Scalable and Congestion-aware Routing for Autonomous Mobility-on-Demand via Frank-Wolfe Optimization

Kiril Solovey
Aeronautics & Astronautics, Stanford
kirilsol@stanford.edu / kirilsol.github.io

* Joint work with Mauro Salazar and Marco Pavone
** Submitted to RSS2019
About me

• BSc., MSc., and PhD. in Computer Science, from Tel Aviv University, Israel

• Now, a PostDoc in Marco Pavone’s ASL group, Aeronautics and Astronautics, Stanford

• Algorithmic foundations of robotics, and in particular motion planning
  • Single and multi-robot systems
  • Combinatorial and sampling-based techniques
  • Complexity and optimality guarantees

• Recently began working on autonomous mobility-on-demand
  • “Motion planning on steroids”
Contribution

• We present an efficient approach for autonomous-mobility-on-demand routing
  • Computes near-optimal solutions for city-scale scenarios within seconds
    • E.g., Manhattan roadmap with 150K user request within 15sec
    • Much faster (at least 10x) than previous work
  • Congestion aware: accounts for congestion induced by the system
    • The first approach to fully incorporate a volume-delay functions (BPR)
Contribution

- **Main idea:** transform AMoD into a *traffic-assignment problem* (TAP)
- TAP does not involve rebalancing vehicles:
  - As cars are private, once a passenger arrives to its destination we’re done
- **Approach:**
  - Model AMoD as a mathematical program
  - Transform it into TAP, that is modeled by *convex program*
  - Employ the *Frank-Wolfe (FW) algorithm* for convex optimization
- **Important:** FW brakes the optimization into a sequence of shortest path queries on a static graph!
  - Very efficient in practice, especially with modern pathfinding techniques
  - We use *Contraction Hierarchies*
Outline

1. Related work
2. Preliminaries (ingredients and formal definition of problem)
3. Traffic assignment and Frank-Wolfe algorithm
4. Algorithm: AMoD routing as TAP
   • Theoretical analysis
5. Experimental results (Manhattan, NY)
6. Conclusion and future work
AMoD: related work

• Huge amount of work on Human-Robot interaction in a transportation
  • Sadigh et al., Pavone et al., and many more...
• We are interested in Robot-Robot interaction
  • Operating a centrally controlled fleet of multiple vehicles
• Early work [Pavone et al.; 2007(!)] considered simplified continuous domains
• More recent work considers real roadmap topologies
  • Queuing-theoretic models [Iglesias et al; 2016], [Zhang and Pavone; 2016]
  • Network flow models [Spieser et al; 2014], [Rossi et al; 2018]
  • Receding-horizon implementation [Iglesias et al; 2018], [Tsao et al; 2018]
• Extensions:
  • Interaction with power grid [Rossi et al; 2018]
  • Interaction with public transit [Salazar et al; 2018]
  • Ridesharing [Cap and Alonso-Mora; 2018], [Tsao et al; 2018]
AMoD: related work

• Algorithmic details
  • Typically use mathematical programming (linear / convex)
  • Difficult to parallelize
  • Sometime limited to small-scale scenarios

• Congestion awareness
  • Majority of approaches ignore congestion affects
  • Threshold model [Rossi et al; 2018]
  • Linear approximation of volume delay function [Salazar et al; 2018]

*Scalable and Congestion-aware AMoD Routing*
AMoD: related work

• Our work is inspired by recent progress in traffic assignment
  • Combines classic Frank-Wolfe algorithm and modern pathfinding techniques
  • Fully congestion aware
  • Computes traffic flows over full road network (Stuttgart, 2.7mil people), for 2h scenario in mere seconds!

**Question:** Can we exploit the similarity between TAP and AMoD, and extend those tools to our setting?

Preliminaries: roadmap network

- Road network is modeled by a directed graph $G = (V, E)$
- Each edge $(i, j) \in E$ associated with a cost function $c_{ij}: \mathbb{N}_+ \to \mathbb{N}_+$
  - Represents the travel as a function of total flow $x_{ij}$ over edge
- We use the popular BPR function:
  \[
  c_{ij}(x_{ij}) = BPR(x_{ij}, \kappa_{ij}, \phi_{ij}) := \phi_{ij} \cdot \left(1 + 0.15 \cdot \left(\frac{x_{ij}}{\kappa_{ij}}\right)^4\right),
  \]
  where $\kappa_{ij} > 0$ is capacity, and $\phi_{ij}$ is free-flow speed
Preliminaries: user requests

• Travel demand represented by passenger requests
  \[ OD = \{(\lambda_m, o_m, d_m)\}_{m=1}^M \]
  • \( \lambda_m > 0 \) represents customer #
  • \( o_m, d_m \in V \) are origin and destination

• \( \lambda_m \) should be interpreted as the average number of customers for the given time period

• To summarize, the input consists of \( G \) and \( OD \)
Traffic Assignment Problem (TAP)

• “Assign requests to paths to minimize the total travel time”
• We model this as a (convex) program
• Let $x_{ijm}$ be the flow of commodity $m$ for edge $(i, j)$
• Constraints: (1) conservation of flow, (2) nonnegative flow

\[
\sum_{j \in V_i^+} x_{ijm} - \sum_{j \in V_i^-} x_{jim} = \lambda_{im}, \quad \forall i \in V, m \in M, \tag{1}
\]

where $\lambda_{im} := \begin{cases} 
\lambda_m, & \text{if } o_m = i, \\
-\lambda_m, & \text{if } o_m = i, \\
0, & \text{otherwise}
\end{cases}$

and $V_i^-, V_i^+$ denote the heads/tails of edges leaving/entering $i \in V$

\[
x_{ijm} \geq 0, \quad \forall (i, j) \in E. \tag{2}
\]
Traffic Assignment Problem (TAP)

Definition (TAP): Given $G, OD$, minimize the expression

$$F_E(x) = \sum_{(i,j) \in E} x_{ij}c_{ij}(x_{ij}),$$

subject to (1),(2), where $x := \{x_{ij} = \sum_{m \in M} x_{ijm} \mid (i, j) \in E\}$.

**Remarks:**

- It corresponds to minimal-cost multi-commodity flow (NP-hard)
- $F_E(x)$ represents the system optimum, i.e., total travel time
- If we replace $x_{ij}c_{ij}(x_{ij})$ with $\int_0^{x_{ij}} c_{ij}(s) \, ds$, then we’ll get the user equilibrium
Autonomous mobility-on-demand (AMoD)

- Vehicles in AMoD systems perform two types of tasks:
  1. Occupied vehicles drive passengers from $o_m$ to $d_m$
  2. Empty vehicles need to drive to the next origin node $o_m'$

- Observations concerning the latter task
  - It is not represented in the TAP formulation
  - It consists of assignment AND routing

- Let $x_{ijr}$ be the rebalancing flow of empty vehicles for edge $(i,j)$

- Additional constraints: (3) conservation of rebalancing flow, (4) nonnegative rebalancing flow

\[
\sum_{j \in V_i^+} x_{ijr} - \sum_{j \in V_i} x_{jir} = r_i, \quad \forall i \in V, \quad (3)
\]

for $r_i := \sum_{m \in M} (I\{d_m = i\} - I\{o_m = i\})\lambda_m$

\[
x_{ijr} \geq 0, \quad \forall (i, j) \in E. \quad (4)
\]
Autonomous mobility-on-demand (AMoD)

**Definition (AMoD):** Given $G, OD$, minimize the expression

$$F_E(\hat{x}) = \sum_{(i,j) \in E} \hat{x}_{ij} c_{ij}(\hat{x}_{ij}),$$

subject to (1),(2),(3),(4) where $\hat{x} := \{\hat{x}_{ij} = x_{ij} + x_{ijr} | (i,j) \in E\}$.

- **Remarks / limitations of model:**
  - We assume that mobility requests do not change with time
  - We model vehicles routes as fractional flows
  - We do not account for microscopic traffic phenomena (e.g., traffic lights, shocks)
  - We constrain the capacity of vehicles to 1
Convex programming for TAP

• Our TAP formulation induces a convex program
  • For every \((i, j)\), the derivative of \(x_{ij} c_{ij}(x_{ij})\) strictly increasing
  • \(F_E(x)\) is a sum of convex functions

• We will employ the Frank-Wolfe (FW) [Frank & Wolfe; 1956] for minimizing \(F_E(x)\):
  • Recently has been used extensively in machine learning [Lacoste-Julien & Jaggi; 2015], [Pëna et al.; 2016], *
  • Has been applied to TAP already in 1985

• Important: FW brakes the optimization into a sequence of shortest path queries on a static graph!

FW optimization

In its core, it’s a gradient decent method.

Algorithm 1 Frank Wolfe ($\bar{F}_E, G, OD$)

1: $x^0 \leftarrow$ feasible solution for TAP; $k \leftarrow 0$
2: while stopping criterion not reached do
3: $y^k \leftarrow \text{argmin}_y \bar{F}_E(x^k) + \nabla \bar{F}_E(x^k)^T(y - x^k)$, s.t. $y^k$ satisfies (1), (2)
4: $\alpha_k \leftarrow \text{argmin}_{\alpha \in [0,1]} \bar{F}_E(x^k + \alpha(y^k - x^k))$
5: $x^{k+1} \leftarrow x^k + \alpha_k(y^k - x^k)$; $k \leftarrow k + 1$
6: return $x^k$

* Pseudo-code based on formulation of [Patriksson, 2015]
** Figure by Stephanie Stutz for public domain, labels added by Martin Jaggi, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=35484532
FW, a closer look

Step 3 turns out to be relatively-simple too!

$$F_E(x^k) + \nabla F_E(x^k)^T(y - x^k)$$

- Minimizing this expression $\Leftrightarrow$ minimizing $\nabla F_E(x^k)^T y$

- Next, note that $\frac{\partial}{\partial x_{ij}} F_E(x^k) = c_{ij}(x_{ij}^k)$

- Thus, the optimal $y$ is simply a set of shortest paths (satisfying (1),(2)) with respect to the graph with (static) edges weights $c_{ij}(x_{ij}^k)$!

- This step is called all-or-nothing assignment

---

* Pseudo-code based on formulation of [Patriksson; 2015]
** Figure by Stephanie Stutz for public domain, labels added by Martin Jaggi, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=35484532

---

Kiril Solovey

Scalable and Congestion-aware AMoD Routing
Shortest path queries in transportation

• All-or-nothing assignment consists of multiple independent queries
• Standard techniques (Dijkstra, A*, APSP) are too expensive
• Algorithms tailored for transportation networks:
  • Highway hierarchies [Sanders and Schultz; 2005]
  • Contraction hierarchies [Geisberger et al.; 2012]
  • Customizable contraction hierarchies [Dibblet et al.; 2016]

* See also Bast et al., “Route planning in transportation networks”. In Algorithm Engineering: Selected Results and Surveys, 2016.
Kiril Solovey
Scalable and Congestion-aware AMoD Routing
AMoD as TAP: roadmap adjustments

- We transform AMoD into TAP
- Main challenge is that empty rebalancers don’t have predefined destination vertices
- Given $G, OD$, we generate $G' = (V', E'), OD'$ where
  $V' = V \cup \{n\}$, $E' = E \cup \{(i, n) | i \in V \text{ and } r_i < 0\}$

$(r_i < 0$ indicates that there are fewer request arriving to $i$ than departing$)$

$OD = \{(2,1,2), (1,2,4), (1,3,4), (2,4,1), (2,4,2)\}$
AMoD as TAP: demand adjustments

• We added a dummy vertex \( n \) and dummy edges \((i, n)\)
• To mimic empty rebalancers as user requests we introduce \( OD' \)
  • Includes all the requests from \( OD \)
  • For every \( i \) with \( r_i > 0 \), add the request \((r_i, o, n)\)

\[
\begin{align*}
OD &= \{(2,1,2), (1,2,4), (1,3,4), (2,4,1), (2,4,2)\} \\
OD' &= OD \cup \{(3,2,6)\}
\end{align*}
\]
AMoD as TAP: guaranteeing rebalancing

• We ensure rebalancing (i.e., the correct number of empty vehicles will reach proper vertices) by setting proper values of capacity and free flow over the dummy edge

• For every \((i, n)\), \(c_{in}(x_{in}) = BPR(x_{in}, \kappa_{in}, \phi_{in}):\)
  • \(\kappa_{in} = -r_i\) (amount of rebalancing request for \(i\))
  • \(\phi_{in} = L\), where \(L\) is a large constant (identical for all dummy edges)

\[\begin{align*}
\kappa_{3,6} &= 1 \\
\kappa_{4,6} &= 2
\end{align*}\]

\[\begin{align*}
OD &= \{(2,1,2), (1,2,4), (1,3,4), (2,4,1), (2,4,2)\} \\
OD' &= OD \cup \{(3,2,6)\}
\end{align*}\]
AMoD as TAP: overall approach

• Given $G, OD$, generate $G', OD'$
• Solve (using FW) the new TAP problem
  • I.e., find $x$ minimizing
    \[ F_{E'}(x) := F_E(x) + F_{\bar{E}}(x), \quad \text{where } F_{\bar{E}}(x) := \sum_{(i,n) \in \bar{E}} x_{in} c_{in}(x_{in}) \]
    real cost \quad dummy cost
  • AMoD flows are given by $x$ (after discarding flow on dummy edges)
  • Cost of AMoD solution is given by $F_E(x)$

Question: what can we say about the theoretical guarantees of $x$?
1. How far $F_E(x)$ is from the optimal solution to the original AMoD problem?
2. What is the fraction of fulfilled rebalancing requests?
Theory: rebalancing constraints

• $L$ can be tuned to satisfy any fraction $< 1$ of requests

• First, motivation for setting $\kappa_{in}, \phi_{in}$
  • Let $R$ the total number of empty rebalancers.

Lemma (optimal assignment for dummy edges): Let $x^* = \arg\min F_{\bar{E}}(x)$ such that $\sum_{(i,n) \in \bar{E}} x_{in}^* = R$. Then

$$x_{\bar{E}}^* = \kappa,$$

where $\kappa := \{\kappa_{in} | (i, n) \in \bar{E}\},$

and $x_{\bar{E}}^*$ is the restriction of $x^*$ to $\bar{E}$.

Proof. Using Lagrange multipliers and relying on the structure of the BPR function.

Kiril Solovey
Scalable and Congestion-aware AMoD Routing
23
Theory: rebalancing constraints

For a given solution $\mathbf{x}$, the expression

$$\frac{||x^*_E - \kappa||_1}{2R}$$

represents fraction of unfulfilled requests.

**Theorem (bounded fraction of unfulfilled requests):**

Let $x^* = \text{argmin } F_{E'}(x)$ subject to constraints (1), (2). Then for every $\delta \in (0,1]$, there exist $L_\delta \in (0, \infty)$ such that if $L > L_\delta$ then $\frac{||x^*_E - \kappa||_1}{2R} \leq \delta$. 
Theory: rebalancing constraints

**Theorem (bounded fraction of unfulfilled requests):** Let \( x^* = \text{argmin} F_{E'}(x) \) subject to constraints (1),(2). Then for every \( \delta \in (0,1] \), there exist \( L_\delta \in (0, \infty) \) such that if \( L > L_\delta \) then \( \frac{\|x^*_\delta - \kappa\|_1}{2R} \leq \delta \).

**Proof.**

- Let \( x^0 \) be an assignment (satisfying (1), (2)) such that
  - \( x^0_E = \kappa \) and \( F_{E'}(x^0) \) is minimized
  - \( x^0 \) corresponds to the optimal solution of the original AMoD problem with full rebalancing
- Fix \( \delta \in (0,1] \) and let \( x^\delta \) be any assignment with \( \|x^\delta_E - \kappa\|_1 / 2R > \delta \)
- We wish to find \( L_\delta \) s.t. for any \( L > L_\delta \) it holds that \( F_{E'}(x^\delta) > F_{E'}(x^0) \)
- This implies that \( x^* = \text{argmin} F_{E'}(x) \) must satisfy at least \( 1 - \delta \) of rebalancing request
Theory: rebalancing constraints

Proof cont.

• By Lemma,
  \[ F_{E'}(x^\delta) > F_{E'}(x^0) \Rightarrow F_{\bar{E}}(x^\delta) - F_{\bar{E}}(x^0) > F_E(x^0) - F_E(x^\delta) \]

• Bounding the RHS is tricky...

• Instead, we will find \( L \) such that \( F_{\bar{E}}(x^\delta) - F_{\bar{E}}(x^0) > F_E(x^0) \)

• We can find (practical) upper bounds for \( F_E(x^0) \)

• Lower-bounding the LHS is simpler since edges of \( \bar{E} \) are not interconnected

• We obtain that \( F_{\bar{E}}(x^\delta) \geq F_{\bar{E}}(x^0) + \text{const} \cdot L \delta^2 \)

• Thus, when \( L = \frac{F_E(x^0)}{\delta^2} \text{const} \) we get the desired result \( \blacksquare \)
Theory: cost of routing scheme

- What can we say about the cost of the actual routing scheme?
  - I.e., what is the relation between $F_E(x^*)$ and $F_E(x^0)$?

**Corollary (bounded cost of routing):** Fix $\delta \in (0,1]$ and choose $L > L_\delta$. Then (i) \(\frac{||x_E^*-\kappa||}{2R} \leq \delta\), and (ii) $F_E(x^*) < F_E(x^0)$, where $x^* = \text{argmin } F_{E'}(x)$ subject to constraints (1),(2).

**Proof.**
- By definition of $x^*$: $F_{E'}(x^*) < F_{E'}(x^0)$
- By Lemma: $F_{\bar{E}}(x^0) < F_{\bar{E}}(x^*)$, thus:

\[
F_{E'}(x^*) = F_E(x^*) + F_{\bar{E}}(x^*) < F_{E'}(x^0) \\
= F_E(x^0) + F_{\bar{E}}(x^0) < F_E(x^0) + F_{\bar{E}}(x^*) \\
\Rightarrow F_E(x^*) < F_E(x^0) \quad \blacksquare
\]
Experiments

• We validate experimentally our theoretical results
• Approach yields near-optimal solutions within seconds, where most requests (99%) are fulfilled, for realistic scenarios (e.g., 150K users)
• Our bounds for $L$ are quite conservative and already small values yield desired outcome in practice
• Running times and convergence rates scale linearly with input size
• Our approach is several orders faster than previous work
Implementation details

• Results obtained over commodity laptop
  • 2.80Ghz * 4 core i7-7600U CPU, 16GB RAM

• C++ Frank-Wolfe implementation was adapted (minor changes) from the routing-framework

• Shortest-path queries performed using RoutingKit, and other speed ups described in [Buchold et al.; 2018]
Data

• Our data is based on [Salazar et al.; 2018]

• Roadmap $G$ is for Manhattan, NY
  • $|V| = 1352, |E| = 3338$
  • Extracted from Open Street Map

• OD-pairs inferred from taxi rides (2012)
  • Morning peak hour
  • We scaled the requests by a factor of 6, to simulate all ride-hailing vehicles
  • $6 \times 25,960$ requests in total

• To simulate exogenous traffic, we increase the cost from $c_{ij}(x_{ij})$ to $c_{ij}(x_{ij} + x_{ij}^e)$
  • We choose a global value $\gamma_{exo}$ and assign $\frac{x_{ij}^e}{\kappa_{ij}} = \gamma_{exo}$
  • Unless stated otherwise $\gamma_{exo} = 0.8$
Validation of theory

- We study how the value of $L$ affects
  - Fraction of unfulfilled requests $\delta$
  - Real cost $C_r := F_E$
  - Dummy cost $C_d := F_{\bar{E}}$

- $L = 3, 6, 12, 24, 48, 96, 192$ minutes

- Plot results for 100 iterations (15sec total)
  - OPT was obtained for $L = 96$ after 10,000 iter.
  - Relative difference for last iteration of OPT was around $2 \cdot 10^{-7}$

- Conclusions:
  - Values plateau within 100 iterations or earlier
  - Relatively small $L$, e.g., 96, already yields desired results
  - Desired $L$ can be found using binary search
Scalability

- We demonstrate the scalability of approach
  - Fix $L = 96$ minutes
  - Scale the original OD set by 1,2,3,4
  - Adjust $\gamma_{exo}$ to 0.8,0.6,0.4,0.2
  - 1000 iterations in total

- Values in brackets:
  - $\delta$ at 100 iterations
  - $C_r$ at 100 iterations
  - running time at 100 iterations

- Conclusion:
  - Running time and convergence rates scale linearly with input size
  - We mention that real data will behave slightly differently due to larger variance
Comparison with previous work

- We compare the cost of the obtained solution of
  - Our approach (full BPR)
  - Piecewise-linear approximation of BPR [Salazar et al.; 2018]
  - Congestion-unaware approach [Pavone et al.; 2011]
- Parameters: $L = 96$, $\gamma_{exo} \in [0.0, 2.0]$
Conclusion and future work

• We presented an efficient approach for congestion-aware AMoD routing

• Although full rebalancing is not guaranteed, solution can be easily fixed to provide it

• Real-time implementation:
  • Receding-horizon optimization
  • Recover integer flows via randomized rounding [Rossi, 2018]
  • Microscopic simulations with AMoDeus+MATSim or SimMobility

• More speedup:
  • Use customizable contraction hierarchies
  • Bundle identical OD pairs
  • Multi-core implementation (algorithm is embarrassingly parallel!)
Conclusion and future work

• Theory:
  • Better estimation for $L_\delta$
  • Convergence rate analysis [Lacoste-Julien and Jaggi, 2015]

• Extensions
  • Ridesharing
  • Dynamic (time-variant) version
    • Dynamic version of Frank-Wolfe is needed!
  • Incorporate more realistic flow properties