Dynamic Vehicle Routing for Robotic Networks
Lecture #1: Introduction

Francesco Bullo1 Emilio Frazzoli2 Marco Pavone2
Ketan Savla2 Stephen L. Smith2

1CCDC
University of California, Santa Barbara
bullo@engineering.ucsb.edu

2LIDS and CSAIL
Massachusetts Institute of Technology
{frazzoli,pavone,ksavla,slsmith}@mit.edu

Workshop at the 2010 American Control Conference
Baltimore, Maryland, USA, June 29, 2010, 8:30am to 5:00pm

Acknowledgements

Funded in part by
AFOSR grant no. FA 8650-07-2-3744 (Michigan/AFRL Collaborative Center on Control Sciences), ARO MURI “Swarms” W911NF-05-1-0219, ARO award DAAD19-03-D-0004 (Institute for Collaborative Biotechnologies), NSF awards #0705451 and #0705453, ONR award N00014-07-1-0721.

Collaborators
Alessandro Arsie (U. of Toledo), Shaunak D. Bopardikar (UCSB), Ruggero Carli (UCSB/Padova), Jorge Cortés (UCSD), Joey W. Durham (UCSB), John J. Enright (Kiva Systems), Paolo Frasca (Roma), Anurag Ganguli (UtopiaCompression), João P. Hespanha (UCSB) Sonia Martínez (UCSD), Karl Obermeyer (UCSB), and Sara Susca (Honeywell).

Lecture outline

1 Acknowledgements

2 Autonomy and Networking Technologies

3 Prototypical DVR problem

4 Literature review

5 Contributions

6 Comparison with alternative approaches
   • Re-optimization
   • Online algorithms

7 Workshop Structure and Schedule

Autonomy and Networking Technologies

Individual members in the group can
• sense its immediate environment
• communicate with others
• process the information gathered
• take a local action in response
Given:
- a group of vehicles, and
- a set of service demands

Objective:
provide service in minimum time
service = take a picture at location
**Prototypical Dynamic Vehicle Routing Problem**

**Given:**
- a group of vehicles, and
- a set of service demands

**Objective:**
provide service in minimum time
service = take a picture at location

**Vehicle routing**  
(All info known ahead of time, Dantzig '59)
Determine a set of paths that allow vehicles to service the demands

**Dynamic vehicle routing**  
(New info in real time, Psaraftis '88)
- New demands arise in real-time
- Existing demands evolve over time

---

**Lecture outline**

1. Acknowledgements
2. Autonomy and Networking Technologies
3. Prototypical DVR problem
4. Literature review
5. Contributions
6. Comparison with alternative approaches
   - Re-optimization
   - Online algorithms
7. Workshop Structure and Schedule

---

**Light and heavy load regimes**

**From coordination and static routing to Dynamic Vehicle Routing**

**Simple coordination problems arise in static environments**

- motion coordination: rendezvous, deployment, flocking
- task allocation, target assignment
- static vehicle routing  
  (P. Toth and D. Vigo '01)

**Routing policies vs planning algorithms**

dynamic, stochastic and adversarial events take place

- design policies (in contrast to pre-planned routes or motion planning algorithms) to specify how to react to events
- dynamic demands add queueing phenomena to the combinatorial nature of vehicle routing
Literature on DVR and queueing for robotic networks

- Shortest path through randomly-generated and worst-case points (Beardwood, Halton and Hammersly, 1959 — Steele, 1990)
- Traveling salesman problem solvers (Lin, Kernighan, 1973)
- DVR formulation on a graph (Psaraftis, 1988)
- DVR on Euclidean plane (Bertsimas and Van Ryzin, 1990–1993)
- Unified receding-horizon policy (Papastavrou, 1996)

Recent developments in DVR for robotic networks:

- Adaptation and decentralization
- Vehicles with dynamics, nonholonomic vehicles, Dubins UAVs
- Pickup & delivery tasks
- Heterogeneous vehicles and team forming
- Distinct-priority and impatient demands
- Moving demands

Contributions of our recent works

Comprehensive framework for DVR in robotic systems

- adaptive DVR policies for single vehicles in light and heavy load
- cooperative DVR policies via partitioning
- scalable distributed partitioning policies under a variety of communication/interaction scenarios
- (models, algorithms and analysis of) service vehicles with dynamics & stochastic and combinatorics of nonholonomic Dubins vehicles performing Traveling Salesman Problems and DVR tasks
- (models, algorithms and analysis of) service vehicles with time constraints and heterogeneous priorities
- (models, algorithms and analysis of) demands requiring service by multiple heterogeneous vehicles simultaneously.

Lecture outline

1. Acknowledgements
2. Autonomy and Networking Technologies
3. Prototypical DVR problem
4. Literature review
5. Contributions
6. Comparison with alternative approaches
   - Re-optimization
   - Online algorithms
7. Workshop Structure and Schedule

Bibliography on DVR and queueing for robotic networks

Plain-vanilla re-optimization?

Example: DVR on segment
- Objective: minimize average waiting time
- Strategy: re-optimize at each event

- For adversarial target generation, vehicle travels forever without ever servicing any request \(\Rightarrow\) unstable queue of outstanding requests
- Even if queue remains bounded, what about performance? how far from the optimal?
**Plain-vanilla re-optimization?**

**Example: DVR on segment**
- Objective: minimize average waiting time
- Strategy: re-optimize at each event

- For adversarial target generation, vehicle travels forever without ever servicing any request \(\implies\) unstable queue of outstanding requests
- Even if queue remains bounded, what about performance? how far from the optimal?
Online algorithms?

Online algorithms

- Online algorithm operates based on input information up to the current time
- Online algorithm is (worst-case) $r$-competitive if
  \[ \text{Cost}_{\text{online}}(I) \leq r \text{Cost}_{\text{optimal offline}}(I), \quad \forall \text{ problem instances } I. \]

Disadvantages

- Cumulative cost
- Worst-case analysis
- Not possible to include a-priori information (e.g., arrival rate)
- Not as clear what competitive ratio means
- So far, only few simple DVR problems admit online algorithms

Lecture outline

1. Acknowledgements
2. Autonomy and Networking Technologies
3. Prototypical DVR problem
4. Literature review
5. Contributions
6. Comparison with alternative approaches
   - Re-optimization
   - Online algorithms
7. Workshop Structure and Schedule

Workshop Structure and Schedule

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:00-8:30am</td>
<td>Coffee Break</td>
<td>Intro to dynamic vehicle routing</td>
</tr>
<tr>
<td>8:30-9:00am</td>
<td>Lecture #1:</td>
<td>Prelims: graphs, TSPs and queues</td>
</tr>
<tr>
<td>9:05-9:50am</td>
<td>Lecture #2:</td>
<td>The single-vehicle DVR problem</td>
</tr>
<tr>
<td>9:55-10:40am</td>
<td>Lecture #3:</td>
<td>The multi-vehicle DVR problem</td>
</tr>
<tr>
<td>10:40-11:00am</td>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>11:00-11:45pm</td>
<td>Lecture #4:</td>
<td>Extensions to vehicle networks</td>
</tr>
<tr>
<td>11:45-1:10pm</td>
<td>Lunch Break</td>
<td>Extensions to different demand models</td>
</tr>
<tr>
<td>1:10-2:10pm</td>
<td>Lecture #5:</td>
<td></td>
</tr>
<tr>
<td>2:15-3:00pm</td>
<td>Lecture #6:</td>
<td>Extensions to different vehicle models</td>
</tr>
<tr>
<td>3:00-3:20pm</td>
<td>Coffee Break</td>
<td>Extensions to different task models</td>
</tr>
<tr>
<td>3:20-4:20pm</td>
<td>Lecture #7:</td>
<td></td>
</tr>
<tr>
<td>4:25-4:40pm</td>
<td>Lecture #8:</td>
<td>Final open-floor discussion</td>
</tr>
<tr>
<td>4:45-5:00pm</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dynamic Vehicle Routing for Robotic Networks

Lecture #2: Preliminary Results in Combinatorics

Francesco Bullo\textsuperscript{1} Emilio Frazzoli\textsuperscript{2} Marco Pavone\textsuperscript{2} Ketan Savla\textsuperscript{2} Stephen L. Smith\textsuperscript{2}

\textsuperscript{1}CCDC
University of California, Santa Barbara
bullo@engineering.ucsb.edu

\textsuperscript{2}LIDS and CSAIL
Massachusetts Institute of Technology
\{frazzoli,pavone,ksavla,slsmith\}@mit.edu

Workshop at the 2010 American Control Conference
Baltimore, Maryland, USA, June 29, 2010, 8:30am to 5:00pm

---

Graph Theory
- Weighted Graphs
- Minimum Spanning Tree

The Traveling Salesman Problem
- Approximation Algorithms
- Metric TSP
- Euclidean TSP

Queueing Theory
- Kendall’s Notation
- Little’s Law and Load Factor

---

Key references for this lecture

**Graph Theory Basics:**

**Combinatorial Optimization:**

**Stochastic TSP:**

**Basic Queueing Theory:**
An undirected graph $G = (V, E)$.
- A path in $G$ is a sequence $v_1, e_1, v_2, \ldots, v_k, e_k, v_{k+1}$, with
  - $e_i \neq e_j$ for $i \neq j$.
  - $v_i \neq v_j$ for all $i \neq j$.
- A circuit or cycle has $v_1 = v_{k+1}$.
- A Hamiltonian path is a path that contains all vertices.
- Similarly define a Hamiltonian cycle or tour.
Graph Theory Review

- An undirected graph \( G = (V, E) \).
- A path in \( G \) is a sequence \( v_1, e_1, v_2, \ldots, v_k, e_k, v_{k+1} \), with
  - \( e_i \neq e_j \) for \( i \neq j \).
  - \( v_i \neq v_j \) for all \( i \neq j \).
- A circuit or cycle has \( v_1 = v_{k+1} \).
- A Hamiltonian path is a path that contains all vertices.
- Similarly define a Hamiltonian cycle or tour.

Weighted Graphs

- A weighted graph \( G = (V, E, c) \) has edge weights \( c : E \to \mathbb{R}_{>0} \).
- In a complete graph, \( E = V \times V \).

Special classes of complete weighted graphs:
- Metric if
  \[
  c(\{v_1, v_2\}) + c(\{v_2, v_3\}) \geq c(\{v_1, v_3\}) \quad \text{for all } v_1, v_2, v_3 \in V.
  \]
- Euclidean if
  \[
  V \subset \mathbb{R}^d \quad \text{and} \quad c(\{v_i, v_j\}) = \|v_i - v_j\|_2.
  \]

Minimum Spanning Tree

- A tree is a graph with no cycles
- A spanning tree of \( G \) is a subgraph that
  - is a tree
  - connects all vertices together

Minimum Spanning Tree Problem

Given: a weighted graph \( G = (V, E, c) \)
Task: find a spanning tree \( T = (E_T, V_T) \) such that \( \sum_{e \in E_T} c(e) \) is minimum.

Can be solved in greedy fashion using Kruskal’s algorithm:
- Recursively adds shortest edge that does not create a cycle
- Runs in \( O(n^2) \) time \( \left( \text{where } |V| = n \right) \)

Minimum Spanning Tree Problem

Given: a weighted graph \( G = (V, E, c) \)
Task: find a spanning tree \( T = (E_T, V_T) \) such that \( \sum_{e \in E_T} c(e) \) is minimum.

Can be solved in greedy fashion using Kruskal’s algorithm:
- Recursively adds shortest edge that does not create a cycle
- Runs in \( O(n^2) \) time \( \left( \text{where } |V| = n \right) \)

Minimum Spanning Tree Problem

Given: a weighted graph \( G = (V, E, c) \)
Task: find a spanning tree \( T = (E_T, V_T) \) such that \( \sum_{e \in E_T} c(e) \) is minimum.

Can be solved in greedy fashion using Kruskal’s algorithm:
- Recursively adds shortest edge that does not create a cycle
- Runs in \( O(n^2) \) time \( \left( \text{where } |V| = n \right) \)
Hamiltonian Cycle Decision Problem

Hamiltonian Cycle

Given: An undirected graph $G$.

Question: Does $G$ contain a Hamiltonian cycle?

Hamiltonian Cycle is $\text{NP-complete}$

(One of Karp’s 21 NP-complete problems)

Recall, a problem is $\text{NP-complete}$ if

- Every solution can be verified in polynomial time ($\text{NP}$).
- Every problem in $\text{NP}$ can be reduced to it.

Traveling Salesman Problem

Traveling Salesman Problem (TSP)

Given: A complete graph $G_n = (V_n, E_n)$ and weights $c : E_n \to \mathbb{R}_{>0}$.

Task: Find a Hamiltonian cycle with minimum weight.

- TSP is $\text{NP-hard}$
- To show $\text{NP-hard}$: Reduce Hamiltonian Cycle to TSP.

Given an undirected graph $G = (V, E)$ with $|V| = n$:

1. Construct complete graph $G_n$ with weight 1 for each edge in $E$ and weight 2 for all other edges.
2. Then $G$ is Hamiltonian $\iff$ optimum TSP tour has length $n$.

Outline

1. Graph Theory
2. The Traveling Salesman Problem
   - Approximation Algorithms
   - Metric TSP
   - Euclidean TSP
3. Queueing Theory

Approximation Algorithms for the TSP

Theorem (Sahni and Gonzalez, 1976)

Unless $P = NP$, there is no $k$-factor approx alg for the TSP for any $k \geq 1$.

Proof Idea: $k$-factor approx would imply poly time algorithm for Hamiltonian Cycle.

In practice for metric and non-metric problems:

- Heuristic: Lin-Kernighan based solvers (Lin and Kernighan, 1973)
  - Empirically ~ 5% of optimal in $O(n^{2.2})$ time.
- Exact: Concorde TSP Solver (Applegate, Bixby, Chvatal, Cook, 2007)
  - Exact solution of Euclidean TSP on 85,900 points!
Metric TSP

Given: A complete metric graph $G_n = (V_n, E_n)$
Task: Find a Hamiltonian cycle with minimum weight.

- The Metric TSP is NP-hard.
- There exist approximation algorithms!

Eulerian Graphs

- **Eulerian graph:** degree of each vertex is even
- **Eulerian walk:** Closed walk containing every edge.
- Graph has Eulerian walk $\iff$ Eulerian.
- Eulerian walk can be computed in $O(|V| + |E|)$ time.

Double-Tree Algorithm

1: Find a minimum spanning tree $T$ of graph $G_n$.
2: $\overline{G} :=$ graph containing two copies of each edge in $T$.
3: Compute Eulerian walk in Eulerian graph $\overline{G}$.
4: Walk gives ordering, ignore all but first occurrence of vertex.
Double-Tree Algorithm

1: Find a minimum spanning tree $T$ of graph $G$.
2: $G'$ := graph containing two copies of each edge in $T$.
3: Compute Eulerian walk in Eulerian graph $G'$.
4: Walk gives ordering, ignore all but first occurrence of vertex.

Theorem

Double-Tree Algorithm is a 2-approx algorithm for the Metric TSP. Its running time is $O(n^2)$.

- Deleting one edge from a tour gives a spanning tree.
- Thus minimum spanning tree is shorter than optimal tour.
- Each edge is doubled.
- Spanning tree can be computed in $O(n^2)$ time.
- Eulerian walk computed in $O(n)$ time.

Christofides’ Algorithm

1: Find a minimum spanning tree $T$ of $G$.
2: Let $W$ be the set of vertices with odd degree in $T$.
3: Find the minimum weight perfect matching $M$ in subgraph generated by $W$.
4: Find an Eulerian path in $G' := (V_n, E(T) \cup M)$, (skip vertices already seen).

Christofides’ Algorithm

1: Find a minimum spanning tree $T$ of $G$.
2: Let $W$ be the set of vertices with odd degree in $T$.
3: Find the minimum weight perfect matching $M$ in subgraph generated by $W$.
4: Find an Eulerian path in $G' := (V_n, E(T) \cup M)$, (skip vertices already seen).
Christofides’ Algorithm

1: Find a minimum spanning tree \( T \) of \( G \).
2: Let \( W \) be the set of vertices with odd degree in \( T \).
3: Find the minimum weight perfect matching \( M \) in subgraph generated by \( W \).
4: Find an Eulerian path in \( G := (V_n, E(T) \cup M) \), (skip vertices already seen).

Christofides’ Algorithm

1: Find a minimum spanning tree \( T \) of \( G \).
2: Let \( W \) be the set of vertices with odd degree in \( T \).
3: Find the minimum weight perfect matching \( M \) in subgraph generated by \( W \).
4: Find an Eulerian path in \( G := (V_n, E(T) \cup M) \), (skip vertices already seen).
### Christofides' Algorithm

**Theorem**

Christofides’ Algorithm gives a 3/2-approx algorithm for the Metric TSP. Its running time is \( O(n^3) \).

- \( L(\text{Christofides}) = L(\text{MST}) + L(M) \).
- But, \( L(\text{MST}) < L(\text{TSP}) \), and
- \( L(M) \leq L(M') \leq L(\text{TSP})/2 \).
  
  Where \( M' \) is the minimum perfect matching of \( W \) using edges that are part of TSP.

**Best known approx algorithm for Metric TSP**

### Euclidean TSP

**Theorem** (Arora, 1998; Mitchell, 1999)

For each fixed \( \epsilon > 0 \), a \((1 + \epsilon)\)-approximate solution can be found in \( O(n^3(\log n)^c) \) time.

Practical value limited to due \( c \)'s dependence on \( \epsilon \).

### Length Bounds for Euclidean TSP

How long is the TSP tour through \( n \) points in unit square?

**Theorem** (few, 1955)

For every set \( Q_n \) of \( n \) points in the unit square

\[
\text{ETSP}(Q_n) \leq \sqrt{2n} + 7/4.
\]

Worst-case lower bound matches:

- Equally space \( n \) points on a grid
  
  Then \( \text{ETSP}(Q_n) = C\sqrt{n} \).

- So, worst-case length \( \geq C\sqrt{n} \).

### Worst-case TSP Length Upper Bound (Intuition)

- Consider \( Q_n := \{x_1, \ldots, x_n\} \) of \( n \) points in unit square.
- There exists \( c > 0 \) such that

\[
\min \left\{ \|x_i - x_j\| : x_i, x_j \in Q_n \right\} \leq \frac{c}{\sqrt{n}}.
\]

- Let \( \ell_n \) denote worst-case TSP length through \( n \) pts.
- Then \( \ell_n \leq \ell_{n-1} + 2c/\sqrt{n} \).
- Summing we get \( \ell(n) \leq C\sqrt{n} \).
Worst-case TSP Length Upper Bound (Intuition)

- Consider $Q_n := \{x_1, \ldots, x_n\}$ of $n$ points in unit square.
- There exists $c > 0$ such that
  $$\min \left\{ \|x_i - x_j\| : x_i, x_j \in Q_n \right\} \leq \frac{c}{\sqrt{n}}.$$
- Let $\ell_n$ denote worst-case TSP length through $n$ pts.
- Then $\ell_n \leq \ell_{n-1} + 2c/\sqrt{n}$.
- Summing we get $\ell(n) \leq C\sqrt{n}$.

---

Worst-case TSP Length Upper Bound (Intuition)

- Consider $Q_n := \{x_1, \ldots, x_n\}$ of $n$ points in unit square.
- There exists $c > 0$ such that
  $$\min \left\{ \|x_i - x_j\| : x_i, x_j \in Q_n \right\} \leq \frac{c}{\sqrt{n}}.$$
- Let $\ell_n$ denote worst-case TSP length through $n$ pts.
- Then $\ell_n \leq \ell_{n-1} + 2c/\sqrt{n}$.
- Summing we get $\ell(n) \leq C\sqrt{n}$. 
**TSP Length for Random Points**

*Theorem (Beardwood, Halton, and Hammersley, 1959)*

Let $Q_n$ be a set of $n$ i.i.d. random variables with compact support in $\mathbb{R}^d$ and distribution $\varphi(x)$. Then, with prob. 1

$$\lim_{n \to +\infty} \frac{\text{ETSP}(Q_n)}{n^{(d-1)/d}} = \beta_{\text{TSP},d} \int_{\mathbb{R}^d} \tilde{\varphi}(x) (d-1)/d \, dx,$$

where $\beta_{\text{TSP},d}$ is a constant independent of $\varphi$, and $\tilde{\varphi}$ is absolutely continuous part of $\varphi$.

For uniform distribution in square of area $A$

$$\frac{\text{ETSP}(Q_n)}{\sqrt{n}} \to \beta_{\text{TSP},2} \sqrt{A} \quad \text{as} \ n \to +\infty.$$

Best estimate of $\beta_{\text{TSP},2}$ is Percus and Martin, 1996

$$\beta_{\text{TSP},2} \approx 0.7120.$$

---

**Summary of Traveling Salesman Problem**

- Solving TSP is *NP-hard*, and no approx algorithms exist.
- For *metric TSP*, still *NP-hard* but good *approx algs* exist.
- For Euclidean TSP, very good heuristics exist.
- Length of tour through $n$ points in unit square:
  - Worst-case is $\Theta(\sqrt{n})$.
  - Uniform random is $\Theta(\sqrt{n})$.
  - For all density functions $O(\sqrt{n})$.

---

**Outline**

1. Graph Theory
2. The Traveling Salesman Problem
3. Queueing Theory
   - Kendall’s Notation
   - Little’s Law and Load Factor

---

**Basic Queueing Model**

- Customers arrive, wait in a queue, and are then processed
- Queue length builds up when arrival rate is larger than service rate

![Queueing Model Diagram](image)

- **Arrivals** modeled as stochastic process with rate $\lambda$
- **Service time** of each customer is a r.v. with finite mean $\bar{s}$ and second moment $\bar{s}^2$.
- **Service rate** is $1/\bar{s}$.
Queueing Notation

Kendall’s Queueing notation $A/B/C$:
- $A =$ the arrival process
- $B =$ the service time distribution
- $C =$ the number of servers

Main codes:
- $D =$ Deterministic
- $M =$ Markovian
  - for arrivals: Poisson process
  - for service times: Exponential distribution
- $G$ (or $GI$) = General distribution (independent among customers)

Example $M/G/m$ queue:
- Poisson arrivals with rate $\lambda$
- General service times with mean $\bar{s}$
- $m$ servers

Little’s Law and Load Factor

Define:
- average wait-time in queue as $\bar{W}$
- average system as $\bar{T} := \bar{W} + \bar{s}$.

Little’s Law / Theorem

For a stable queue $\bar{N} = \lambda \bar{W}$

- For $m$ servers, define load factor as $\rho := \frac{\lambda \bar{s}}{m}$
- Necessary condition for stable queue is $\rho < 1$.

Wait-time examples

For $M/D/1$ queue:
$$\bar{W} = \frac{\rho \bar{s}}{2(1 - \rho)}$$

For $M/G/1$ queue:
$$\bar{W} = \frac{\lambda \bar{s}^2}{2(1 - \rho)}$$

For $G/G/1$ queue (Kingman, 1962):
$$\bar{W} \leq \frac{\lambda (\sigma_s^2 + \sigma_z^2)}{2(1 - \rho)}$$
and the upper bound becomes exact as $\rho \to 1^-$.

Lecture outline

1. Graph Theory
   - Weighted Graphs
   - Minimum Spanning Tree

2. The Traveling Salesman Problem
   - Approximation Algorithms
   - Metric TSP
   - Euclidean TSP

3. Queueing Theory
   - Kendall’s Notation
   - Little’s Law and Load Factor
Dynamic Vehicle Routing for Robotic Networks
Lecture #3: The single-vehicle DVR problem

Francesco Bullo\textsuperscript{1} Emilio Frazzoli\textsuperscript{2} Marco Pavone\textsuperscript{2}
Ketan Savla\textsuperscript{2} Stephen L. Smith\textsuperscript{2}

\textsuperscript{1}CCDC
University of California, Santa Barbara
bullo@engineering.ucsb.edu

\textsuperscript{2}LIDS and CSAIL
Massachusetts Institute of Technology
\{frazzoli,pavone,ksavla,slsmith\}@mit.edu

Workshop at the 2010 American Control Conference
Baltimore, Maryland, USA, June 29, 2010, 8:30am to 5:00pm

Lecture outline

1. Queueing-theoretical model for DVR
2. Lower bounds on performance (m=1)
3. Control policies


The problem

DVR - distinct features

- service demands vary over time
- information about future is stochastic
- real-time routing policies
- queueing phenomena

DVR is fundamentally a queueing problem:

- arrival process
- service model
- performance measure

General queueing-theoretical model for DVR 1/2

Arrival process: spatio-temporal Poisson

- time intensity $\lambda > 0$
- spatial density $\varphi$: $\mathbb{P}[\text{demand in } S] = \int_S \varphi(x) \, dx$
- inter-arrival times and locations are i.i.d.

Service model:

- $m$ holonomic vehicles with maximum velocity $v$
- vehicles provide a random on-site service
- on-site service times are i.i.d. (equal on average to $\bar{s}$)
- demand removed from the system upon on-site service completion
General queueing-theoretical model for DVR 1/2

Arrival process: spatio-temporal Poisson
- time intensity $\lambda > 0$
- spatial density $\varphi$: $P[\text{demand in } S] = \int_S \varphi(x) \, dx$
- inter-arrival times and locations are i.i.d.

Service model:
- $m$ holonomic vehicles with maximum velocity $v$
- vehicles provide a random on-site service
- on-site service times are i.i.d. (equal on average to $\bar{s}$)
- demand removed from the system upon on-site service completion

Relation to standard queueing systems
- DVR model close to $M/G/m$ queue
- key difference: service times are not i.i.d. in general

Service time correlations in DVR:
- service time = travel time + on-site service
- FCFS policy
- unconditional expected travel time between two consecutive demands $\approx 0.52$.
- conditional expected travel time between two consecutive demands $> 0.52$.

M/G/m methodology is not applicable!

General queueing-theoretical model for DVR 2/2

Performance measure: steady-state system time of demands $\bar{T}$

Problem statement
Solve optimization problem over all causal routing policies $\pi$:

$$\inf_{\pi} \bar{T}_\pi$$

Relation to standard queueing systems
- DVR model close to $M/G/m$ queue
- key difference: service times are not i.i.d. in general

Service time correlations in DVR:
- service time = travel time + on-site service
- FCFS policy
- unconditional expected travel time between two consecutive demands $\approx 0.52$.
- conditional expected travel time between two consecutive demands $> 0.52$.

M/G/m methodology is not applicable!
A first look at the problem: stability

- \( \lambda \cdot E[\text{service time}]/m \) fraction of time each vehicle is busy

Necessary condition for stability:
System is stable if \( \lambda \cdot E[\text{service time}]/m < 1 \).
Since \( \bar{s} \leq E[\text{service time}] \), a weaker necessary condition is:
\[
\rho = \frac{\lambda \bar{s}}{m} < 1
\]

Sufficient condition for stability:
Surprisingly, \( \rho < 1 \) is also sufficient for stability \( \implies \) stability condition is independent of the size and shape of \( Q \).

Analysis approach

- Lack of i.i.d. property substantially complicates analysis
- General approach:
  - lower bounds on performance, independent of algorithms,
  - design of algorithms and upper bound on their performance, possibly in asymptotic regimes (i.e., \( \rho \to 0^+ \) and \( \rho \to 1^- \))

Lecture outline

1. Queueing-theoretical model for DVR
2. Lower bounds on performance (\( m=1 \))
3. Control policies

Light-load lower bound

Median
- minimizer \( p^* \) of
\[
p \mapsto \int_Q \|x - p\| \varphi(x) dx = E_\varphi[\|X - p\|]
\]
- best a priori location to reach next demand

Lower bound (most useful when \( \lambda \to 0^+ \))

For all policies \( \pi \):
\[
\mathbb{T}_\pi \geq E_\varphi[\|X - p^*\|]/\nu + \bar{s}
\]

Proof sketch:
- \( \mathbb{T} = \mathbb{W}_{\text{travel}} + \mathbb{W}_{\text{on-site}} + \bar{s} \)
- \( \mathbb{W}_{\text{travel}} \geq E_\varphi[\|X - p^*\|]/\nu \)
**Light-load lower bound**

- **Median**
  - minimizer $p^*$ of
    \[ p \mapsto \int_Q \|x - p\| \phi(x) dx = \mathbb{E}_\phi[\|X - p\|] \]
  - best a priori location to reach next demand

- **Lower bound (most useful when $\lambda \to 0^+$)**
  For all policies $\pi$: $T_\pi \geq \mathbb{E}_\phi[\|X - p^*\|]/v + \bar{z}$

**Proof sketch:**

- $T = W_{\text{travel}} + W_{\text{on-site}} + \bar{z}$.
- $W_{\text{travel}} \geq \mathbb{E}_\phi[\|X - p^*\|]/v$
Heavy-load lower bound

**Definition (Spatially-biased and -unbiased policies)**

A policy $\pi$ is said to be

- **spatially unbiased** if system time is independent of demand location
- **spatially biased** if system time depends on demand location

**Heavy-load lower bound**

Spatially-unbiased policies:

$$T_\pi \geq \frac{\beta^2_{TSP}}{2} \frac{\lambda \left( \int_Q \varphi^{1/2}(x) dx \right)^2}{\nu^2 (1 - \varrho)^2} \quad \text{as } \varrho \to 1^-$$

Spatially-biased policies:

$$T_\pi \geq \frac{\beta^2_{TSP}}{2} \frac{\lambda \left( \int_Q \varphi^{2/3}(x) dx \right)^3}{\nu^2 (1 - \varrho)^2} \quad \text{as } \varrho \to 1^-$$

**Proof sketch (for unbiased policies)**

Proof of the lower bound:

- the idea is to use stability arguments (which are independent of policies!)
- let $D$ be the travel inter-demand distance
- one can show that
  $$D \geq \beta_{TSP} \frac{\int_Q \varphi^{1/2}(x) dx}{\nu \sqrt{2 N}} \quad \text{as } \varrho \to 1^-,$$

  with $N$ average number of waiting demands
- for stability:
  $$\bar{s} + \frac{D}{\nu} \leq \frac{1}{\lambda} \quad \Rightarrow \quad \bar{s} + \beta_{TSP} \frac{\int_Q \varphi^{1/2}(x) dx}{\nu \sqrt{2 N}} \leq 1/\lambda$$
- since $N = \lambda W$ and $\bar{T} = W + \bar{s}$ one obtains:
  $$\bar{T}^* \geq \frac{\beta^2_{TSP}}{2} \frac{\lambda \left( \int_Q \varphi^{1/2}(x) dx \right)^2}{\nu^2 (1 - \varrho)^2}$$

Lecture outline

1. Queueing-theoretical model for DVR
2. Lower bounds on performance ($m=1$)
3. Control policies

---


**An optimal light load policy**

### Stochastic Queueing Median (SQM)

Compute median \( p^* \). Then:
1. service demands in FCFS order
2. return to \( p^* \) after each service is completed

### Optimality of SQM policy

\[
\lim_{\lambda \to 0^+} \frac{T_{SQM}}{T^*} = 1
\]

**Proof sketch**
- \( \lambda \to 0^+ \), \( P \) [demand generated when system is empty] \( \to 1 \)
- \( \Rightarrow \) all demands generated with the vehicle at \( p^* \)
- \( \Rightarrow \) \( T_{SQM} = E_{\varphi}(\|X - p^*\|)/v + \bar{s} \)

---

**An optimal spatially-unbiased heavy-load policy**

### Unbiased TSP (UTSP)

Partition \( Q \) into \( r \) subregions \( Q_k \) with \( \int_{Q_k} \varphi(x)dx = 1/r \). Then:
1. within each subregion form sets of size \( n/r \)
2. deposit sets into a queue
3. service sets FCFS by following a TSP tour

Optimize over \( n \).

### Optimality of UTSP policy

\[
\lim_{\rho \to 1^+} \frac{T_{UTSP}(r)}{T_U^*} \leq 1 + 1/r
\]

**Proof**

**(r = 1)**
- idea: reduction to GI/G/1 queue
- \( j \)th set viewed as \( j \)th customer: arrival and service times are i.i.d.!
- inter-arrival distribution is Erlang of order \( n \)
- expected service time is \( n \bar{s} + \beta_{TSP} \sqrt{n} \int_{Q} \varphi^{1/2}(x)dx/v \)
- standard results give upper bound on the wait in queue for a set
- then easy to find upper bound for individual demands
Relation with non-spatial queueing systems:
- Wait time grows as $(1 - \varrho)^{-2}$ instead of $(1 - \varrho)^{-1}$!
- DVR problems are fundamentally different from traditional queueing systems (techniques, results, etc.)

Analysis techniques:
- For light load: locational optimization
- For heavy load: reduction to classic queueing systems or control-theoretical methods

Biased/unbiased:
- Biased service provides strict reduction of optimal system time for any non-uniform $\varphi$

Adaptivity

SQM policy not adaptive:
- SQM unstable as $\varrho \to 1^{-}$
- Intuition: average per-demand travel $\overline{D}$ is fixed
- But stability condition implies $\overline{D} < (1 - \varrho)/\lambda$

UTSP and BTSP policies not adaptive:
- For stability of the queue of sets:
  $$\frac{\lambda}{n} \left( n \overline{s} + \beta_{TSP} \sqrt{n} \int_{Q} \varphi^{1/2}(x) dx / v \right) < 1$$
- Then one should a priori select:
  $$n > \frac{\lambda^2 \beta_{TSP}^2}{\lambda} \left( \int_{Q} \varphi^{1/2}(x) dx \right)^2 \left( \sqrt{v^2 (1 - \varrho)^2} \right)$$
- $\Rightarrow$ wrong selection of $n$ might lead to instability or unacceptable deterioration in performance

DC policy (with $r \to +\infty$)

Implementation:
- NP-hard computation, but effective heuristics

Adaptation: the policy does not require knowledge of
- Vehicle velocity $v$, environment $Q$
- Arrival rate $\lambda$
- Expected on-site service $\overline{s}$

Performance:
- In light load, delay is optimal
- In heavy load, delay is optimal
- Stable in any load condition

Very little known outside of asymptotic regimes
Proof \((r=1)\)

Light load:
- \(\tilde{p}^* \to p^*\) and recovers SQM

Heavy load:
- no well-defined notion of "\(j\)th customer"
- focus on dynamical system

\[
E[n_{i+1}] \leq \lambda \mathbb{E} \left[ \sum_{q=1}^{n_i} s_q + TSP(n_i) \right] \\
\leq \lambda \left( \bar{s} E[n_i] + \beta_{TSP} \int_Q \varphi^{1/2}(x) dx \sqrt{E[n_i]}/v \right)
\]

- upper bound trajectories with the trajectories of virtual dynamical system

\[
z_{i+1} = q z_i + (\lambda/v) \beta_{TSP} \int_Q \varphi^{1/2}(x) dx \sqrt{z_i}
\]

\[
T_{DC} \leq \lim_{i \to +\infty} z_i/\lambda
\]

Receding-Horizon policy

Receding-Horizon (RH)

For \(\eta \in (0,1]\), single agent performs:
1: while no customers, move to empirical median \(\tilde{p}^*\)
2: while customers waiting
   - compute TSP tour through current demands
   - service \(\eta\)-fraction of path

RH policy

Implementation:
- NP-hard computation, but effective heuristics

Adaptation: the policy does not require knowledge of
- vehicle velocity \(v\), environment \(Q\)
- arrival rate \(\lambda\) and spatial density function \(\varphi\)
- expected on-site service \(\bar{s}\)

Performance:
- in light load, delay is optimal
- in heavy load, delay is within a multiplicative factor from optimal
- multiplicative factor depends upon \(\varphi\) and is conjectured to equal 2

Lecture outline

1. Queueing-theoretical model for DVR
2. Lower bounds on performance \((m=1)\)
3. Control policies
Dynamic Vehicle Routing for Robotic Networks
Lecture #4: The multi-vehicle DVR problem

Francesco Bullo¹  Emilio Frazzoli²  Marco Pavone²
Ketan Savla²  Stephen L. Smith²

¹CCDC
University of California, Santa Barbara
bullo@engineering.ucsb.edu

²LIDS and CSAIL
Massachusetts Institute of Technology
{frazzoli,pavone,ksavla,slsmith}@mit.edu

Workshop at the 2010 American Control Conference
Baltimore, Maryland, USA, June 29, 2010, 8:30am to 5:00pm

Lecture outline

1 Territory Partitioning

2 The multi-vehicle DVR problem

3 Multi-vehicle DVR policies based on partitioning

Load balancing in DVR via territory partitioning

Resource allocation in DVR is transcribed into partitioning!
Focus of this lecture is mutivehicle DVR via optimal partitioning

Territory partitioning is ... art

Ocean Park Paintings, by Richard Diebenkorn (1922-1993)
Territory partitioning: optimality and behaviors

DESIGN of performance metrics

- how to cover a region with \( n \) minimum-radius overlapping disks?
- how to design a minimum-distortion (fixed-rate) vector quantizer?
- where to place mailboxes in a city / cache servers on the internet?

ANALYSIS of cooperative distributed behaviors

- how do animals share territory?
- how do they decide foraging ranges?
- how do they decide nest locations?
- what if each robot goes to “center” of own dominance region?
- what if each robot moves away from closest vehicle?

Multi-center functions

Expected wait time (in light load)

\[
\mathcal{H}(p, v) = \int_{v_1} ||x - p_1|| \, dx + \cdots + \int_{v_n} ||x - p_n|| \, dx
\]

- \( n \) robots at \( p = \{p_1, \ldots, p_n\} \)
- environment is partitioned into \( v = \{v_1, \ldots, v_n\} \)

\[
\mathcal{H}(p, v) = \sum_{i=1}^{n} \int_{v_i} f(||x - p_i||) \varphi(x) \, dx
\]

- (map: \( \mathbb{R}^2 \rightarrow \mathbb{R}_0^+ \) density
- (map: \( \mathbb{R}_0^+ \rightarrow \mathbb{R} \) penalty function


Optimal partitioning

The Voronoi partition \( \{V_1, \ldots, V_n\} \) generated by points \( (p_1, \ldots, p_n) \)

\[
V_i(p) = \{x \in Q \mid ||x - p_i|| \leq ||x - p_j||, \forall j \neq i\}
\]

= \( Q \bigcap \bigcup_j \) (half plane between \( i \) and \( j \), containing \( i \))

Descartes 1644, Dirichlet 1850, Voronoi 1908, Thiessen 1911, Fortune 1986 (sweepline algorithm \( O(n \log(n)) \))

Optimal centering (for region \( v \) with density \( \varphi \))

<table>
<thead>
<tr>
<th>function of ( p )</th>
<th>minimizer = center</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p \rightarrow \int_v</td>
<td></td>
</tr>
<tr>
<td>( p \rightarrow \int_v</td>
<td></td>
</tr>
<tr>
<td>( p \rightarrow \text{area}(v \cap \text{disk}(p, r)) )</td>
<td>( r )-area center</td>
</tr>
<tr>
<td>( p \rightarrow \text{radius of largest disk centered}</td>
<td>incenter</td>
</tr>
<tr>
<td>( \text{at } p \text{ enclosed inside } v )</td>
<td></td>
</tr>
<tr>
<td>( p \rightarrow \text{radius of smallest disk cen-}</td>
<td>circumcenter</td>
</tr>
<tr>
<td>( \text{tered at } p \text{ enclosing } v )</td>
<td></td>
</tr>
</tbody>
</table>

From online Encyclopedia of Triangle Centers
How to compute the median of a convex set

For convex planar set $Q$ with strictly positive density $\varphi$,

$$\mathcal{H}_{FW}(p) = \int_{Q} \|p - x\| \varphi(x) dx$$

- $\mathcal{H}_{FW}$ is strictly convex
- the global minimum point is in $Q$ and is called median of $Q$
- compute median via gradient flow with

$$\frac{d}{dp} \mathcal{H}_{FW}(p) = \int_{Q} \frac{p - x}{\|p - x\|} \varphi(x) dx$$

From optimality conditions to algorithms

$$\mathcal{H}(p, v) = \sum_{i=1}^{n} \int_{V_i} f(\|x - p_i\|) \varphi(x) dx$$

Theorem (Alternating Algorithm, Lloyd '57)

1. at fixed positions, optimal partition is Voronoi
2. at fixed partition, optimal positions are "generalized centers"
3. alternate $v$-$p$ optimization

$\implies$ local optimum = center Voronoi partition

Gradient algorithm for multicenter function

After assuming $v$ is Voronoi partition,

$$\mathcal{H}(p) = \sum_{j=1}^{n} \int_{V_j(p)} f(\|x - p_j\|) \varphi(x) dx$$

For $f$ smooth, note simplifications for boundary terms

$$\frac{\partial \mathcal{H}}{\partial p_i}(p) = \int_{V_i(p)} \frac{\partial}{\partial p_i} f(\|x - p_i\|) \varphi(x) dx$$

Gradient algorithm for multicenter function

After assuming $v$ is Voronoi partition,

$$\mathcal{H}(p) = \sum_{j=1}^{n} \int_{V_j(p)} f(\|x - p_j\|) \varphi(x) dx$$

For $f$ smooth, note simplifications for boundary terms

$$\frac{\partial \mathcal{H}}{\partial p_i}(p) = \int_{V_i(p)} \frac{\partial}{\partial p_i} f(\|x - p_i\|) \varphi(x) dx$$

$$+ \int_{\partial V_i(p)} f(\|x - p_i\|) \langle n_i(x), \frac{\partial x}{\partial p_i} \rangle \varphi(x) dx$$
Gradient algorithm for multicenter function

After assuming \( v \) is Voronoi partition,

\[
\mathcal{H}(p) = \sum_{j=1}^{n} \int_{V_j(p)} f(||x - p_j||)\varphi(x)dx
\]

For \( f \) smooth, note simplifications for boundary terms

\[
\frac{\partial \mathcal{H}}{\partial p_i}(p) = \int_{V_i(p)} \frac{\partial}{\partial p_i} f(||x - p_i||)\varphi(x)dx
\]

\[
+ \int_{\partial V_i(p)} f(||x - p_i||) \langle n_i(x), \frac{\partial x}{\partial p_i} \rangle \varphi(x)dx
\]

\[
+ \sum_{j \text{neigh } i} \int_{\partial V_j(p) \cap \partial V_i(p)} f(||x - p_j||) \langle n_{ji}(x), \frac{\partial x}{\partial p_i} \rangle \varphi(x)dx
\]

\text{contrib from neighbors}

Example optimal partition

Lecture outline

1. Territory Partitioning
2. The multi-vehicle DVR problem
3. Multi-vehicle DVR policies based on partitioning

Multi-vehicle DVR problem

- results on single-vehicle DVR generalize easily to the multi-vehicle case
- previous methodology (locational optimization, queueing and control theory, combinatorics) applicable to this case
- main new idea: partitioning

Heavy-load lower bound

spatially-unbiased policies: \( T_\pi \geq \frac{\beta_{TSP}^2}{2} \frac{\lambda \left( \int_0^1 \phi^{1/2}(x) dx \right)^2}{m^2 v^2 (1 - \varrho)^2} \) as \( \varrho \rightarrow 1^-

spatially-biased policies: \( T_\pi \geq \frac{\beta_{TSP}^2}{2} \frac{\lambda \left( \int_0^1 \phi^{2/3}(x) dx \right)^3}{m^2 v^2 (1 - \varrho)^2} \) as \( \varrho \rightarrow 1^-

Proof sketch (for unbiased policies):
- Recall inter-demand distance \( D \geq \beta_{TSP} \frac{\int_0^1 \phi^{1/2}(x) dx}{\sqrt{2N}} \), as \( \varrho \rightarrow 1^- \)
- for stability with \( m \) vehicles:
  \[ s + \frac{D}{v} \leq \frac{m}{\lambda} \quad \Rightarrow \quad s + \beta_{TSP} \frac{\int_0^1 \phi^{1/2}(x) dx}{v \sqrt{2N}} \leq m/\lambda \]
- \( N = \lambda W \) and \( T = W + s \quad \Rightarrow \quad T* \geq \frac{\beta_{TSP}^2}{2} \frac{\lambda \left( \int_0^1 \phi^{1/2}(x) dx \right)^2}{m^2 v^2 (1 - \varrho)^2} \)

Light-load lower bound

Multi-Median
- minimizer \( p^* = \{p_1^*, \ldots, p_m^*\} \) of
  \[ p \mapsto E_p \left[ \min_i \|X - p_i\| \right] = \sum_{i=1}^m \int_{V_i} \|x - p_i\| \phi(x) dx \]

Lower bound (most useful when \( \lambda \rightarrow 0^+ \))
For all policies \( \pi \): \( T* \geq \frac{\beta_{TSP}^2}{2} \frac{\lambda \left( \int_0^1 \phi^{1/2}(x) dx \right)^2}{m^2 v^2 (1 - \varrho)^2} \) as \( \varrho \rightarrow 1^- \)

Proof sketch:
- multi-median: best a priori location to reach a newly arrived demand

Heavy-load lower bound

spatially-unbiased policies: \( T_\pi \geq \frac{\beta_{TSP}^2}{2} \frac{\lambda \left( \int_0^1 \phi^{1/2}(x) dx \right)^2}{m^2 v^2 (1 - \varrho)^2} \) as \( \varrho \rightarrow 1^- \)

spatially-biased policies: \( T_\pi \geq \frac{\beta_{TSP}^2}{2} \frac{\lambda \left( \int_0^1 \phi^{2/3}(x) dx \right)^3}{m^2 v^2 (1 - \varrho)^2} \) as \( \varrho \rightarrow 1^- \)

Proof sketch (for unbiased policies):
- Recall inter-demand distance \( D \geq \beta_{TSP} \frac{\int_0^1 \phi^{1/2}(x) dx}{\sqrt{2N}} \), as \( \varrho \rightarrow 1^- \)
- for stability with \( m \) vehicles:
  \[ s + \frac{D}{v} \leq \frac{m}{\lambda} \quad \Rightarrow \quad s + \beta_{TSP} \frac{\int_0^1 \phi^{1/2}(x) dx}{v \sqrt{2N}} \leq m/\lambda \]
- \( N = \lambda W \) and \( T = W + s \quad \Rightarrow \quad T* \geq \frac{\beta_{TSP}^2}{2} \frac{\lambda \left( \int_0^1 \phi^{1/2}(x) dx \right)^2}{m^2 v^2 (1 - \varrho)^2} \)
An optimal light-load policy

**m Stochastic Queueing Median (mSQM)**

Compute multi-median $p^*$ and assign one vehicle at each median point. Then:
1. Assign demand that falls in $V_i$ to vehicle $i$
2. Each vehicle's service demands in FCFS order
3. Each vehicle returns to $p_k^*$ after each service is completed

**Proof sketch of optimality**
- As $\lambda \rightarrow 0^+$, $\mathbb{P} [\text{demand generated when system is empty}] \rightarrow 1$
- $\Rightarrow$ all demands are generated with the vehicles at $p^*$

An optimal spatially-unbiased heavy-load policy

**Unbiased TSP (UTSP)**
Partition $Q$ into $r$ subregions $Q_k$ with $\int_{Q_k} \varphi(x)dx = 1/r$. Then:
1. Within each subregion form sets of size $n/r$
2. Deposit sets in a queue
3. Service sets FCFS with the first available vehicle by following a TSP tour
Optimize over $n$.

**Optimality of UTSP policy**
$$\lim_{\rho \rightarrow 1^-} \frac{T_{\text{UTSP}}(r)}{T_U^*} \leq 1 + 1/r$$

**Proof sketch of optimality (r=1)**
- Reduction to $\text{GI/G/M}$
Lecture outline

1 Territory Partitioning
2 The multi-vehicle DVR problem
3 Multi-vehicle DVR policies based on partitioning


### Partitioning policies

**Definition ($\pi$-partitioning policy)**

Given $m$ vehicles and single-vehicle policy $\pi$:

- Workspace divided into $m$ subregions
- One-to-one correspondence vehicles/subregions
- Each agent executes the single-vehicle policy $\pi$ within its own subregion

![Diagram](image)

### Motivation

**Performance:**

- light load: problem reduces to locational optimization
- heavy load:
  - delay of optimal single vehicle policy scales as $\lambda |Q|$
  - by (equitably) partitioning, delay reduces to $\lambda |Q| / m = \lambda |Q| / m^2$
  - $\Rightarrow$ delay scales as $m^{-2}$, as in the lower bound

**Implementation:**

- systematic approach to lift adaptive single-vehicle policies to multi-vehicle policies
- coupled with distributed partitioning algorithms, provides distributed multi-vehicle policies

**Performance:**

- light load: problem reduces to locational optimization
- heavy load:
  - delay of optimal single vehicle policy scales as $\lambda |Q|$
  - by (equitably) partitioning, delay reduces to $\lambda |Q| / m = \lambda |Q| / m^2$
  - $\Rightarrow$ delay scales as $m^{-2}$, as in the lower bound

**Implementation:**

- systematic approach to lift adaptive single-vehicle policies to multi-vehicle policies
- coupled with distributed partitioning algorithms, provides distributed multi-vehicle policies

**Motivation**

- systematic approach to lift adaptive single-vehicle policies to multi-vehicle policies
- coupled with distributed partitioning algorithms, provides distributed multi-vehicle policies

**Motivation**

- systematic approach to lift adaptive single-vehicle policies to multi-vehicle policies
- coupled with distributed partitioning algorithms, provides distributed multi-vehicle policies
Motivation

Performance:
- light load: problem reduces to locational optimization
- heavy load:
  1. delay of optimal single vehicle policy scales as $\lambda |Q|$
  2. by (equitably) partitioning, delay reduces to $\lambda \frac{|Q|}{m} = \frac{\lambda |Q|}{m^2}$
  3. $\Rightarrow$ delay scales as $m^{-2}$, as in the lower bound

Implementation:
- systematic approach to lift adaptive single-vehicle policies to multi-vehicle policies
- coupled with distributed partitioning algorithms, provides distributed multi-vehicle policies

distributed multi-vehicle policy = single-vehicle policy + optimal partitioning + distributed algorithm for partitioning

Optimal partitioning in heavy load

Intuition
- per-vehicle workload is $\propto \lambda \int_{Q_k} \varphi(x)dx$
- per-vehicle service capacity is $\propto \lambda \int_{Q_k} \varphi^{1/2}(x)dx$
- optimal partitioning = equalizing per-vehicle workload and service capacity

Definition
A partition $\{Q_k\}_{k=1}^m$ is:
- equitable if $\int_{Q_k} \varphi(x)dx = \int_Q \varphi(x)dx/m$
- simultaneously equitable if
  1. $\int_{Q_k} \varphi(x)dx = \int_Q \varphi(x)dx/m$, and
  2. $\int_{Q_k} \varphi^{1/2}(x)dx = \int_Q \varphi^{1/2}(x)dx/m$

Simultaneously equitable partitions exist for any $Q$ and $\varphi$
(S. Bespamyatnikh, D. Kirkpatrick, and J. Snoeyink, 2000)
Optimal partitioning in heavy load

**Theorem**

Given single-vehicle optimal policy \( \pi^* \), a \( \pi^* \)-partitioning policy using a simultaneously equitable partition is an optimal unbiased policy

**Proof sketch**

- \( P \) [demand arrives in \( Q_k \)] = \( \int_{Q_k} \varphi(x) \, dx = 1/m \)
- arrival rate in region \( k \): \( \lambda_k = \lambda/m \)
- \( \Rightarrow \frac{\lambda_k}{m} \bar{S} = \lambda \bar{S}/m = \varrho < 1 \Rightarrow \) system is stable
- conditional density for region \( k \): \( \varphi(x)/\left( \int_{Q_k} \varphi(x) \, dx \right) = m \varphi(x) \)
- \( T = \sum_{k=1}^{m} \left( \int_{Q_k} \varphi(x) \, dx \frac{\text{TSP}}{2} \sqrt{\frac{\lambda_k}{v^2(1-\varphi)}} \left[ \int_{Q_k} \sqrt{\frac{\varphi(x)}{\int_{Q_k} \varphi(x) \, dx}} \, dx \right]^2 \right) \)
- \( = \sum_{k=1}^{m} \frac{1}{m} T_{\pi^*} \frac{1}{m} \)

**Comments**

If \( \{Q_k\}_{k=1}^{m} \) is only equitable wrt to \( \varphi^{1/2} \)...
- \( \exists \bar{k} \) such that \( \varrho_{\bar{k}} = \lambda(1/m + \varepsilon) \bar{S} = \varrho + \varepsilon \lambda \bar{S} \)
- potentially, policy unstable for \( \varrho < 1 \)!

If \( \{Q_k\}_{k=1}^{m} \) is only equitable wrt to \( \varphi \)...
- per-vehicle service capacity is unbalanced \( \Rightarrow \) policy stable but not optimal
- guaranteed to be within \( m \) of optimal unbiased performance

**Special cases**

Case \( \bar{S} = 0 \):
- stability not an issue:
  \[ \frac{\lambda}{\text{generation rate}} - \frac{m \cdot \text{TSP} \text{length}(n)}{\text{service rate}} = \text{demand growth rate} \]
- since \( \text{TSP} \text{length}(n) \propto \sqrt{n} \Rightarrow \) stability for all \( \lambda, m \)
- \( \text{equitability only wrt to } \varphi^{1/2} \text{ provides optimal performance} \)

Case \( \varphi = \text{uniform} \):
- equitable wrt to \( \varphi \Rightarrow \) equitable wrt to \( \varphi^{1/2} \)
- no need to use algorithms for simultaneous equitability
Special cases

Case $\tilde{s} = 0$:
- stability not an issue:
  \[
  \lambda - \frac{m \cdot n}{\text{TSPlength}(n)} = \text{demand growth rate}
  \]
  - since TSPlength$(n) \propto \sqrt{n}$ hence stability for all $\lambda, m$
- equitability only wrt to $\varphi^{1/2}$ provides optimal performance

Case $\varphi = \text{uniform}$:
- equitable wrt to $\varphi$ \Rightarrow equitable wrt to $\varphi^{1/2}$
- no need to use algorithms for simultaneous equitability

Lecture outline

1. Territory Partitioning
2. The multi-vehicle DVR problem
3. Multi-vehicle DVR policies based on partitioning

Workshop Structure and Schedule

<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:00-8:30am</td>
<td>Coffee Break</td>
<td></td>
</tr>
<tr>
<td>8:30-9:00am</td>
<td>Lecture #1: Intro to dynamic vehicle routing</td>
<td></td>
</tr>
<tr>
<td>9:05-9:50am</td>
<td>Lecture #2: Prelims: graphs, TSPs and queues</td>
<td></td>
</tr>
<tr>
<td>9:55-10:40am</td>
<td>Lecture #3: The single-vehicle DVR problem</td>
<td></td>
</tr>
<tr>
<td>10:40-11:00am</td>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>11:00-11:45pm</td>
<td>Lecture #4: The multi-vehicle DVR problem</td>
<td></td>
</tr>
<tr>
<td>11:45-1:10pm</td>
<td>Lunch Break</td>
<td></td>
</tr>
<tr>
<td>1:10-2:10pm</td>
<td>Lecture #5: Extensions to vehicle networks</td>
<td></td>
</tr>
<tr>
<td>2:15-3:00pm</td>
<td>Lecture #6: Extensions to different demand models</td>
<td></td>
</tr>
<tr>
<td>3:00-3:20pm</td>
<td>Coffee Break</td>
<td></td>
</tr>
<tr>
<td>3:20-4:20pm</td>
<td>Lecture #7: Extensions to different vehicle models</td>
<td></td>
</tr>
<tr>
<td>4:25-4:40pm</td>
<td>Lecture #8: Extensions to different task models</td>
<td></td>
</tr>
<tr>
<td>4:45-5:00pm</td>
<td></td>
<td>Final open-floor discussion</td>
</tr>
</tbody>
</table>
Dynamic Vehicle Routing for Robotic Networks
Lecture #5: Extensions to vehicle networks and distributed algorithms

Francesco Bullo¹ Emilio Frazzoli² Marco Pavone² Ketan Savla² Stephen L. Smith²

¹CCDC
University of California, Santa Barbara
bullo@engineering.ucsb.edu

²LIDS and CSAIL
Massachusetts Institute of Technology
{frazzoli,pavone,ksavla,slsmith}@mit.edu

Workshop at the 2010 American Control Conference
Baltimore, Maryland, USA, June 29, 2010, 8:30am to 5:00pm

Lecture outline

1 Motivation and inspiration from biology
2 Intro to comm models, multi-agent networks and distributed algorithms
3 Partitioning with synchronous proximity-graphs communication
4 Partitioning with gossip (asynchronous pair-wise) communication
5 Partitioning with no explicit inter-vehicle communication
   - No explicit communication policy
   - Game-theoretic interpretation

 Territory partitioning via centralized space planning

UCSB Campus Development Plan, 2008

Territory partitioning akin to animal territory dynamics

Tilapia mossambica, “Hexagonal Territories,” Barlow et al., ’74

Red harvester ants, “Optimization, Conflict, and Nonoverlapping Foraging Ranges,” Adler et al., ’03

Sage sparrows, “Territory dynamics in a sage sparrows population,” Petersen et al ’87
Territory partitioning: behaviors and optimality

**DESIGN** of performance metrics

- how to cover a region with \( n \) minimum-radius overlapping disks?
- how to design a minimum-distortion (fixed-rate) vector quantizer?
- where to place mailboxes in a city / cache servers on the internet?

**ANALYSIS** of cooperative distributed behaviors

- how do animals share territory?
- how do they decide foraging ranges?
- how do they decide nest locations?
- what if each robot goes to “center” of own dominance region?
- what if each robot moves away from closest vehicle?

Intro to communication models, multi-agent networks and distributed algorithms

References


Objective

- meaningful + tractable model
- information/control/communication tradeoffs

Preliminary: Processor network and distributed algorithm

**Processor network**:

- group of processors capable to exchange messages along edges and perform local computations

**Distributed algorithm** for a network of processors consists of

- \( W[i] \), the processor state set
- \( A \), the communication alphabet
- \( stf[i] : W[i] \times A^n \to W[i] \), the state-transition map
- \( msg[i] : W[i] \to A \), the message-generation map
A robotic network is:
- A set of robots moving in space \( Q \)
- An interaction graph

**Relevant graphs**
- Fixed, directed, balanced
- Switching
- Proximity/geometric or state-dependent
- Random, random geometric (packet losses)

**Message model**
- Message
- Packet/bits

**Sensing model**
- Absolute coords other robots
- Absolute coords environment boundary

**Communication models for robotic networks**

**Synchronous control and communication**
- Communication schedule
- Communication alphabet
- Set of values for processor vars
- Message-generation function
- State-transition functions
- Control function
Lecture outline

1. Motivation and inspiration from biology
2. Intro to comm models, multi-agent networks and distributed algorithms
3. Partitioning with synchronous proximity-graphs communication
4. Partitioning with gossip (asynchronous pair-wise) communication
5. Partitioning with no explicit inter-vehicle communication
   - No explicit communication policy
   - Game-theoretic interpretation


Spatially-distributed policies for DVR

Key idea

Distributed multi-vehicle policy = single-vehicle policy + optimal partitioning + distributed algorithm for partitioning

Light load

Optimal pre-positioning
⇒ median Voronoi diagrams

Heavy load

Workload balance
⇒ equitable partitions

Voronoi i-centering law

At each comm round:
1: acquire neighbors’ positions
2: compute own dominance region
3: move towards center of own dominance region

Experimental Partitioning

Equitable and median Voronoi diagrams with synchronous proximity-graphs communication

"Ambitious" goal:
Distributed algorithm to partition the workspace according to:
1. median Voronoi diagram (relevant in light-load)
2. equitable (relevant in heavy load)

Voronoi Diagrams
Voronoi partition \( \{V_1, \ldots, V_m\} \) generated by points \( (p_1, \ldots, p_m) \):
\[ V_i = \{ x \in \Omega | \|x - p_i\|^2 \leq \|x - p_j\|^2, \forall j \neq i \} \]
In general, an equitable Voronoi Diagram fails to exist...
Equitable and median Voronoi diagrams with synchronous proximity-graphs communication

“Ambitious” goal:
Distributed algorithm to partition the workspace according to:
- median Voronoi diagram (relevant in light-load)
- equitable (relevant in heavy load)

Voronoi Diagrams
Voronoi partition \( \{ V_1, \ldots, V_m \} \) generated by points \( (p_1, \ldots, p_m) \):
\[
V_i = \{ x \in \mathbb{Q} \mid \| x - p_i \|^2 \leq \| x - p_j \|^2, \forall j \neq i \}
\]
In general, an equitable Voronoi Diagram fails to exist...

Partitioning using Power Diagrams
Power distance
- \( p = (p_1, \ldots, p_m) \) collection of points in \( \mathbb{Q} \subset \mathbb{R}^2 \)
- each \( p_i \) has assigned a weight \( w_i \in \mathbb{R} \)
- power distance function \( d_P(x, p_i; w_i) = \| x - p_i \|^2 - w_i \)

Power Diagrams
Power diagram \( \{ V_1, \ldots, V_m \} \) generated by weighted points \( (p_1, w_1), \ldots, (p_m, w_m) \):
\[
V_i = \{ x \in \mathbb{Q} \mid \| x - p_i \|^2 - w_i \leq \| x - p_j \|^2 - w_j, \forall j \neq i \}
\]
Existence theorem

Let \( p = (p_1, \ldots, p_m) \) be the positions of \( m \geq 1 \) distinct points in \( Q \). Then there exist weights \((w_1, \ldots, w_m)\) such that the corresponding Power diagram is equitable with respect to \( \varphi \).

Existence theorem

Let \( p = (p_1, \ldots, p_m) \) be the positions of \( m \geq 1 \) distinct points in \( Q \). Then there exist weights \((w_1, \ldots, w_m)\) such that the corresponding Power diagram is equitable with respect to \( \varphi \).
Gradient descent law for equitable partitioning

- \( w_i \) locally controlled by vehicle \( i \)
- locational optimization function

\[
H(w) = \sum_{i=1}^{m} \left( \int_{V_i(w)} \varphi(x) \, dx \right)^{-1} = \sum_{i=1}^{m} \frac{1}{|V_i(w)|} \\
\text{spatially-distributed gradient: } \frac{\partial H}{\partial w_i} = \sum_{j \in N_i} \alpha_{ij} \left( \frac{1}{|V_j|} - \frac{1}{|V_i|} \right)
\]

Gradient law for equitable partitioning

At each comm round:
1: acquire neighbors’ positions
2: compute own dominance region
3: \( w_i \leftarrow w_i - \gamma \frac{\partial H}{\partial w_i} \)

Convergence result

**Theorem (Convergence)**

Assume that the \( p_i \)'s are distinct. Then, the \( w_i \)'s converge asymptotically to a vector of weights that yields an equitable Power diagram

- guaranteed convergence for any set of distinct points
  \( \Rightarrow \) global convergence result
- distributed over the dual graph of the induced Power diagram
  \( \Rightarrow \) communication, on average, with six neighbors
- adjusting the weights sufficient to obtain an equitable diagram
  \( \Rightarrow \) move the \( p_i \)'s to optimize secondary objectives

Including the median Voronoi diagram property

Close to Voronoi:
- basic idea: keep the weights close to zero
- modify the gradient descent law as

\[
\dot{w}_i = - \frac{\partial H}{\partial w_i} - w_i, \quad \dot{p}_i - \frac{\partial H}{\partial w_i} w_i = 0
\]

Motion toward the median:
- basic idea: add a term that enforces computation of the median
- gradient term for computation of the median:

\[
\frac{\partial H_{FW}}{\partial p_i} = \int_{V_i} \frac{p_i - x}{\|p_i - x\|} \varphi(x) \, dx
\]
- modify the gradient descent law as

\[
\dot{w}_i = - \frac{\partial H}{\partial w_i}, \quad \dot{p}_i = \frac{\partial H_{FW}}{\partial p_i} \psi \left( \frac{\partial H}{\partial p_i}, \frac{\partial H_{FW}}{\partial p_i} \right)
\]

Simulation
Lecture outline

1. Motivation and inspiration from biology
2. Intro to comm models, multi-agent networks and distributed algorithms
3. Partitioning with synchronous proximity-graphs communication
4. Partitioning with gossip (asynchronous pair-wise) communication
5. Partitioning with no explicit inter-vehicle communication
   - No explicit communication policy
   - Game-theoretic interpretation

Partitioning with gossip communication

Voronoi+centering law requires:
1. synchronous communication
2. communication along edges of dual graph

Minimalist coordination

- is synchrony necessary?
- is it sufficient to communicate peer-to-peer (gossip)?
- what are minimal requirements?

Gossip (asynchronous pair-wise) partitioning policy

- Random communication between two regions
- Compute two centers
- Compute bisector of centers
- Partition two regions by bisector

Indoor example implementation

- Player/Stage platform
- realistic robot models in discretized environments
- integrated wireless network model & obstacle-avoidance planner


Peer-to-peer convergence analysis (proof sketch 1/3)

Lyapunov function for peer-to-peer territory partitioning

\[ \mathcal{H}(v) = \sum_{i=1}^{n} \int_{V_i} f(\|\text{center}(v_i) - q\|)\phi(q)dq \]

- State space is not finite-dimensional
- Non-convex disconnected polygons
- Peer-to-peer map is not deterministic, ill-defined and discontinuous
- Two regions could have same centers

The space of partitions (proof sketch 2/3)

Definition (Space of finitely-convex partitions)

Fix \( \ell \), the set \( v \) is collections of \( n \) subsets of \( Q \), \( \{v_1, \ldots, v_n\} \), such that

- \( v_1 \cup \cdots \cup v_n = Q \),
- \( \text{interior}(v_i) \cap \text{interior}(v_j) = \emptyset \) if \( i \neq j \), and
- Each \( v_i \) is union of \( \ell \) convex sets

Given sets \( A \) and \( B \), symmetric distance is:

\[ d_\Delta(A, B) = \text{area}\left( (A \cup B) \setminus (A \cap B) \right) \]

Theorem (topological properties of the space of finitely-convex partitions)

Partition space with \((u, v) \mapsto \sum_{i=1}^{n} d_\Delta(u_i, v_i)\) is metric and compact

Convergence with persistent switches (proof sketch 3/3)

- \( X \) is metric space
- Finite collection of maps \( T_i : X \to X \) for \( i \in I \)
- Consider sequences \( \{x_\ell\}_{\ell \geq 0} \subset X \) with

\[ x_{\ell+1} = T_i(\ell)(x_\ell) \]

Assume:

- \( W \subset X \) compact and positively invariant for each \( T_i \)
- \( U : W \to \mathbb{R} \) decreasing along each \( T_i \)
- \( U \) and \( T_i \) are continuous on \( W \)
- There exists probability \( p \in ]0, 1[ \) such that, for all indices \( i \in I \) and times \( \ell \), we have

\[ \text{Prob}\left[ x_{\ell+1} = T_i(\ell)(x_\ell) \mid \text{past} \right] \geq p \]

If \( x_0 \in W \), then almost surely

\[ x_\ell \to (\text{intersection of sets of fixed points of all } T_i) \cap U^{-1}(c) \]
### Lecture Outline

1. Motivation and inspiration from biology
2. Intro to comm models, multi-agent networks and distributed algorithms
3. Partitioning with synchronous proximity-graphs communication
4. Partitioning with gossip (asynchronous pair-wise) communication
5. **Partitioning with no explicit inter-vehicle communication**
   - No explicit communication policy
   - Game-theoretic interpretation

---

**Motivation**

**Gradient policy**

- Cost function: \( \mathcal{H}(p) = \sum_{j=1}^{n} \int_{V_j(p)} ||q - p_j|| \varphi(q) dq \)
- \( \dot{p}_i = - \frac{\partial \mathcal{H}}{\partial p_i}(p) = - \int_{V_i(p)} \frac{\partial}{\partial p_i} ||q - p_i|| \varphi(q) dq \)
- \( p(t) \) converges to a critical point of \( \mathcal{H}(p) \)

- Similar result using the gossip partitioning policy

**Salient Features**

- Explicit agent-to-agent communication
- Needs knowledge of \( \varphi \)

---

**Partitioning with no explicit inter-vehicle communication**

**Inspiration**: Distributed MacQueen algorithm

- Pick any \( m \) generator points \( (p_1, \ldots, p_m) \in Q^m \)
- Iteratively sample points \( q_j \) according to probability density function \( \varphi \)
- At each iteration \( j \):
  - Assign the sampled point to the nearest generator \( i^*(q_j) \in \{1, \ldots, m\} \)
  - Update the position of generator \( i^* \) as
    \[
    p_{i^*} = \frac{(\# \text{pts assigned in past}) p_{i^*} + q_j}{\# \text{pts assigned in past} + 1}
    \]
If the system can fulfill service requests as fast as the rate at which new service requests are generated, then the system is able to visit targets at a rate that is—on average—at least achievable performance and quality of service. Remarkably, algorithms for vector quantization [53] can devise a policy that is stabilizing and yields a quality of service under its effect, the expected number of outstanding targets serviced in the past.

Illustration

At each iteration, the no-communication algorithm computes the "Fermat-Weber (FW) point" with respect to the set of tasks serviced by a vehicle; MacQueen algorithm computes the mean

\[ \text{FW}_i = \arg\min_{p \in Q} \sum_{q \in \text{tasks}_i} \|q - p\| \]

\[ \text{Mean}_i = \frac{1}{|\text{tasks}_i|} \sum_{q \in \text{tasks}_i} q \]

- No simple recursion like the MacQueen algorithm is needed to store locations of all the tasks serviced in the past.
- Sequence of FW points exhibit more complex behavior than the sequence of means.
Analysis of the algorithm

- $p_i(t)$: loitering location of agent $i$ at time $t$
- Sufficient to study convergence of $(p_1(t), \ldots, p_m(t))$

Convergence result
$p(t)$ converges to a critical point of $\mathcal{H}(p)$ with probability one.

Key steps in the proof
- Convergence of the sequence of Fermat-Weber points:
  - $C_i(t) := \{ y \in Q \mid \| \sum_{q \in \text{past tasks}} y - q \| \leq 1 \}$
  - By the properties of the Fermat-Weber point, $p_i(t_j) \in C_i(t_j)$
  - Prove that $p_i(t_{j+1}) \in C_i(t_j)$
  - Prove that $\lim_{j \to \infty} \text{diam}(C_i(t_j)) = 0$ with prob. 1; this implies $p_i(t_j) \to p_i^*$ with prob 1
  - $p_i^*$ is the median of its own Voronoi cell

Coverage as a geometric game

Strategies
- $p = (p_1, \ldots, p_m) \in Q^m$
- When a new task is generated, every vehicle move towards its location

Utility Function
- Upon its generation, each task offers continuous reward at rate unity
- A task expires as soon as two vehicles are present at its location or after $\text{diam}(Q)$ time, whichever occurs first.
- Utility function: expected time spent alone at the next task location
  $$U_i(p_i, p_{-i}) = \mathbb{E}_p[R_i(p, q)] = \mathbb{E}_p \left[ \max \left\{ 0, \min_{j \neq i} \| p_j - q \| - \| p_i - q \| \right\} \right]$$
Coverage as a geometric game

**Strategies**
- \( p = (p_1, \ldots, p_m) \in Q^m \)
- When a new task is generated, every vehicle moves towards its location.

**Utility Function**
- Upon its generation, each task offers continuous reward at rate unity.
- A task expires as soon as two vehicles are present at its location or after \( \text{diam}(Q) \) time, whichever occurs first.
- Utility function: expected time spent alone at the next task location

\[
\mathcal{U}_i(p_i, p_{-i}) = \mathbb{E}_\varphi [R_i(p, q)] = \mathbb{E}_\varphi \left[ \max \left\{ 0, \min_{j \neq i} \| p_j - q \| - \| p_i - q \| \right\} \right]
\]

Properties of the Game

- Potential function: \( \psi(p) = -\sum_{i=1}^m \int_{V_i(p)} \| p_i - q \| \varphi(q) dq \)
- The coverage spatial game is a potential game \( (\mathcal{U}_i(p) = \psi(p) - \psi(p_{-i})) \)
- \( \mathcal{U} \) is a Wonderful Life utility function

Characterization of Equilibria
- critical point of \( \mathcal{H} \) \( \iff \) pure Nash equilibrium

No communication policy as a learning algorithm

**Complete Information**

\[
\dot{p}_i = \frac{\partial}{\partial p_i} \mathcal{U}_i(p) = -\int_{V_i(p)} \frac{p_i - q}{\| p_i - q \|} \varphi(q) dq \quad \Rightarrow \text{gradient descent policy}
\]

**Limited Information**
- No knowledge of \( \varphi \)
- No inter-agent communication

**Approximations**
- Empirical Utility Maximization:
  - \( \hat{p}_i(t) = \arg\max_{x \in Q} \sum_{q \sim \varphi} R_i(x, p_{-i}, q) \)
  - \( \hat{R}_i(x, p_{-i}, q) = \text{diam}(Q) - \| x - q \| \) if vehicle \( i \) reaches task located at \( q \) first, else \( \hat{R}_i(x, p_{-i}, q) = 0 \).
No communication policy as a learning algorithm

Complete Information
\[ \dot{p}_i = \frac{\partial}{\partial p_i} U_i(p) = - \int_{V_i(p)} \frac{p_i - q}{\|p_i - q\|} \varphi(q) dq \quad \Rightarrow \text{gradient descent policy} \]

Limited Information
- No knowledge of \( \varphi \)
- No inter-agent communication

Approximations
- Empirical Utility Maximization:
  \[ p_i(t) = \arg\max_{x \in Q} \sum_{q \sim \varphi} \hat{R}_i(x, p_{-i}, q) \]
- \( \hat{R}_i(x, p_{-i}, q) = \text{diam}(Q) - \|x - q\| \) if vehicle \( i \) reaches task located at \( q \) first, else \( \hat{R}_i(x, p_{-i}, q) = 0 \).

No communication policy as a learning algorithm

Complete Information
\[ \dot{p}_i = \frac{\partial}{\partial p_i} U_i(p) = - \int_{V_i(p)} \frac{p_i - q}{\|p_i - q\|} \varphi(q) dq \quad \Rightarrow \text{gradient descent policy} \]

Limited Information
- No knowledge of \( \varphi \)
- No inter-agent communication

Approximations
- Empirical Utility Maximization:
  \[ p_i(t) = \arg\max_{x \in Q} \sum_{q \sim \varphi} \hat{R}_i(x, p_{-i}, q) \]
- \( \hat{R}_i(x, p_{-i}, q) = \text{diam}(Q) - \|x - q\| \) if vehicle \( i \) reaches task located at \( q \) first, else \( \hat{R}_i(x, p_{-i}, q) = 0 \).

Lecture outline

1. Motivation and inspiration from biology
2. Intro to comm models, multi-agent networks and distributed algorithms
3. Partitioning with synchronous proximity-graphs communication
4. Partitioning with gossip (asynchronous pair-wise) communication
5. Partitioning with no explicit inter-vehicle communication
   - No explicit communication policy
   - Game-theoretic interpretation

Workshop Structure and Schedule

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:00-8:30am</td>
<td>Coffee Break</td>
<td>Intro to dynamic vehicle routing</td>
</tr>
<tr>
<td>8:30-9:00am</td>
<td>Lecture #1:</td>
<td>Prelims: graphs, TSPs and queues</td>
</tr>
<tr>
<td>9:05-9:50am</td>
<td>Lecture #2:</td>
<td>The single-vehicle DVR problem</td>
</tr>
<tr>
<td>9:55-10:40am</td>
<td>Lecture #3:</td>
<td></td>
</tr>
<tr>
<td>10:40-11:00am</td>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>11:00-11:45pm</td>
<td>Lecture #4:</td>
<td>The multi-vehicle DVR problem</td>
</tr>
<tr>
<td>11:45-1:10pm</td>
<td>Lunch Break</td>
<td></td>
</tr>
<tr>
<td>1:10-2:10pm</td>
<td>Lecture #5:</td>
<td>Extensions to vehicle networks</td>
</tr>
<tr>
<td>2:15-3:00pm</td>
<td>Lecture #6:</td>
<td>Extensions to different demand models</td>
</tr>
<tr>
<td>3:00-3:20pm</td>
<td>Coffee Break</td>
<td></td>
</tr>
<tr>
<td>3:20-4:20pm</td>
<td>Lecture #7:</td>
<td>Extensions to different vehicle models</td>
</tr>
<tr>
<td>4:25-4:40pm</td>
<td>Lecture #8:</td>
<td>Extensions to different task models</td>
</tr>
<tr>
<td>4:45-5:00pm</td>
<td></td>
<td>Final open-floor discussion</td>
</tr>
</tbody>
</table>
Dynamic Vehicle Routing for Robotic Networks
Lecture #6: Different Demand Models

Francesco Bullo\textsuperscript{1} Emilio Frazzoli\textsuperscript{2} Marco Pavone\textsuperscript{2}
Ketan Savla\textsuperscript{2} Stephen L. Smith\textsuperscript{2}

\textsuperscript{1}CCDC
University of California, Santa Barbara
bullo@engineering.ucsb.edu

\textsuperscript{2}LIDS and CSAIL
Massachusetts Institute of Technology
\{frazzoli,pavone,ksavla,slsmith\}@mit.edu

Workshop at the 2010 American Control Conference
Baltimore, Maryland, USA, June 29, 2010, 8:30am to 5:00pm

Motivation: Time-Critical Tasks

Motivating Scenario
- Group of UAVs equipped with sensors, monitoring region
- Alerted of events that require close-range observation

Events with time constraints:
- Each event must be observed within a time-window

Events with priority levels:
- Each event has associated level of importance (e.g. 1 to 10)

Lecture outline

1. Stochastic Time Constraints
   - Policy Independent Lower Bound
   - Nearest Depot Assignment Policy
   - Batch Policy

2. Priority Classes of Demands
   - Policy Independent Lower Bound
   - Separate Queues Policy

M. Pavone and E. Frazzoli. Dynamic vehicle routing with stochastic time constraints. In IEEE Int. Conf. on Robotics and Automation, Anchorage, AK, May 2010

DVR with stochastic time constraints

**Model:**
- basic DVR model +
- demand $j$ active for a random patience time $G_j$
- $G_j$'s i.i.d. sequence $\sim F_G$
- demand $j$ expires if not serviced within $G_j$

**Service constraint:**
- $\lim_{j \to +\infty} \mathbb{P}_\pi [W_j < G_j]$: acceptance probability for policy $\pi$
- $\phi^d \in (0, 1)$: desired acceptance probability
- constraint: $\lim_{j \to +\infty} \mathbb{P}_\pi [W_j < G_j] \geq \phi^d$

**Problem formulation**

**Problem statement**

Solve problem $OPT$:

$$\min_{\pi} |\pi|, \text{ subject to } \lim_{j \to \infty} \mathbb{P}_\pi [W_j < G_j] \geq \phi^d$$

**Well-posedness**

- Existence: $\lim_{j \to \infty} \mathbb{P}_\pi [W_j < G_j]$ exists for all $\pi$
- Ergodicity: $\lim_{j \to \infty} \mathbb{P}_\pi [W_j < G_j] = \lim_{t \to +\infty} N^*(t)/N(t)$ (a.s.)

**Proof sketch:**

- main idea: theory of regenerative processes
- regeneration points: times a new demand finds the system empty
- expected length of busy cycles is finite
- use classic limit theorems

**Lower bound**

**Intuition for lower bound:**

$$\mathbb{P} [W_j < G_j] \leq \mathbb{P} \left[ \min_{k \in \{1, \ldots, m\}} \frac{\|X_j - X_k\|}{v} < G_j \right]$$

$$\leq \sup_{(p_1, \ldots, p_m) \in Q^m} \mathbb{P} \left[ \min_{k \in \{1, \ldots, m\}} \frac{\|X_j - p_k\|}{v} < G_j \right]$$

$$\overset{\Delta}{=} \mathcal{H}(p_1, \ldots, p_m)$$

**Lower bound**

$OPT$ is lower bounded by:

$$OPT : \min_{m \in \mathbb{N}_{>0}} m \text{ s.t. } \sup_{(p_1, \ldots, p_m) \in Q^m} \mathcal{H}(p_1, \ldots, p_m) \geq \phi^d$$

**Devised algorithms to solve $OPT$**
Lower bound

Intuition for lower bound:

\[ P[W_j < G_j] \leq P \left[ \min_{k \in \{1, \ldots, m\}} \frac{\|X_j - X_k\|}{v} < G_j \right] \]
\[ \leq \sup_{(p_1, \ldots, p_m) \in Q^m} P \left[ \min_{k \in \{1, \ldots, m\}} \frac{\|X_j - p_k\|}{v} < G_j \right] \]
\[ \leq \sup_{(p_1, \ldots, p_m) \in Q^m} P \left[ \min_{k \in \{1, \ldots, m\}} \frac{\|X_j - p_k\|}{v} < G_j \right] \]
\[ = \gamma(p_1, \ldots, p_m) \]

\[ \text{OPT is lower bounded by:} \]
\[ \text{OPT} : \min_{m \in \mathbb{N}_{>0}} m \]
\[ \text{s.t.} \sup_{(p_1, \ldots, p_m) \in Q^m} \mathcal{H}(p_1, \ldots, p_m) \geq \phi^d \]

Devised algorithms to solve OPT

NDA policy (optimal as \( \lambda \to 0 \))

Nearest Depot Assignment (NDA) policy

Compute maximum of \( \mathcal{H}(\bar{p}_1, \ldots, \bar{p}_m) \).

Then:
1. \( \bar{p}_k \) is depot of \( k \)th vehicle
2. nearest-depot assignment
3. FCFS service

Proof sketch:
- as usual, as \( \lambda \to 0^+ \), the problem reduces to optimal pre-positioning
NDA policy (optimal as $\lambda \to 0$)

Nearest Depot Assignment (NDA) policy
Compute maximum of $\mathcal{H}$: $(\bar{p}_1, \ldots, \bar{p}_m)$.
Then:
1: $\bar{p}_k$ is depot of $k$th vehicle
2: nearest-depot assignment
3: FCFS service

Proof sketch:
- as usual, as $\lambda \to 0^+$, the problem reduces to optimal pre-positioning

Batch policy

Batch (B) policy
Partition $Q$ into $m$ simultaneously equitable subregions and assign one vehicle to each subregion. Then:
1: each vehicle services demands by forming TSP tours

Performance of batch policy
- if $s=0$: $m_B = \min\left\{m \mid \sup_{\theta \in \mathbb{R}^+} (1 - F_G(\theta))(1 - \frac{\lambda \cdot \text{const}}{m^2}) \geq \phi^d\right\}$
- with time windows: $m_B/m^* \leq 3.78$, when $\lambda$ large and $\phi^d \to 1^-$

Characterization of batch policy

Proof sketch ($m=1$):
- upper bound expected length of TSP tour with $\text{const} \cdot \lambda/m^2$, via control-theoretical methods
- use Markov’s ineq to lower bound:

$$\mathbb{P}[W < G] \geq \mathbb{P}[W < G | 2\text{TSP} < \theta] \mathbb{P}[2\text{TSP} < \theta] \geq (1 - F_G(\theta))(1 - \mathbb{E}[2\text{TSP}] / \theta)$$

Lecture outline

1. Stochastic Time Constraints
   - Policy Independent Lower Bound
   - Nearest Depot Assignment Policy
   - Batch Policy

2. Priority Classes of Demands
   - Policy Independent Lower Bound
   - Separate Queues Policy
Demands with priority levels

- \( m \) vehicles
- \( n \) classes of demands
  - 1 = highest priority
  - \( n \) = lowest priority
- Poisson arrivals \( \lambda_1, \ldots, \lambda_n \)
- locations uniformly distributed
  can extend to non-uniform \( \varphi \)

Steady-state expected system-time \( T_1, \ldots, T_n \)

**Goal for vehicles**

\[
\text{Minimize } c_1 T_1 + \cdots + c_n T_n \quad (\uparrow c_i \Rightarrow \uparrow \text{ priority of class } i)
\]


FB, EF, MP, KS, SLS (UCSB, MIT)  Dynamic Vehicle Routing (Lecture 6/8)  29jun10 @ Baltimore, ACC  12 / 23

Demands with priority levels

- \( m \) vehicles
- \( n \) classes of demands
  - 1 = highest priority
  - \( n \) = lowest priority
- Poisson arrivals \( \lambda_1, \ldots, \lambda_n \)
- locations uniformly distributed
  can extend to non-uniform \( \varphi \)

Steady-state expected system-time \( T_1, \ldots, T_n \)

**Goal for vehicles**

\[
\text{Minimize } c_1 T_1 + \cdots + c_n T_n \quad (\uparrow c_i \Rightarrow \uparrow \text{ priority of class } i)
\]


FB, EF, MP, KS, SLS (UCSB, MIT)  Dynamic Vehicle Routing (Lecture 6/8)  29jun10 @ Baltimore, ACC  12 / 23
Literature Review

Classic Priority Queuing


Related Combinatorial Problems


Load Factor and Stability

Stable: Queue remains bounded

Define load factor as

$$\varrho := \frac{\lambda_1\bar{s}_1 + \cdots + \lambda_n\bar{s}_n}{m}$$

- $\lambda_i$ = arrival rate for class $i$
- $\bar{s}_i$ = average on-site service time for class $i$

As before, necessary stability condition is $\varrho < 1$

Two asymptotic regimes

- Light load $\varrho \to 0^+$
- Heavy load $\varrho \to 1^-$

Light load

In light load:

- Each vehicle can return to a median between arrivals
- Priority levels do not change behavior.

Optimal solution:

$m$ vehicle SQM policy is optimal (or an adaptive policy)

$m$ Stochastic Queueing Median ($m$-SQM)

Compute $m$-median locations and assign one vehicle to each location.

Then:

1. service demands in FCFS order
2. return to median after each service is completed

Lower Bound in Heavy Load

Let $\mathcal{T}_c^* = \text{optimal value of cost } c_1\mathcal{T}_1 + \cdots + c_n\mathcal{T}_n$.

Lower bound for every policy

$$\mathcal{T}_c^* \geq \frac{\beta_{\text{TSP}}|Q|}{2m^2v^2(1-\varrho)^2} \sum_{\alpha=1}^{n} \left( c_\alpha + 2 \sum_{j=\alpha+1}^{n} c_j \right) \lambda_\alpha$$

Problem parameters:

- arrival rates $\lambda_1, \ldots, \lambda_n$
- weights $c_1, \ldots, c_n$
- number of vehicles $m$
- environment area $|Q|$
- vehicle speed $v$
Proof Idea of Lower Bound

- Allow remote service of some classes: \( r_\alpha \in \{0, 1\} \) for each class \( \alpha \)
- Travel distance is \( r_\alpha \bar{d}_\alpha \)
- For stability: \( \sum_{i=1}^{n} \lambda_i (r_i \bar{d}_i / v + \bar{s}_i) < m \)
- Can bound travel distance as

\[
\bar{d}_\alpha \geq \beta_{\text{TSP}} \sqrt{2} \sqrt{\frac{|Q|}{\sum_i r_i N_i}}
\]

- Generates a linear program with \( 2^n - 1 \) constraints, one for each combination \( \{r_1, \ldots, r_n\} \)
- Solution to LP is largest lower bound

Separate Queues Policy

Input: Probability distribution \( p = [p_1, \ldots, p_n] \).

Separate Queues Policy

Partition environment into \( m \) equal area regions and assign one vehicle to each region.
Then:
1: Select a class according to probability dist \( p \)
2: Service all demands of selected class following TSP
3: Repeat

Policy performance optimized over \( p \).

Separate Queues Performance

Heavy load performance
For the SQ policy,

\[
\frac{T_{c,\text{SQ}}}{T_c} \leq 2n^2
\]

as \( q \to 1^{-} \).

Heuristic Improvements:

- Receding horizon: service only a fraction \( \eta \) of TSP
- When following TSP, service newly arrived demands within \( \epsilon \) of TSP.

\[
\epsilon = \sqrt{\frac{\mu|Q|}{\sum_{\alpha=1}^{\alpha=n} N_{\alpha}}},
\]

where \( \mu \) is fractional in tour length (i.e., 0.1 for 10% increase)

Simulation of Separate Queues Policy

Simulation:

- Class 1 = yellow
- Class 2 = grey
- \( c_1 = 0.8 \) and \( c_2 = 0.2 \)
- \( p = [0.82, 0.18] \)
Proof idea for upper bound

- In heavy-load, shortest path through $N$ points:
  \[ \beta_{\text{TSP}} \sqrt{|Q|N} \text{ with prob. 1 (BHH theorem)} \]
- Study expected # of outstanding demands at each iteration
  \[ N_i(t+1) \leq f(N_1(t), \ldots, N_m(t), p, \lambda, \bar{s}) \]
- Function $f$ has a linear part plus a sub-linear part
- Bound evolution by stable linear system for all $\varrho < 1$
  \[ \mathcal{N}(t+1) = A(p, \lambda, \bar{s}) \mathcal{N}(t) + B(p, \lambda, \bar{s}) \]
- Allows computation of $\limsup_{t \to +\infty} N_i(t)$
- Apply Little’s theorem $\overline{N}_i = \lambda_i \overline{T}_i$

Lecture outline

1. Stochastic Time Constraints
   - Policy Independent Lower Bound
   - Nearest Depot Assignment Policy
   - Batch Policy

2. Priority Classes of Demands
   - Policy Independent Lower Bound
   - Separate Queues Policy

Workshop Structure and Schedule

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:00-8:30am</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>8:30-9:00am</td>
<td>Lecture #1: Intro to dynamic vehicle routing</td>
</tr>
<tr>
<td>9:05-9:50am</td>
<td>Lecture #2: Prelims: graphs, TSPs and queues</td>
</tr>
<tr>
<td>9:55-10:40am</td>
<td>Lecture #3: The single-vehicle DVR problem</td>
</tr>
<tr>
<td>10:40-11:00am</td>
<td>Break</td>
</tr>
<tr>
<td>11:00-11:45pm</td>
<td>Lecture #4: The multi-vehicle DVR problem</td>
</tr>
<tr>
<td>11:45-1:10pm</td>
<td>Lunch Break</td>
</tr>
<tr>
<td>1:10-2:10pm</td>
<td>Lecture #5: Extensions to vehicle networks</td>
</tr>
<tr>
<td>2:15-3:00pm</td>
<td>Lecture #6: Extensions to different demand models</td>
</tr>
<tr>
<td>3:00-3:20pm</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>3:20-4:20pm</td>
<td>Lecture #7: Extensions to different vehicle models</td>
</tr>
<tr>
<td>4:25-4:40pm</td>
<td>Lecture #8: Extensions to different task models</td>
</tr>
<tr>
<td>4:45-5:00pm</td>
<td>Final open-floor discussion</td>
</tr>
</tbody>
</table>
Vehicle routing with differential constraints

- What happens if the vehicles are subject to non-integrable differential constraints on their motion?
  - Minimum turn radius, constant speed (UAVs, Dubins cars)
  - Minimum turn radius, able to reverse (Reeds-Shepp cars)
  - Differential drive robots (e.g., tanks).
  - Bounded acceleration vehicles (e.g., helicopters, spacecraft).

- Fundamentally different problems, combining combinatorial task specifications with differential geometry and optimal control.

- Decompose the problem, study the asymptotic cases:
  - Heavy load: Traveling salesperson problems.
  - Light load: optimal loitering "stations".

Models of vehicles with differential constraints

- Dubins vehicle
  \[
  \begin{align*}
  \dot{x} &= \cos \theta \\
  \dot{y} &= \sin \theta \\
  \dot{\theta} &= \omega \\
  |\omega| &\leq 1/\rho
  \end{align*}
  \]

- Reeds-Shepp car
  \[
  \begin{align*}
  \dot{x} &= v \cos \theta \\
  \dot{y} &= v \sin \theta \\
  \dot{\theta} &= \omega \\
  v &\in \{-1, 1\} \\
  |\omega| &\leq 1/\rho
  \end{align*}
  \]

- Differential drive
  \[
  \begin{align*}
  \dot{x} &= \frac{1}{\rho} (\omega_1 + \omega_r) \cos \theta \\
  \dot{y} &= \frac{1}{\rho} (\omega_1 + \omega_r) \sin \theta \\
  \dot{\theta} &= \frac{1}{\rho} (\omega_r - \omega_1) \\
  |\omega_1| &\leq 1, \ |\omega_r| \leq 1
  \end{align*}
  \]

- Double integrator
  \[
  \begin{align*}
  \dot{x} &= u \\
  \|x\| &\leq 1 \\
  \|u\| &\leq 1
  \end{align*}
  \]
DTRP formulation

Problem setup
- m identical vehicles in \( Q \)
- Spatio-temporal Poisson process: rate \( \lambda \) and uniform spatial density
- On-site service time \( s = 0 \)

Objective
- Control policy \( \pi = \{ \text{task assignment, scheduling, loitering} \} \)
- \( T_\pi := \limsup_{i \to \infty} \mathbb{E}[\text{wait time of task } i] \); \( T_\pi^* = \inf_\pi T_\pi \)
- Design \( \pi \) for which \( T_\pi \) is equal to or within a constant factor of \( T_\pi^* \)

Stabilizability
- \( \lambda - m \cdot \frac{n}{\text{TSPlength}(n)} \) = task growth rate
  - \( n \): # outstanding tasks
- \( \text{TSPlength}(n) \) strictly sub-linear \( \implies \) stability \( \forall \lambda, m \)
- Euclidean \( \text{TSPlength}(n) = \Theta(n^{1/2}) \) (Beardwood et. al. ’59)
- Euclidean TSP based path planning heuristic \( \implies O(n) \)
- Traveling salesperson problems for differential vehicles.

Stabilizability
- \( \lambda - m \cdot \frac{n}{\text{TSPlength}(n)} \) = task growth rate
  - \( n \): # outstanding tasks
- \( \text{TSPlength}(n) \) strictly sub-linear \( \implies \) stability \( \forall \lambda, m \)
- Euclidean \( \text{TSPlength}(n) = \Theta(n^{1/2}) \) (Beardwood et. al. ’59)
- Euclidean TSP based path planning heuristic \( \implies O(n) \)
- Traveling salesperson problems for differential vehicles.
Outline of the lecture

1. Models of vehicles with differential constraints
2. Traveling salesperson problems
3. The heavy load case
4. The light load case
5. Phase transition in the light load

Traveling Salesperson Problem

Problem Statement

Find the shortest closed curve feasible for the vehicle through a given finite set of points in the plane

- NP-hardness a consequence of the NP-hardness of the Euclidean TSP.
- Does the cost of this TSP increase SUBLINEARLY with $n$?
- Is there a polynomial-time algorithm that returns a tour of length $o(n)$?
- What is the quality of the solution?

Literature review

Towards an upper bound: tiling based algorithms

- The way the ETSP tours are constructed relies on the scaling properties of tours: the length of the tour scales as the coordinates of the points.

- No such scaling exists for the TSP for vehicles with differential constraints, e.g., the bound on the curvature for the Dubins vehicle does not scale with the coordinates of the points!

- Any tiling-based algorithm must account for a "preferential direction", e.g., by penalizing turning for Dubins vehicles.
Bead construction

Bead properties

- Length \( (p_-, q, p_+) \leq \ell + o(\ell^2) \) for all \( q \in B \)
- Width: \( w(\ell) = \frac{\ell^2}{8\rho} + o(\ell^3) \)
- The beads tile the plane
- Useful for Dubins vehicle, Reeds-Shepp car and double integrator

Diamond-like cell for differential drive

Bead construction

Bead properties

- Length \( (p_-, q, p_+) \leq \ell + o(\ell^2) \) for all \( q \in B \)
- Width: \( w(\ell) = \frac{\ell^2}{8\rho} + o(\ell^3) \)
- The beads tile the plane
- Useful for Dubins vehicle, Reeds-Shepp car and double integrator

Diamond-like cell for differential drive

The single-sweep tiling algorithm

- Tile the region with beads
- Sweep the bead rows, while servicing all the targets in every bead as follows:
  - Service every task \( q \) in \( B_- \) using the \( p_- \rightarrow q \rightarrow p_- \) protocol
  - Move from \( p_- \) to \( p_+ \)
  - Service every task \( q \) in \( B_+ \) using the \( p_+ \rightarrow q \rightarrow p_+ \) protocol

The single-sweep tiling algorithm

- Tile the region with beads
- Sweep the bead rows, while servicing all the targets in every bead as follows:
  - Service every task \( q \) in \( B_- \) using the \( p_- \rightarrow q \rightarrow p_- \) protocol
  - Move from \( p_- \) to \( p_+ \)
  - Service every task \( q \) in \( B_+ \) using the \( p_+ \rightarrow q \rightarrow p_+ \) protocol
Analysis of the single-sweep tiling algorithm

Path length calculations

\[ TSP(n) = \text{(bead row length + move to next bead row)} \times \# \text{ bead rows + move to service each task} \times \# \text{ tasks} + \text{tour closure length} \]

- For a Reeds-Shepp car, as \( \ell \to 0 \):
  \[ TSP(n) \leq \left( \sqrt{\frac{|Q|}{\ell^2}} + \frac{\ell}{2} \right) \frac{\sqrt{|Q|}}{w(\ell)^{1/2}} + \ell n + 2 \left( \sqrt{|Q|} + \rho \pi \right) \]
  \[ \leq 16 \rho \frac{|Q|}{\ell^2} + 8 \rho \frac{\sqrt{|Q|}}{\ell} + \ell n + 2 \left( \sqrt{|Q|} + \rho \pi \right) \quad (\because w(\ell) \approx \frac{\ell^2}{8\rho}) \]

- Pick \( \ell = \left( \frac{32 \rho |Q|}{n} \right)^{1/3} \) (i.e., \( \frac{|B|}{|Q|} = \frac{2}{n} \)) \( \implies TSP(n) = O(n^{2/3}) \).

Analysis of the single-sweep tiling algorithm

Path length calculations

\[ TSP(n) = \text{(bead row length + move to next bead row)} \times \# \text{ bead rows + move to service each task} \times \# \text{ tasks} + \text{tour closure length} \]

- For a Dubins vehicle, as \( \ell \to 0 \):
  \[ TSP(n) \leq \left( \sqrt{\frac{|Q|}{\ell^2}} + \frac{w(\ell)}{2} + \kappa \right) \frac{\sqrt{|Q|}}{w(\ell)^{1/2}} + (\ell + \kappa) n + 2 \sqrt{|Q|} + \kappa \]
  \[ \leq 16 \rho \frac{|Q|}{\ell^2} + \sqrt{|Q|} + 16 \kappa \frac{\sqrt{|Q|}}{\ell} + \ell n + \kappa n + 2 \sqrt{|Q|} + \kappa \]

- The \( \kappa n \) term grows linearly in \( n \) for all \( \ell \implies TSP(n) = O(n) \).
Analysis of the single-sweep tiling algorithm

**Path length calculations**

\[ TSP(n) = (\text{bead row length} + \text{move to next bead row}) \times \# \text{ bead rows} + \text{move to service each task} \times \# \text{ tasks} + \text{tour closure length} \]

- For a Dubins vehicle, as \( \ell \to 0 \):
  \[ TSP(n) \leq \left( \sqrt{|Q|} + \frac{w(\ell)}{2} + \kappa \right) \frac{\sqrt{|Q|}}{w(\ell)^{1/2}} + (\ell + \kappa)n + 2\sqrt{|Q|} + \kappa \]
  \[ \leq \ 16\rho \frac{|Q|}{\ell^2} + \sqrt{|Q|} + 16\kappa \frac{|Q|}{\ell^2} + \ell n + \kappa n + 2\sqrt{|Q|} + \kappa \]
  - The \( \kappa n \) term grows linearly in \( n \) for all \( \ell \implies TSP(n) = O(n) \)

**The recursive sweep tiling algorithm**

- Tile \( Q \) with beads such that: \( \frac{|R|}{|Q|} = \frac{1}{2n} \) (i.e., \( \ell \sim n^{-1/3} \))
- Sweep the bead rows, visiting one target per non-empty bead.
- Iterate, using at the \( i \)-th phase a "meta-bead" composed of \( 2^{i-1} \) beads.
- After \( \log n \) phases, visit the outstanding targets in any arbitrary order, e.g., with a greedy strategy.
Analysis of the recursive algorithm

- Theorem: For a Dubins vehicle, with probability one,
  \[
  \limsup_{n \to \infty} \frac{TSP(n)}{n^{2/3}} \leq 24 \sqrt[3]{\frac{\rho|Q|}{n}} \left(1 + \frac{7}{3} \pi \frac{\rho}{\sqrt{|Q|}}\right)
  \]

Outline of the proof

- \(\Pr(\lim_{n \to \infty} \# \text{ tasks remaining after phase } i^* > 24 \log n) = 0\)
- Path length calculations:
  - Phase 1 path length \(O\left(\frac{1}{n^{2/3}}\right) = O\left(n^{2/3}\right)\) \((\because \ell \sim n^{-1/3})\)
  - Subsequent phase path lengths are decreasing geometric series; path length for all \(i^*\) phases is \(O\left(n^{2/3}\right)\)
  - Path length by greedy heuristic is \(O(\log n)\)

Summary of TSPs

- Lower bound: \(\mathbb{E}[TSP(n)] \in \Omega(n^{2/3})\)
- Upper bound: \(\mathbb{E}[TSP(n)] \in O(n^{2/3})\)
- \(TSP(n)\) is of order \(n^{2/3}\); constant factor approximation algorithms
- Computational complexity of the algorithms is of order \(n\)

Stabilizability of the DTRP

- \(\frac{\lambda}{TSP(n)} - \frac{m \cdot n}{TSP(n)} = \text{task growth rate}\)
- \(n: \# \text{ outstanding tasks}\)
- \(\mathbb{E}[TSP(n)] \in \Theta(n^{2/3}) \implies \text{trivial receding horizon TSP-based policies are stable for the DTRP for all } \lambda \text{ and } m\)
Outline of the lecture

1. Models of vehicles with differential constraints
2. Traveling salesperson problems
3. The heavy load case
4. The light load case
5. Phase transition in the light load

The heavy load case: nearest neighbor lower bound

Outline of the calculations

- Let \( n_\pi \) be the number of outstanding tasks at steady-state under stable policy \( \pi \)
- Calculate (an upper bound on) expected distance from an arbitrary vehicle configuration to closest among \( n_\pi \) points, \( \delta^*(n_\pi) \)
- At steady-state: \( \frac{\lambda}{m} = \frac{1}{\mathbb{E}[\delta^*(n_\pi)]]} \)
- Little’s formula: \( \lambda T_\pi = n_\pi \)

Example: Dubins vehicle

\[ \mathbb{E}[\delta^*(n_\pi)] = \frac{3}{4} \left( \frac{3\rho |Q|}{n_\pi} \right)^{1/3} \]

Steady state+ Little’s formula: \( \frac{\lambda}{m} = \frac{4}{3} \left( \frac{\lambda T_\pi}{3\rho |Q|} \right)^{1/3} \)

\[ \lim_{\frac{\lambda}{m} \to +\infty} \frac{\lambda T_\pi m^3}{\lambda^3} \geq \frac{81}{64} \rho |Q| \]

The multiple sweep tiling algorithm

The single vehicle version

- Tile \( Q \) with beads of length \( \ell = c/\lambda \)
- Update outstanding task list
- Execute single sweep tiling algorithm
- Goto 2.

The multi-vehicle version

- Divide \( Q \) into \( m \) equal "strips"
- Assign one vehicle to every strip
- Each vehicle executes the multiple sweep algorithm in its own strip
The multiple sweep tiling algorithm

The single vehicle version

- Tile \( Q \) with beads of length \( \ell = c/\lambda \)
- Update outstanding task list
- Execute single sweep tiling algorithm
- Goto 2.

The multi-vehicle version

- Divide \( Q \) into \( m \) equal "strips"
- Assign one vehicle to every strip
- Each vehicle executes the multiple sweep algorithm in its own strip

Analysis of the multiple sweep algorithm

**General protocol**

- Each bead can be treated as a separate queue, with Poisson arrival process with intensity \( \lambda_B = \lambda \frac{|B|}{|Q|} \)
- The vehicle visits each bead with at a rate no smaller than \( \mu_B \approx (\text{single sweep path length})^{-1} \)
- The system time is no greater than the system time for the corresponding M/D/1 queue: \( \bar{T}^* \leq \frac{1}{\mu_B} \left( 1 + \frac{1}{2} \frac{\lambda_B}{\mu_B - \lambda_B} \right) \)
- Optimize over \( \ell \)

**Example: Dubins vehicle**

\[
\lambda_B = \frac{\ell^2 \lambda}{16 \rho |Q|}; \quad \mu_B \geq \frac{\ell^2 m}{16 \rho |Q|} \left( 1 + \frac{7}{3} \pi \frac{\rho}{\sqrt{|Q|}} \right)^{-1}
\]

\[
\limsup_{\lambda \to +\infty} \frac{\bar{T}^* m^3}{\lambda^2} \leq 71 \rho |Q| \left( 1 + \frac{7}{3} \pi \frac{\rho}{\sqrt{|Q|}} \right)^3
\]

Outline of the lecture

1. Models of vehicles with differential constraints
2. Traveling salesperson problems
3. The heavy load case
4. The light load case
5. Phase transition in the light load
The light load case

- The target generation rate is very small: $\lambda/m \to 0^+$

In such case:
- Almost surely all vehicles will have enough time to return to some "loitering station" between task completion/generation times
- The problem is reduced to the choice of the loitering stations that minimizes the system time

Introducing differential constraints

- Novel challenges:
  - Vehicles possibly cannot stop (e.g., Dubins vehicle, Reeds-Shepp car)
  - Strategies are more complex than defining a loitering "point"
- How many of the results from the Euclidean case carry over to this case?

A simple lower bound

- The length of shortest feasible path from a vehicle positioned at $p \in \mathbb{R}^2$ to an arbitrary point $q \in Q$ is lower bounded by $\|q - p\|$

- A simple lower bound on $T^*$ is obtained by relaxing differential constraints

  $T^* \geq \mathcal{H}_m^*(Q)$

  $\mathcal{H}_m^*(Q) = \Theta \left( \frac{1}{\sqrt{m}} \right)$

The Median Circling (MC) Policy

Assign "virtual" generators to each agent. All agents do the following, in parallel (possibly asynchronously):
- Update the generator position according to a gradient descent law.
- Service targets in own region, returning to a "loitering circle" of radius $2.91\rho$ centered on their generator position when done

- We have
  $$\lim_{\lambda/m \to 0^+} T_{MC} \leq \mathcal{H}_m^*(Q) + 3.76\rho$$

- Furthermore,
  $$\lim_{\mathcal{H}_m \to \infty, \lambda/m \to 0^+} \frac{T_{MC}}{T^*} = 1.$$
Assign "virtual" generators to each agent. All agents do the following, in parallel (possibly asynchronously):

- Update the generator position according to a gradient descent law.
- Service targets in own region, returning to a "loitering circle" of radius $2.91\rho$ centered on their generator position when done.

We have

$$\lim_{\lambda/m \to 0^+} T_{MC} \leq H_m^*(Q) + 3.76\rho$$

Furthermore,

$$\lim_{m \to \infty, \lambda/m \to 0^+} \frac{T_{MC}}{T^*} = 1.$$
The Strip Loitering (SL) policy

- Divide the environment $Q$ into strips of width $\min\left\{ \frac{k_2(Q, \rho)}{m^{2/3}}, 2\rho \right\}$
- Design a closed loitering path that bisects the strips. All vehicles move along this path, equally spaced, with dynamic regions of responsibility.
- Each vehicle services targets in own region, returning to the "nominal" position on the loitering path.

\[
\lim_{m \to +\infty} T_{SL} m^{1/3} \leq k_3(Q, \rho), \quad \text{and} \quad \lim_{m \to +\infty} \frac{T_{SL}}{m} \leq k_4(Q, \rho).
\]

Illustration of the SL policy

Outline of the lecture

1. Models of vehicles with differential constraints
2. Traveling salesperson problems
3. The heavy load case
4. The light load case
5. Phase transition in the light load
Phase transition in the light load

- We have two policies: Median Circling (MC), and Strip Loitering (SL). Which is better?

Define the non-holonomic density $d_p = \frac{v^2 m}{|Q|}$.
- MC is optimal when $d_p \rightarrow 0$.
- SL is within a constant factor of the optimal as $d_p \rightarrow +\infty$.

**phase transition**: the optimal organization changes from territorial (MC) to gregarious (SL) depending on the non-holonomic density of the agents.

---

Estimate of the critical density

- Ignoring boundary conditions (e.g., consider the unbounded plane), we can compare the coverage cost for the two policies analytically:

$$T_{SL} < T_{MC} \iff d_p > 0.0587$$

(i.e., transition occurs when the area of the dominance region is about 4-5 times the area of the minimum turning radius circle).

Simulation results yield $d_p^{\text{crit}} \approx 0.0759$ (within a factor 1.3 of the analytical result).

---

Phase transition in the light load

- We have two policies: Median Circling (MC), and Strip Loitering (SL). Which is better?

Define the non-holonomic density $d_p = \frac{v^2 m}{|Q|}$.
- MC is optimal when $d_p \rightarrow 0$.
- SL is within a constant factor of the optimal as $d_p \rightarrow +\infty$.

**phase transition**: the optimal organization changes from territorial (MC) to gregarious (SL) depending on the non-holonomic density of the agents.

---

Estimate of the critical density

- Ignoring boundary conditions (e.g., consider the unbounded plane), we can compare the coverage cost for the two policies analytically:

$$T_{SL} < T_{MC} \iff d_p > 0.0587$$

(i.e., transition occurs when the area of the dominance region is about 4-5 times the area of the minimum turning radius circle).

Simulation results yield $d_p^{\text{crit}} \approx 0.0759$ (within a factor 1.3 of the analytical result).
Dynamic Vehicle Routing Summary

<table>
<thead>
<tr>
<th>Euclidean vehicle</th>
<th>Dubins vehicle, Reeds-Shepp car Double integrator, Differential drive</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{E}[\text{TSP Length}]$ ($n \to \infty$)</td>
<td>$\Theta(n^2)$</td>
</tr>
<tr>
<td>$\bar{T}^*$ ($\frac{1}{m} \to \infty$)</td>
<td>$\Theta\left(\frac{1}{m^2}\right)$</td>
</tr>
<tr>
<td>$\bar{T}^*$ ($\frac{1}{m} \to 0$, $\frac{m}{</td>
<td>Q</td>
</tr>
<tr>
<td>$\bar{T}^*$ ($\frac{1}{m} \to 0$, $\frac{m}{</td>
<td>Q</td>
</tr>
</tbody>
</table>

Workshop Structure and Schedule

- 8:00-8:30am: Coffee Break
- 8:30-9:00am: Lecture #1: Intro to dynamic vehicle routing
- 9:05-9:50am: Lecture #2: Prelims: graphs, TSPs and queues
- 9:55-10:40am: Lecture #3: The single-vehicle DVR problem
- 10:40-11:00am: Break
- 11:00-11:45am: Lecture #4: The multi-vehicle DVR problem
- 11:45-1:10pm: Lunch Break
- 1:10-2:10pm: Lecture #5: Extensions to vehicle networks
- 2:15-3:00pm: Lecture #6: Extensions to different demand models
- 3:00-3:20pm: Coffee Break
- 3:20-4:20pm: Lecture #7: Extensions to different vehicle models
- 4:25-4:40pm: Lecture #8: Extensions to different task models
- 4:45-5:00pm: Final open-floor discussion

Lecture outline

1. Models of vehicles with differential constraints
2. Traveling salesperson problems
3. The heavy load case
4. The light load case
5. Phase transition in the light load
Dynamic Vehicle Routing for Robotic Networks
Lecture #8: Different Task Models

Francesco Bullo1  Emilio Frazzoli2  Marco Pavone2  Ketan Savla2  Stephen L. Smith2

1CCDC
University of California, Santa Barbara
bullo@engineering.ucsb.edu

2LIDS and CSAIL
Massachusetts Institute of Technology
{frazzoli,pavone,ksavla,slsmith}@mit.edu

Workshop at the 2010 American Control Conference
Baltimore, Maryland, USA, June 29, 2010, 8:30am to 5:00pm

Motivation for Team Forming

- Group of vehicles monitoring a region
- Several different sensing modalities:
  - electro-optical,
  - infra-red,
  - synthetic aperture radar,
  - foliage penetrating radar,
  - etc.
- Each event requires a subset of sensing modalities
- Equip each vehicle with a single sensing modality
- Form appropriate team to properly assess each event

How do we create teams in real-time to observe each event (service each request)?

Lecture outline

1 Dynamic Team Forming

2 Three Policies
- Complete Team
- Task-Specific Team Policy
- Scheduled Task-Specific Team Policy

3 Analysis of Policies
- Throughput vs System Time
- Comparison of Results

Motivation for Team Forming

- Group of vehicles monitoring a region
- Several different sensing modalities:
  - electro-optical,
  - infra-red,
  - synthetic aperture radar,
  - foliage penetrating radar,
  - etc.
- Each event requires a subset of sensing modalities
- Equip each vehicle with a single sensing modality
- Form appropriate team to properly assess each event

How do we create teams in real-time to observe each event (service each request)?

1 Dynamic Team Forming

2 Three Policies
- Complete Team
- Task-Specific Team Policy
- Scheduled Task-Specific Team Policy

3 Analysis of Policies
- Throughput vs System Time
- Comparison of Results

Literature Review

Scaling laws in Robotic Networks


Throughput vs Delay in Wireless Networks


Graph Coloring
T. A. McKee and F. R. McMorris. Topics in Intersection Graph Theory, volume 2 of Monographs on Discrete Mathematics and Applications. SIAM, 1999

Lecture outline

1 Dynamic Team Forming
2 Three Policies
3 Analysis of Policies

Dynamic Team Forming

Set of services \( \{r_1, \ldots, r_k\} \).

Vehicle properties:
- \( k \) different vehicle types.
- Vehicle type \( j \in \{1, \ldots, k\} \), can provide only service \( r_j \).

Task (demand) model:
- Poisson and Uniform arrivals
- Each task requires a subset of services in \( \{r_1, \ldots, r_k\} \).
- \( K \) different types of tasks
- Tasks of type \( \alpha \) arrive at rate \( \lambda_\alpha \).
- Task completed when required vehicles simultaneously spend on-site service time at location.


Load Factor and Stability

- \( R_\alpha \in \{0, 1\}^k \) is zero-one column vector recording services required for task-type \( \alpha \).
- on-site service for task-type \( \alpha \) is \( \bar{s}_\alpha \)
- \( m_j \) vehicles provide service \( r_j \).

Necessary stability condition:

\[
\begin{bmatrix}
\lambda_1 \bar{s}_1 \\
\vdots \\
\lambda_K \bar{s}_K
\end{bmatrix} \begin{bmatrix}
R_1 & \cdots & R_K
\end{bmatrix} < 
\begin{bmatrix}
m_1 \\
\vdots \\
m_K
\end{bmatrix}
\]

Load factor is now a vector

Example of Team Forming

- \( k = 4 \) different services, \( \{r_1, r_2, r_3, r_4\} \).
- \( m = 8 \) vehicles, two of each type: \( m_j = 2 \) for \( j \in \{1, 2, 3, 4\} \).
- \( K = 6 \) task types, \( \{r_1\}, \{r_2\}, \{r_3\}, \{r_4\}, \{r_1, r_3\}, \{r_2, r_4\} \).

Task type \( \alpha = \{r_1, r_3\} \) has on-site service \( \bar{s}_\alpha \), arrival rate \( \lambda_\alpha \), and

\[
R_\alpha = [1 0 1 0]^T.
\]
Lecture outline

1. Dynamic Team Forming

2. Three Policies
   - Complete Team Policy
   - Task-Specific Team Policy
   - Scheduled Task-Specific Team Policy

3. Analysis of Policies

Task-Type Unbiased Policies

For a policy $\pi$:
- System time of each task-type $T_{\pi,1}, \ldots, T_{\pi,K}$
- Feasible set of system times are subset of $\mathbb{R}^K$
- Optimization space similar to priority queues but with teaming

To simplify, consider task-type unbiased policies

$$T_{\pi,1} = T_{\pi,2} = \cdots = T_{\pi,K} =: T_{\pi}$$

and the optimization: $\inf_{\pi} T_{\pi}$.

Policy 1: Complete Team Policy

**Complete Team Policy**

1. Form $\min\{m_1, \ldots, m_k\}$ teams of $k$ vehicles, each team contains one vehicle of each type.
2. Have each team meet and move as a single entity.
3. In each region run UTSP policy (from Lecture 3).

Can also use Divide & Conquer policy for each team

Two services $y, b$
- 3 task-types $y, b, \{y, b\}$
- 4 vehicles
  - 2 yellow
  - 2 blue

FB, EF, MP, KS, SLS (UCSB, MIT)  Dynamic Vehicle Routing (Lecture 8/8)  29Jun10 @ Baltimore, ACC  11 / 24
Policy 2: Task-specific Team Policy

- $m_j$ vehicles provide service $r_j$.
- $r_j$ appears in $e_j^T [R_1 \cdots R_K] \mathbf{1}_K$ task types.
- If $m_j \geq e_j^T [R_1 \cdots R_K] \mathbf{1}_K \Rightarrow$ enough vehicles of type $j$ to create dedicated team for each task type.
- Create $m_{\text{TST}}$ teams, where:

$$m_{\text{TST}} := \left\lfloor \min_j \left\{ \frac{m_j}{e_j^T \mathbf{1}_K} \right\} \right\rfloor$$

Task-Specific Team Policy

1. For each of the $K$ task types, create $m_{\text{TST}}$ teams of vehicles.
2. Service each task by one of its $m_{\text{TST}}$ corresponding teams, according to the UTSP policy.

**Policy 3: Scheduled Task-Specific Team Policy**

Scheduled Task-specific team policy

Partition $Q$ into $\min_i \{m_i\}$ regions and assign one robot of each type to each region.

1. In each region form a queue for each task type.
2. For each time slot in the schedule:
   - Divide robots into teams to form required task types.
   - For each team, service corresponding queue with TSP tour.
Policy 3: Scheduled Task-Specific Team Policy

Service schedule:
- two time slots $L = 2$
- slot one: $\{y\}, \{b\}$
- slot two: $\{y, b\}$

Assumptions for Analysis

Assumptions:
- $m_i = m/k$ for each vehicle type $i$.
- $\lambda_\alpha = \lambda/K$ for each task-type $\alpha$.
- on-site service has mean $\bar{s}$ and is upper bounded by $s_{\text{max}}$.
- $pK$ of the $K$ task-types require service $r_j$, where $p \in [1/k, 1]$.

With assumptions, necessary stability condition becomes
$$\frac{\lambda}{m} < \frac{1}{pk\bar{s}}.$$  

Define per-vehicle throughput as $B_m := \lambda/m$.

Throughput vs System Time Profile

$$B_m \mapsto \begin{cases} \max \left\{ T_{\text{min}}, \frac{T_{\text{ord}}(B_m/B_{\text{crit}})}{1 - B_m/B_{\text{crit}}} \right\}, & \text{if } B_m < B_{\text{crit}}, \\ +\infty, & \text{if } B_m \geq B_{\text{crit}}. \end{cases}$$

- $T_{\text{min}}$ = minimum achievable system time for positive throughput.
- $B_{\text{crit}}$ = maximum achievable throughput (or capacity).
- $T_{\text{ord}}$ = system time at a constant fraction of capacity $(3 - \sqrt{5})/2 \approx 38\%$ of capacity $B_{\text{crit}}$.

Example (Single vehicle DVR)

$$B_{\text{crit}} = \frac{1}{\bar{s}}$$

$$T_{\text{min}} = \mathbb{E}_{p} [||X - p^*||]/v + \bar{s} \quad \text{ (light load)}$$

$$T_{\text{ord}} \approx C(\int Q \varphi^{1/2}(x)dx)^2/v^2 \quad \text{ (heavy load numerator)}$$
### Throughput vs System Time Profile

![Graph showing the relationship between throughput and system time.](image)

### System Time for each Policy

<table>
<thead>
<tr>
<th>Policy</th>
<th>$T_{\text{ord}}$</th>
<th>$T_{\text{ord}}$</th>
<th>$B_{\text{crit}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower bound ($T^*$)</td>
<td>$\sqrt{k}$</td>
<td>$k$</td>
<td>$\frac{1}{pk^2}$</td>
</tr>
<tr>
<td>Complete Team</td>
<td>$\sqrt{k}$</td>
<td>$k$</td>
<td>$\frac{1}{k^3}$</td>
</tr>
<tr>
<td>Task-Specific</td>
<td>$\sqrt{pk\kappa}$</td>
<td>$pk\kappa$</td>
<td>$\frac{1}{pk^2}$</td>
</tr>
<tr>
<td>Scheduled Task-Specific</td>
<td>$L\sqrt{k}$</td>
<td>$Lk$</td>
<td>$\frac{\kappa}{\max L}$</td>
</tr>
</tbody>
</table>

where $L \in [p\kappa, \kappa]$

**Best policies for different scenarios:**
- If throughput is low, then use complete team
- If $p$ is close to 1, then use complete team
- If $p$ is close to $1/k$, then for best capacity use
  - Task-Specific if enough vehicles
  - Scheduled Task-Specific otherwise

### Lecture outline

1. **Dynamic Team Forming**
2. **Three Policies**
   - Complete Team
   - Task-Specific Team Policy
   - Scheduled Task-Specific Team Policy
3. **Analysis of Policies**
   - Throughput vs System Time
   - Comparison of Results

### Workshop Structure and Schedule

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:00-8:30am</td>
<td><strong>Coffee Break</strong></td>
</tr>
<tr>
<td>8:30-9:00am</td>
<td>Lecture #1: Intro to dynamic vehicle routing</td>
</tr>
<tr>
<td>9:05-9:50am</td>
<td>Lecture #2: Prelims: graphs, TSPs and queues</td>
</tr>
<tr>
<td>9:55-10:40am</td>
<td>Lecture #3: The single-vehicle DVR problem</td>
</tr>
<tr>
<td>10:40-11:00am</td>
<td><strong>Break</strong></td>
</tr>
<tr>
<td>11:00-11:45pm</td>
<td>Lecture #4: The multi-vehicle DVR problem</td>
</tr>
<tr>
<td>11:45-1:10pm</td>
<td>Lecture #5: Extensions to vehicle networks</td>
</tr>
<tr>
<td>1:10-2:10pm</td>
<td><strong>Lunch Break</strong></td>
</tr>
<tr>
<td>2:15-3:00pm</td>
<td>Lecture #6: Extensions to different demand models</td>
</tr>
<tr>
<td>3:00-3:20pm</td>
<td><strong>Coffee Break</strong></td>
</tr>
<tr>
<td>3:20-4:20pm</td>
<td>Lecture #7: Extensions to different vehicle models</td>
</tr>
<tr>
<td>4:25-4:40pm</td>
<td>Lecture #8: Extensions to different task models</td>
</tr>
<tr>
<td>4:45-5:00pm</td>
<td>Final open-floor discussion</td>
</tr>
</tbody>
</table>