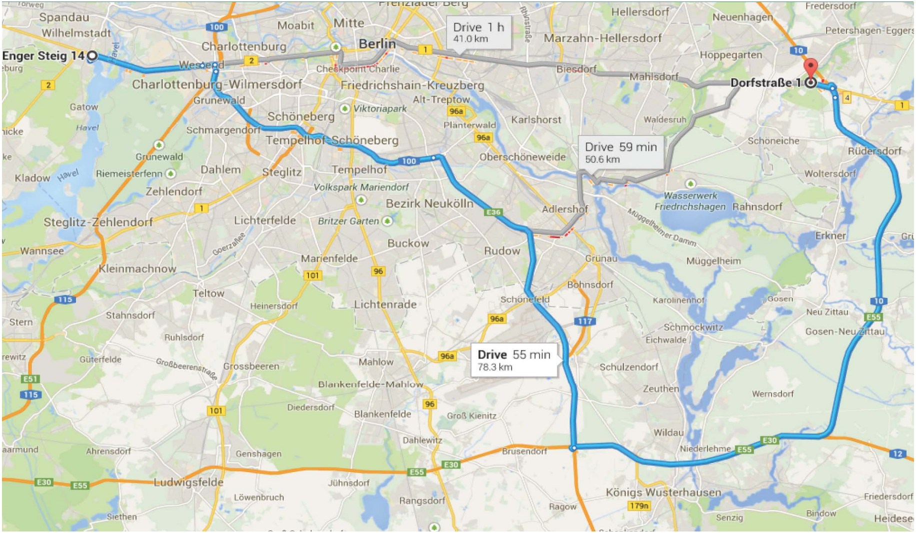
Distance Oracle: Practical Issues



Departure	8:00 – 20:00	20:00 – 08:00
Distance	50.6 Km	48.7 Km
Travel Time	46.5 mins	45.1 mins
Query Time	0.7 ms	0.7 ms

Distance Oracle: Practical Issues

Google Maps, Tuesday 15:45



Conclusions & Open Issues

- First efficient time-dependent distance oracle

Conclusions & Open Issues

- First efficient time-dependent distance oracle
- Approach sensitive to network's
  - degree of asymmetry (ζ)
  - rate of (shortest-)travel-time evolution ($\Lambda_{\max}$)

Conclusions & Open Issues

- First efficient time-dependent distance oracle
- Approach sensitive to network's
  - degree of asymmetry (ζ)
  - rate of (shortest-)travel-time evolution ($\Lambda_{\max}$)
- Builds upon new approximate algorithm for computing one-to-all time-dependent distance summaries

Conclusions & Open Issues

- First efficient time-dependent distance oracle
- Approach sensitive to network's
  - degree of asymmetry (ζ)
  - rate of (shortest-)travel-time evolution (Λ_{max})
- Builds upon new approximate algorithm for computing one-to-all time-dependent distance summaries
- Quite efficient in practice

Conclusions & Open Issues

- First efficient time-dependent distance oracle
- Approach sensitive to network's
  - degree of asymmetry (ζ)
  - rate of (shortest-)travel-time evolution (Λ_{max})
- Builds upon new approximate algorithm for computing one-to-all time-dependent distance summaries
- Quite efficient in practice
- **Open:** can we avoid dependence on K^* ?

Outline

Summary

- Transportation networks give rise to large-scale optimization problems
- Novel algorithms can have a great impact in their efficient and effective solution

Thank you for your attention

Questions

Τέλος Ενότητας

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.

Σημείωμα Αναφοράς

Copyright Πανεπιστήμιο Πατρών, Χρήστος Ζαρολιάγκης «Μελέτη Περιπτώσεων στη Λήψη Αποφάσεων: Algorithms for Transport Optimisation: Theory and Practice». Έκδοση: 1.0. Πάτρα 2015. Διαθέσιμο από τη δικτυακή διεύθυνση:

https://eclass.upatras.gr/courses/MATH959/

Σημείωμα Αδειοδότησης

Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση, Όχι Παράγωγα Έργα 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».

[1] http://creativecommons.org/licenses/by-nc-nd/4.0/

Ως **Μη Εμπορική** ορίζεται η χρήση:

- που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
- που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
- που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

Σημείωμα Χρήσης Έργων Τρίτων

Το Έργο αυτό κάνει χρήση των ακόλουθων έργων:

Διαφάνεια 5, 6, 114-121:

http://www.finanzen.net/nachricht/TomTom-Users-Capture-the-Road-Network-3-000-Times-1485950

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει) μαζί με τους συνοδευόμενους υπερσυνδέσμους.