Dynamic Vehicle Routing for Robotic Networks
Lecture #1: Introduction

Francesco Bullo[1] Emilio Frazzoli[2] Marco Pavone[2]
Ketan Savla[2] Stephen L. Smith[2]

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

[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

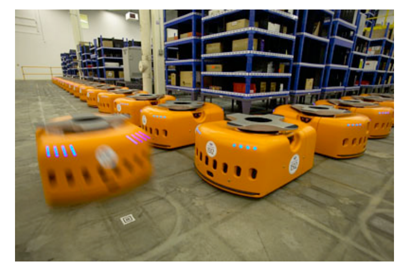
Acknowledgements

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).

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

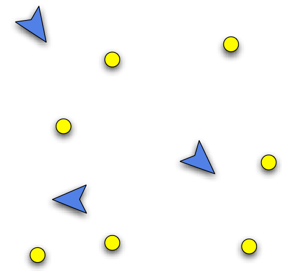


Lecture outline

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)

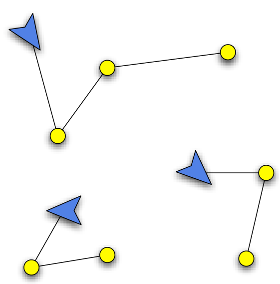
Determine a set of paths that allow vehicles to service the demands

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

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)

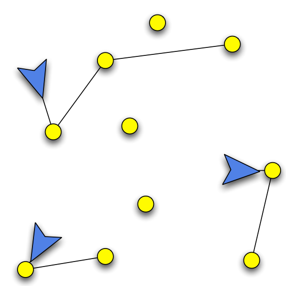
Determine a set of paths that allow vehicles to service the demands

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

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

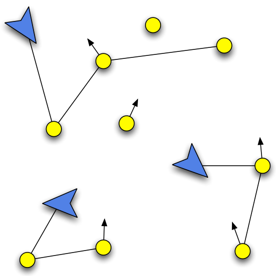
Dynamic vehicle routing (New info in real time, Psaraftis '88)
- New demands arise in real-time
- Existing demands evolve over time

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



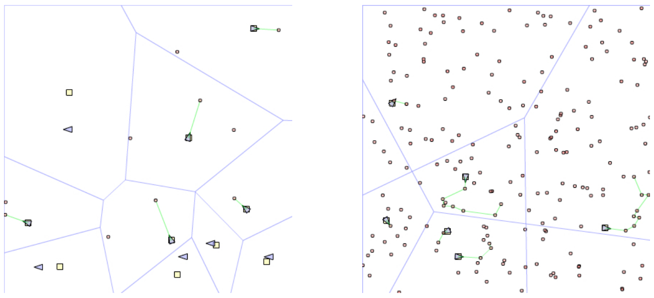
Vehicle routing (All info known ahead of time, Dantzig '59)

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

Light and heavy load regimes



Lecture outline

From coordination and static routing to Dynamic Vehicle Routing

Simple coordination problems arise in static environments

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

Routing policies vs planning algorithms

dynamic, stochastic and adversarial events take place

1. design **policies** (in contrast to pre-planned routes or motion planning algorithms) to specify how to react to events
2. 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

Lecture outline

Contributions of our recent works

Comprehensive framework for DVR in robotic systems

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

Bibliography on DVR and queueing for robotic networks

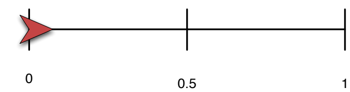
1. K. Savla, E. Frazzoli, and F. Bullo. Traveling Salesperson Problems for the Dubins vehicle. *IEEE Transactions on Automatic Control*, 53(6):1378–1391, 2008
2. K. Savla, F. Bullo, and E. Frazzoli. Traveling Salesperson Problems for a double integrator. *IEEE Transactions on Automatic Control*, 54(4):788–793, 2009
3. J. J. Enright, K. Savla, E. Frazzoli, and F. Bullo. Stochastic and dynamic routing problems for multiple UAVs. *AIAA Journal of Guidance, Control, and Dynamics*, 34(4):1152–1166, 2009
4. S. L. Smith and F. Bullo. The dynamic team forming problem: Throughput and delay for unbiased policies. *Systems & Control Letters*, 58(10-11):709–715, 2009
5. M. Pavone, N. Bisnik, E. Frazzoli, and V. Isler. A stochastic and dynamic vehicle routing problem with time windows and customer impatience. *ACM/Springer Journal of Mobile Networks and Applications*, 14(3):350–364, 2009
6. A. Arsie, K. Savla, and E. Frazzoli. Efficient routing algorithms for multiple vehicles with no explicit communications. *IEEE Transactions on Automatic Control*, 54(10):2302–2317, 2009
7. S. L. Smith, M. Pavone, F. Bullo, and E. Frazzoli. Dynamic vehicle routing with priority classes of stochastic demands. *SIAM Journal on Control and Optimization*, 48(5):3224–3245, 2010
8. M. Pavone, K. Savla, and E. Frazzoli. Sharing the load. *IEEE Robotics and Automation Magazine*, 16(2):52–61, 2009
9. M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo. Equitable partitioning policies for mobile robotic networks. *IEEE Transactions on Automatic Control*, 2010. (Submitted Dec 2008 and Aug 2009) to appear
10. M. Pavone, E. Frazzoli, and F. Bullo. Distributed and adaptive algorithms for vehicle routing in a stochastic and dynamic environment. *IEEE Transactions on Automatic Control*, May 2010. (Submitted, Apr 2009) to appear
11. S. D. Bopardikar, S. L. Smith, F. Bullo, and J. P. Hespanha. Dynamic vehicle routing for translating demands: Stability analysis and receding-horizon policies. *IEEE Transactions on Automatic Control*, 55(11), 2010. (Submitted, Mar 2009) to appear
12. S. D. Bopardikar, S. L. Smith, and F. Bullo. On vehicle placement to intercept moving targets. *Automatica*, March 2010. Submitted
13. F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. L. Smith. Dynamic vehicle routing for robotic systems. *Proceedings of the IEEE*, May 2010. Submitted
14. H. A. Waisanen, D. Shah, and M. A. Dahleh. A dynamic pickup and delivery problem in mobile networks under information constraints. *IEEE Transactions on Automatic Control*, 53(6):1419–1433, 2008
15. R. Szechtman, M. Kress, K. Lin, and D. Cfir. Models of sensor operations for border surveillance. *Naval Research Logistics*, 55(1):27–41, 2008
16. S. Itani. *Dynamic Systems and Subadditive Functionals*. PhD thesis, Massachusetts Institute of Technology, 2009

Lecture outline

Plain-vanilla re-optimization?

Example: DVR on segment

- Objective: minimize average waiting time
- Strategy: re-optimize at each event



1. For adversarial target generation, vehicle travels forever without ever servicing any request $\Longrightarrow$ **unstable queue of outstanding requests**
2. Even if queue remains bounded, what about performance? how far from the optimal?

Plain-vanilla re-optimization?

Example: DVR on segment

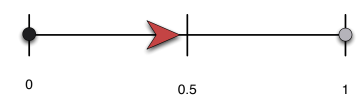
- Objective: minimize average waiting time
- Strategy: re-optimize at each event

1. For adversarial target generation, vehicle travels forever without ever servicing any request $\Longrightarrow$ **unstable queue of outstanding requests**
2. 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

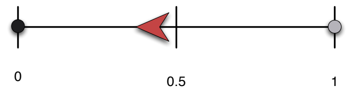


1. For adversarial target generation, vehicle travels forever without ever servicing any request $\Longrightarrow$ **unstable queue of outstanding requests**
2. Even if queue remains bounded, what about performance? how far from the optimal?

Plain-vanilla re-optimization?

Example: DVR on segment

- Objective: minimize average waiting time
- Strategy: re-optimize at each event



1. For adversarial target generation, vehicle travels forever without ever servicing any request $\Longrightarrow$ **unstable queue of outstanding requests**
2. 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

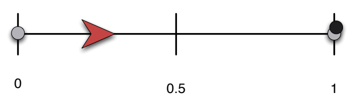
1. For adversarial target generation, vehicle travels forever without ever servicing any request $\Longrightarrow$ **unstable queue of outstanding requests**
2. Even if queue remains bounded, what about performance? how far from the optimal?

FB, EF, MP, KS, SLS (UCSB, MIT) Dynamic Vehicle Routing (Lecture 1/8) 29jun10 @ Baltimore, ACC 15 / 18

Plain-vanilla re-optimization?

Example: DVR on segment

- Objective: minimize average waiting time
- Strategy: re-optimize at each event



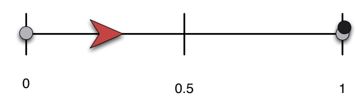
1. For adversarial target generation, vehicle travels forever without ever servicing any request $\Longrightarrow$ **unstable queue of outstanding requests**
2. Even if queue remains bounded, what about performance? how far from the optimal?

FB, EF, MP, KS, SLS (UCSB, MIT) Dynamic Vehicle Routing (Lecture 1/8) 29jun10 @ Baltimore, ACC 15 / 18

Plain-vanilla re-optimization?

Example: DVR on segment

- Objective: minimize average waiting time
- Strategy: re-optimize at each event



1. For adversarial target generation, vehicle travels forever without ever servicing any request $\Longrightarrow$ **unstable queue of outstanding requests**
2. Even if queue remains bounded, what about performance? how far from the optimal?

FB, EF, MP, KS, SLS (UCSB, MIT) Dynamic Vehicle Routing (Lecture 1/8) 29jun10 @ Baltimore, ACC 15 / 18

Online algorithms?

Online algorithms (Jaillet and M. R. Wagner '06)

- online algorithm operates based on input information up to the current time
- online algorithm is (worst-case) r-competitive if

$$\mathrm{Cost}_{\mathrm{online}}(I) \leq r\mathrm{Cost}_{\mathrm{optimal\ offline}}(I), \qquad \forall \text{ problem instances } I.$$

Disadvantages

1. cumulative cost
2. worst-case analysis
3. not possible to include *a-priori* information (e.g., arrival rate)
4. not as clear what competitive ratio means
5. so far, only few simple DVR problems admit online algorithms

Lecture outline

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*	
11:00-11:45pm	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

Dynamic Vehicle Routing for Robotic Networks
Lecture #2: Preliminary Results in Combinatorics

Francesco Bullo[1] Emilio Frazzoli[2] Marco Pavone[2]
Ketan Savla[2] Stephen L. Smith[2]

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

[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

Key references for this lecture

Graph Theory Basics:
R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, 2 edition, 2000

Combinatorial Optimization:
B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*, volume 21 of *Algorithmics and Combinatorics*. Springer, 4 edition, 2007

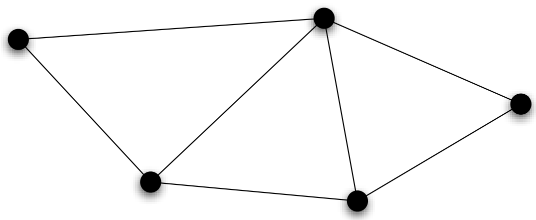
Stochastic TSP:
J. M. Steele. *Probability Theory and Combinatorial Optimization*. SIAM, 1987

Basic Queueing Theory:
L. Kleinrock. *Queueing Systems. Volume I: Theory*. Wiley, 1975

Outline

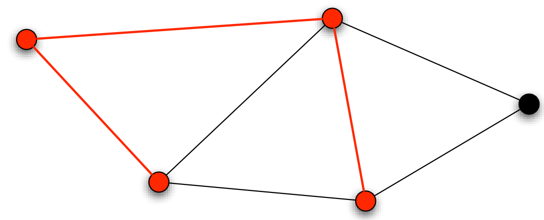
Graph Theory Review

- An undirected graph $G = (V, E)$.
- a path in G is a sequence $v_1, e_1, v_2, \dots, v_k, e_k, v_{k+1}$, 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

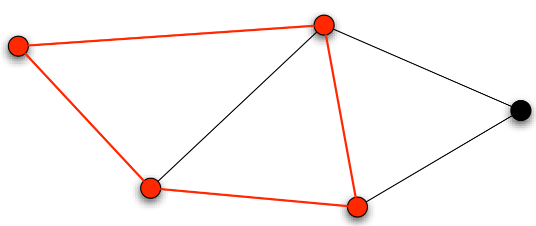
- An undirected graph $G = (V, E)$.
- a path in G is a sequence $v_1, e_1, v_2, \dots, v_k, e_k, v_{k+1}$, with
  - $e_i \neq e_j$ for $i \neq j$.
  - $v_i \neq v_j$ for all $i \neq j$.
- A circuit or cycle has $v_1 = v_{k+1}$.
- A Hamiltonian path is a path that contains all vertices.
- Similarly define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



FB, EF, MP, KS, SLS (UCSB, MIT) Dynamic Vehicle Routing (Lecture 2/8) 29jun10 @ Baltimore, ACC 5 / 29

Graph Theory Review

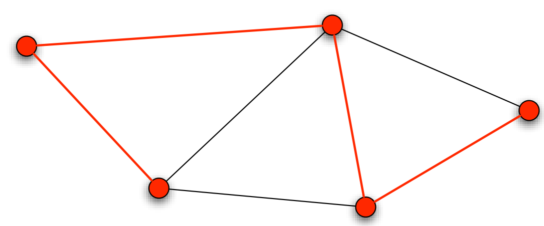
- An undirected graph $G = (V, E)$.
- a path in G is a sequence $v_1, e_1, v_2, \dots, v_k, e_k, v_{k+1}$, with
  - $e_i \neq e_j$ for $i \neq j$.
  - $v_i \neq v_j$ for all $i \neq j$.
- A circuit or cycle has $v_1 = v_{k+1}$.
- A Hamiltonian path is a path that contains all vertices.
- Similarly define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



FB, EF, MP, KS, SLS (UCSB, MIT) Dynamic Vehicle Routing (Lecture 2/8) 29jun10 @ Baltimore, ACC 5 / 29

Graph Theory Review

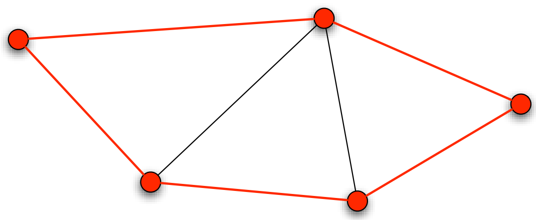
- An undirected graph $G = (V, E)$.
- a path in G is a sequence $v_1, e_1, v_2, \dots, v_k, e_k, v_{k+1}$, with
  - $e_i \neq e_j$ for $i \neq j$.
  - $v_i \neq v_j$ for all $i \neq j$.
- A circuit or cycle has $v_1 = v_{k+1}$.
- A Hamiltonian path is a path that contains all vertices.
- Similarly define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



FB, EF, MP, KS, SLS (UCSB, MIT) Dynamic Vehicle Routing (Lecture 2/8) 29jun10 @ Baltimore, ACC 5 / 29

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

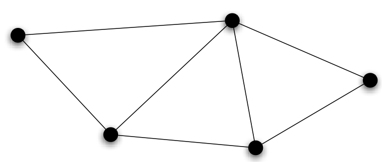
$$c(\{v_1, v_2\}) + c(\{v_2, v_3\}) \geq c(\{v_1, v_3\}) \text{ for all } v_1, v_2, v_3 \in V.$$

- **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
  1. is a tree
  2. connects all vertices together



Minimum Spanning Tree Problem

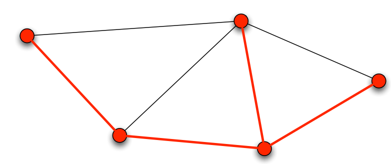
Given: a weighted graph $G - (V, E, c)$
Task: find a spanning tree $T = (E_T, V_T)$ such that $\sum_{e \in E_T} c(e)$ is minimum.

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 (where $|V| = n$)

Minimum Spanning Tree

- A **tree** is a graph with no cycles
- A **spanning tree** of G is a subgraph that
  1. is a tree
  2. 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 (where $|V| = n$)

Hamiltonian Cycle Decision Problem

Hamiltonian Cycle

Given: An undirected graph G.

Question: Does G contain a Hamiltonian cycle?

Hamiltonian Cycle is NP-complete

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

Recall, a problem is NP-complete if

- Every solution can be verified in polynomial time (NP).
- Every problem in NP can be reduced to it.

Outline

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 NP-hard
- To show 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 $\Leftrightarrow$ optimum TSP tour has length n.

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 $\sim 5\%$ 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

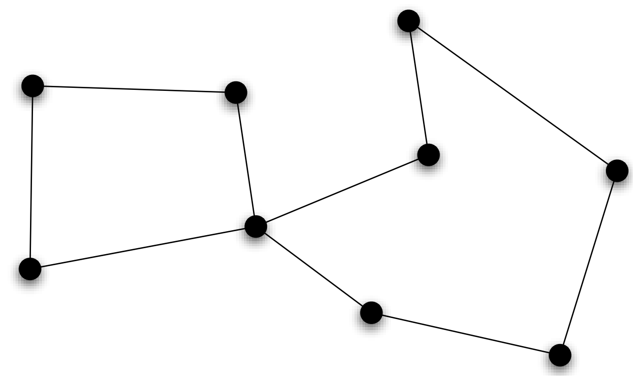
Metric TSP

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

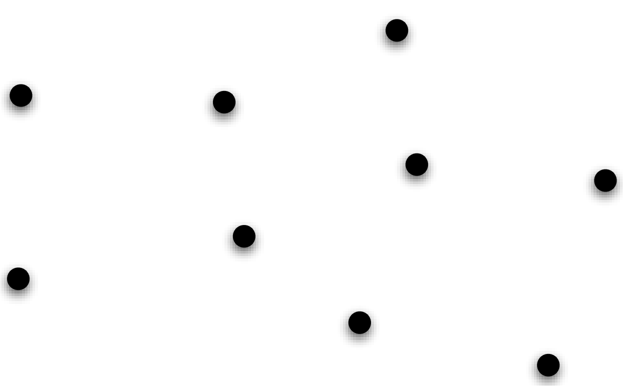
- Eulerian graph: degree of each vertex is even
- Eulerian walk: Closed walk containing every edge.
- Graph has Eulerian walk $\Leftrightarrow$ Eulerian.
- Eulerian walk can be computed in $O(|V| + |E|)$ time.



Double-Tree Algorithm

Double-Tree Algorithm

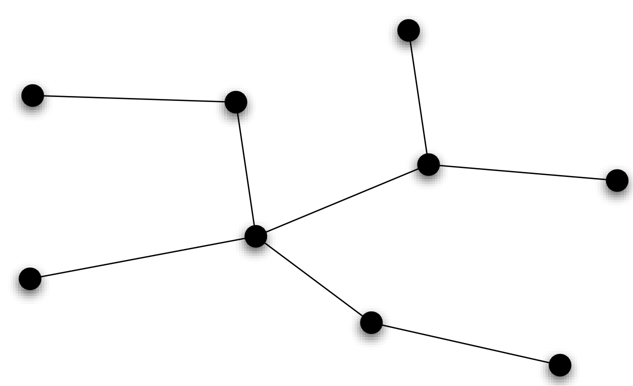
```
Find a minimum spanning tree T of graph G_n.
Ḡ := graph containing two copies of each edge in T.
Compute Eulerian walk in Eulerian graph Ḡ.
Walk gives ordering, ignore all but first occurrence
of vertex.
```



Double-Tree Algorithm

Double-Tree Algorithm

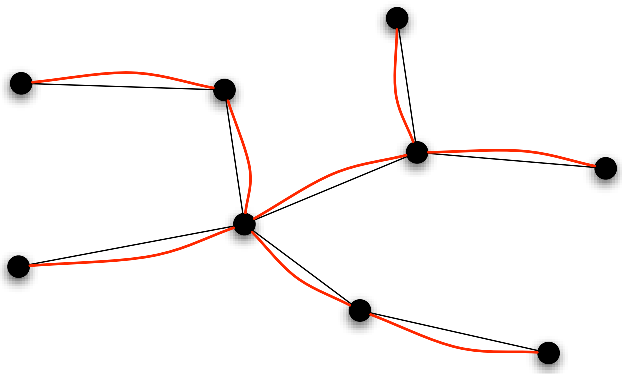
```
Find a minimum spanning tree T of graph G_n.
Ḡ := graph containing two copies of each edge in T.
Compute Eulerian walk in Eulerian graph Ḡ.
Walk gives ordering, ignore all but first occurrence
of vertex.
```



Double-Tree Algorithm

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

Theorem

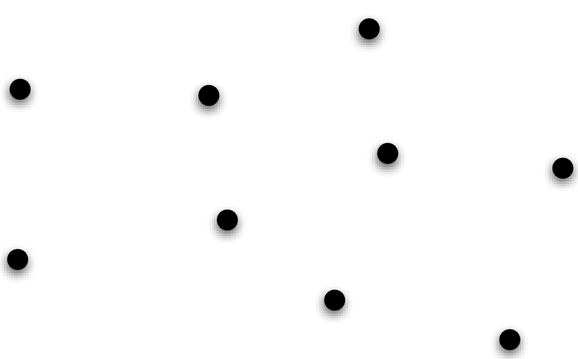
Double-Tree Algorithm is a 2-approx algorithm for the Metric TSP. Its running time is $O(n^2)$.

- 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

Christofides' Algorithm

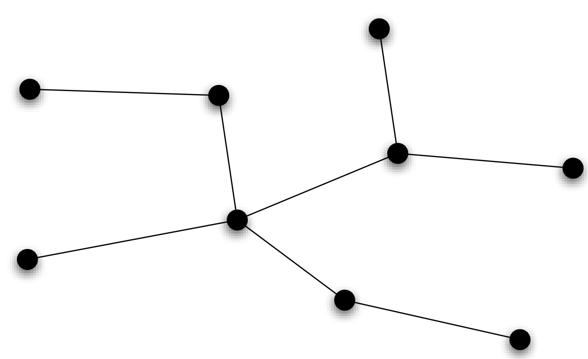
1: Find a minimum spanning tree T of G.
2: Let W be the set of vertices with odd degree in T.
3: Find the minimum weight perfect matching M in subgraph generated by W.
4: Find an Eulerian path in $G := (V_n, E(T) \cup M)$, (skip vertices already seen).



Christofides' Algorithm

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).

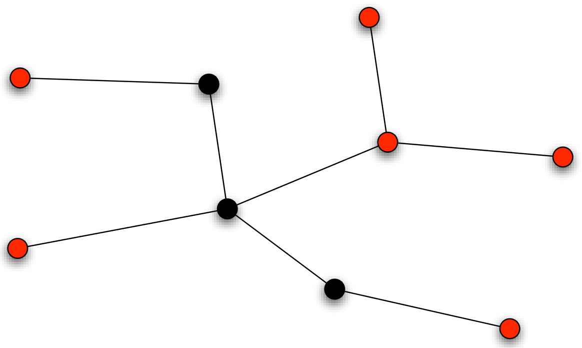


FB, EF, MP, KS, SLS (UCSB, MIT) Dynamic Vehicle Routing (Lecture 2/8) 29jun10 @ Baltimore, ACC 16 / 29

Christofides' Algorithm

Christofides' Algorithm

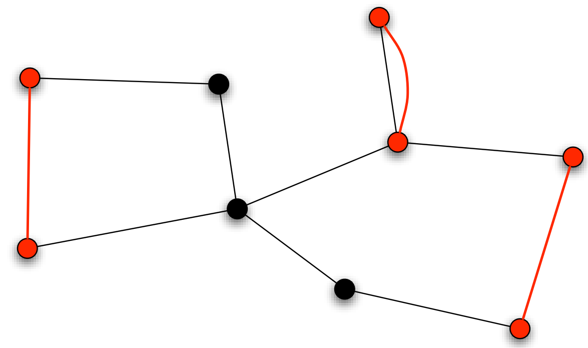
1: Find a minimum spanning tree T of G.
2: Let W be the set of vertices with odd degree in T.
3: Find the minimum weight perfect matching M in subgraph generated by W.
4: Find an Eulerian path in $G := (V_n, E(T) \cup M)$, (skip vertices already seen).



Christofides' Algorithm

Christofides' Algorithm

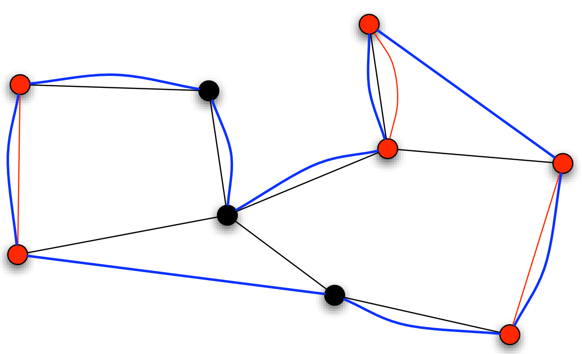
1: Find a minimum spanning tree T of G.
2: Let W be the set of vertices with odd degree in T.
3: Find the minimum weight perfect matching M in subgraph generated by W.
4: Find an Eulerian path in $G := (V_n, E(T) \cup M)$, (skip vertices already seen).



Christofides' Algorithm

Christofides' Algorithm

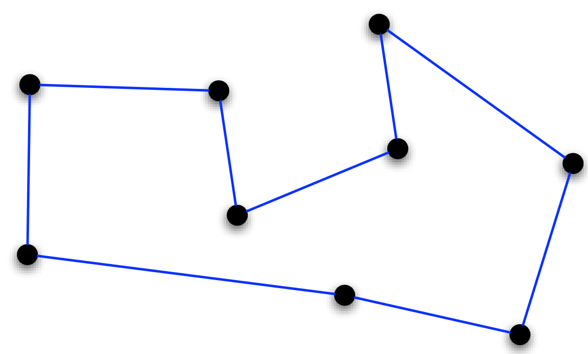
1: Find a minimum spanning tree T of G.
2: Let W be the set of vertices with odd degree in T.
3: Find the minimum weight perfect matching M in subgraph generated by W.
4: Find an Eulerian path in $G := (V_n, E(T) \cup M)$, (skip vertices already seen).



Christofides' Algorithm

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 $O\big(n^3(\log n)^c\big)$ time.

Practical value limited to due c's dependence on ϵ.

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

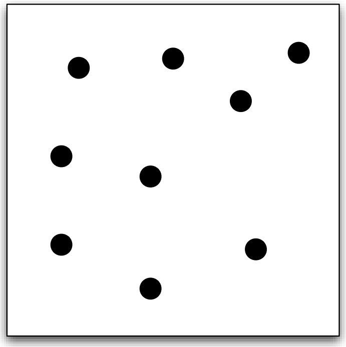
$$\text{ETSP}(Q_n) \leq \sqrt{2n} + 7/4.$$

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 $Q_n := \{x_1, \dots, x_n\}$ 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 ℓ_n denote worst-case TSP length through n pts.
- Then $\ell_n \leq \ell_{n-1} + 2c/\sqrt{n}$.
- Summing we get $\ell(n) \leq C\sqrt{n}$.

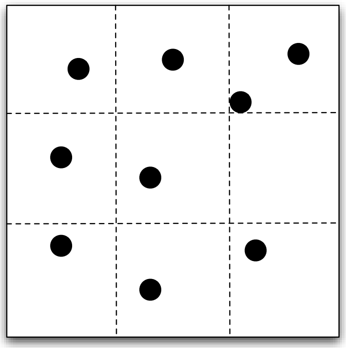


Worst-case TSP Length Upper Bound (Intuition)

- Consider $Q_n := \{x_1, \dots, x_n\}$ 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 ℓ_n denote worst-case TSP length through n pts.
- Then $\ell_n \leq \ell_{n-1} + 2c/\sqrt{n}$.
- Summing we get $\ell(n) \leq C\sqrt{n}$.

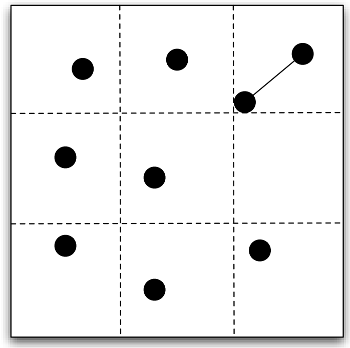


Worst-case TSP Length Upper Bound (Intuition)

- Consider $Q_n := \{x_1, \dots, x_n\}$ 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 ℓ_n denote worst-case TSP length through n pts.
- Then $\ell_n \leq \ell_{n-1} + 2c/\sqrt{n}$.
- Summing we get $\ell(n) \leq C\sqrt{n}$.

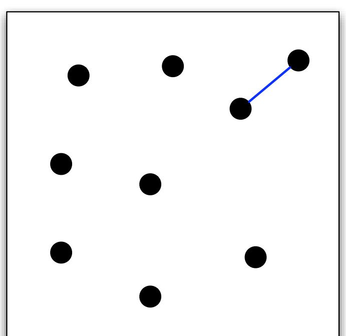


Worst-case TSP Length Upper Bound (Intuition)

- Consider $Q_n := \{x_1, \dots, x_n\}$ 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 ℓ_n denote worst-case TSP length through n pts.
- Then $\ell_n \leq \ell_{n-1} + 2c/\sqrt{n}$.
- Summing we get $\ell(n) \leq C\sqrt{n}$.

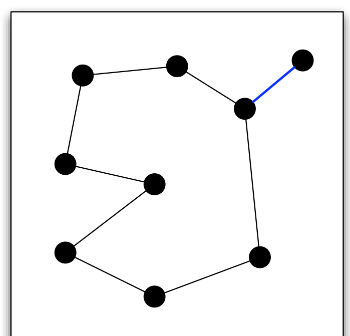


Worst-case TSP Length Upper Bound (Intuition)

- Consider $Q_n := \{x_1, \dots, x_n\}$ 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 ℓ_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

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

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

For uniform distribution in square of area A

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

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

$$\beta_{\mathrm{TSP},2} \simeq 0.7120.$$

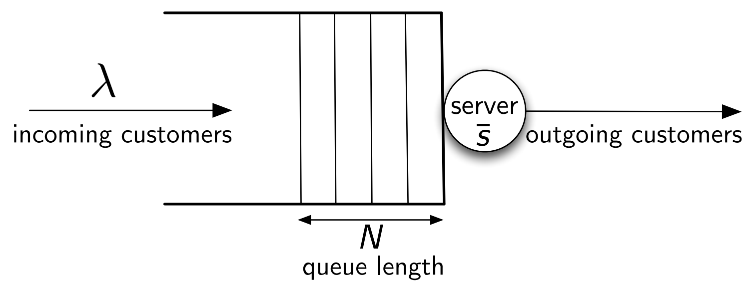
Summary of Traveling Salesman Problem

- 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

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



- **Arrivals** modeled as stochastic process with rate λ
- **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 λ
- General service times with mean $\bar{s}$
- m servers

Little's Law and Load Factor

Define:

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

Little's Law/Theorem

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

- For m servers, define **load factor** as

$$\varrho := \frac{\lambda \bar{s}}{m}$$

- **Necessary condition** for stable queue is $\varrho < 1$.

Wait-time examples

For $M/D/1$ queue:

$$\overline{W} = \frac{\varrho \bar{s}}{2(1-\varrho)}$$

For $M/G/1$ queue:

$$\overline{W} = \frac{\lambda \bar{s^2}}{2(1-\varrho)}$$

For $G/G/1$ queue (Kingman, 1962):

$$\overline{W} \leq \frac{\lambda(\sigma_a^2 + \sigma_s^2)}{2(1-\varrho)}$$

and the upper bound becomes exact as $\varrho \to 1^-$.

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[1] Emilio Frazzoli[2] Marco Pavone[2]
Ketan Savla[2] Stephen L. Smith[2]

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

[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

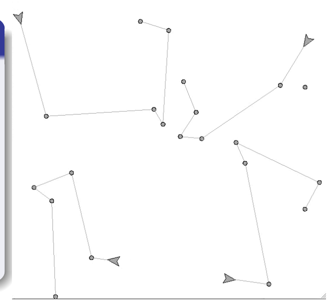
1. Queueing-theoretical model for DVR
2. Lower bounds on performance (m=1)
3. Control policies

D. J. Bertsimas and G. J. van Ryzin. A stochastic and dynamic vehicle routing problem in the Euclidean plane. *Operations Research*, 39:601–615, 1991

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**:

1. arrival process
2. service model
3. performance measure

General queueing-theoretical model for DVR 1/2

Arrival process: spatio-temporal Poisson

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

Service model:

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

General queueing-theoretical model for DVR 1/2

Arrival process: spatio-temporal Poisson

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

Service model:

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

General queueing-theoretical model for DVR 2/2

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

Problem statement

Solve optimization problem over all causal routing policies π:

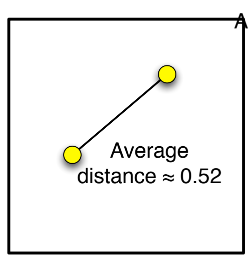
$$\inf_{\pi} \overline{T}_{\pi}$$

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 ≈ 0.52.
- conditional expected travel time between two consecutive demands > 0.52.



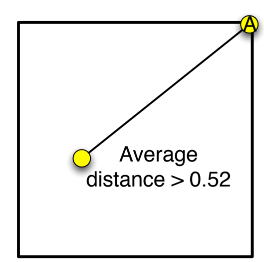
M/G/m methodology is not applicable!

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 ≈ 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

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

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

$$\varrho = \lambda \bar{s}/m < 1$$

Sufficient condition for stability:
Surprisingly, $\varrho < 1$ is also sufficient for stability $\implies$ stability condition is **independent** of the size and shape of $\mathcal{Q}$

Analysis approach

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

Lecture outline

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

D. J. Bertsimas and G. J. van Ryzin. A stochastic and dynamic vehicle routing problem in the Euclidean plane. *Operations Research*, 39:601–615, 1991

D. J. Bertsimas and G. J. van Ryzin. Stochastic and dynamic vehicle routing with general interarrival and service time distributions. *Advances in Applied Probability*, 25:947–978, 1993

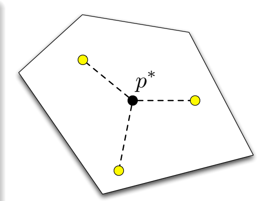
Light-load lower bound

Median

- minimizer p^* of

$$p \mapsto \int_{\mathcal{Q}} \|x - p\| \varphi(x) dx = \mathbb{E}_\varphi[\|X - p\|]$$

- best a priori location to reach next demand



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

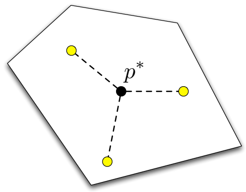
For all policies π: $\overline{T}_\pi \geq \mathbb{E}_\varphi[\|X - p^*\|]/v + \bar{s}$

Proof sketch:
- $\overline{T} = \overline{W}_{\text{travel}} + \overline{W}_{\text{on-site}} + \bar{s}$.
- $\overline{W}_{\text{travel}} \geq \mathbb{E}_\varphi[\|X - p^*\|]/v$

Light-load lower bound

Median

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

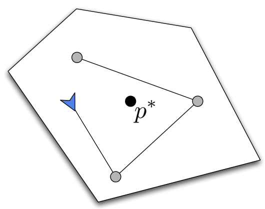


Lower bound (most useful when $\lambda \to 0^+$)

For all policies π: $\overline{T}_\pi \geq \mathbb{E}_\varphi[\|X - p^*\|]/v + \bar{s}$

Proof sketch:

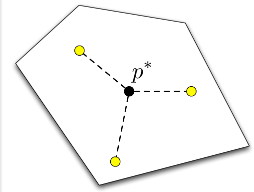
- $\overline{T} = \overline{W}_{\text{travel}} + \overline{W}_{\text{on-site}} + \bar{s}$.
- $\overline{W}_{\text{travel}} \geq \mathbb{E}_\varphi[\|X - p^*\|]/v$



Light-load lower bound

Median

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

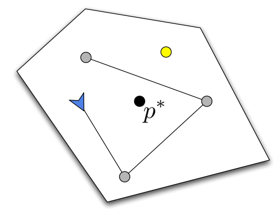


Lower bound (most useful when $\lambda \to 0^+$)

For all policies π: $\overline{T}_\pi \geq \mathbb{E}_\varphi[\|X - p^*\|]/v + \bar{s}$

Proof sketch:

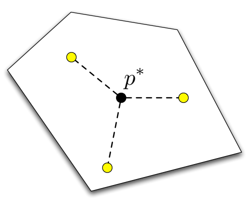
- $\overline{T} = \overline{W}_{\text{travel}} + \overline{W}_{\text{on-site}} + \bar{s}$.
- $\overline{W}_{\text{travel}} \geq \mathbb{E}_\varphi[\|X - p^*\|]/v$



Light-load lower bound

Median

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

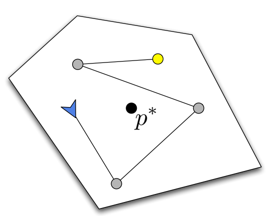


Lower bound (most useful when $\lambda \to 0^+$)

For all policies π: $\overline{T}_\pi \geq \mathbb{E}_\varphi[\|X - p^*\|]/v + \bar{s}$

Proof sketch:

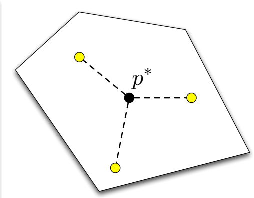
- $\overline{T} = \overline{W}_{\text{travel}} + \overline{W}_{\text{on-site}} + \bar{s}$.
- $\overline{W}_{\text{travel}} \geq \mathbb{E}_\varphi[\|X - p^*\|]/v$



Light-load lower bound

Median

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

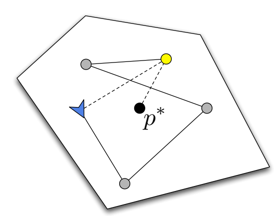


Lower bound (most useful when $\lambda \to 0^+$)

For all policies π: $\overline{T}_\pi \geq \mathbb{E}_\varphi[\|X - p^*\|]/v + \bar{s}$

Proof sketch:

- $\overline{T} = \overline{W}_{\text{travel}} + \overline{W}_{\text{on-site}} + \bar{s}$.
- $\overline{W}_{\text{travel}} \geq \mathbb{E}_\varphi[\|X - p^*\|]/v$



Heavy-load lower bound

Definition (Spatially-biased and -unbiased policies)

A policy π is said to be

1. *spatially unbiased* if system time is independent of demand location
2. *spatially biased* if system time depends on demand location

Heavy-load lower bound

$$\text{spatially-unbiased policies:}\quad \overline{T}_\pi \geq \frac{\beta_{\mathrm{TSP}}^2}{2} \frac{\lambda \left(\int_Q \varphi^{1/2}(x)dx \right)^2}{v^2\,(1-\varrho)^2} \qquad \text{as } \varrho \to 1^-$$

$$\text{spatially-biased policies:}\quad \overline{T}_\pi \geq \frac{\beta_{\mathrm{TSP}}^2}{2} \frac{\lambda \left(\int_Q \varphi^{2/3}(x)dx \right)^3}{v^2\,(1-\varrho)^2} \qquad \text{as } \varrho \to 1^-$$

Heavy-load lower bound

Definition (Spatially-biased and -unbiased policies)

A policy π is said to be

1. *spatially unbiased* if system time is independent of demand location
2. *spatially biased* if system time depends on demand location

Heavy-load lower bound

$$\text{spatially-unbiased policies:}\quad \overline{T}_\pi \geq \frac{\beta_{\mathrm{TSP}}^2}{2} \frac{\lambda \left(\int_Q \varphi^{1/2}(x)dx \right)^2}{v^2\,(1-\varrho)^2} \qquad \text{as } \varrho \to 1^-$$

$$\text{spatially-biased policies:}\quad \overline{T}_\pi \geq \frac{\beta_{\mathrm{TSP}}^2}{2} \frac{\lambda \left(\int_Q \varphi^{2/3}(x)dx \right)^3}{v^2\,(1-\varrho)^2} \qquad \text{as } \varrho \to 1^-$$

Proof sketch (for unbiased policies)

Proof of the lower bound:

- the idea is to use **stability arguments** (which are independent of policies!)
- let $\overline{D}$ be the travel **inter-demand** distance
- one can show that

$$\overline{D} \geq \beta_{\mathrm{TSP}} \frac{\int_Q \varphi^{1/2}(x)dx}{\sqrt{2\,\overline{N}}} \qquad \text{as } \varrho \to 1^-,$$

  with $\overline{N}$ average number of waiting demands
- for stability:

$$\bar{s} + \frac{\overline{D}}{v} \leq \frac{1}{\lambda} \qquad \Longrightarrow \qquad \bar{s} + \beta_{\mathrm{TSP}} \frac{\int_Q \varphi^{1/2}(x)dx}{v\,\sqrt{2\,\overline{N}}} \leq 1/\lambda$$

- since $\overline{N} = \lambda \overline{W}$ and $\overline{T} = \overline{W} + \bar{s}$ one obtains:

$$\overline{T}^* \geq \frac{\beta_{\mathrm{TSP}}^2}{2} \frac{\lambda \left(\int_Q \varphi^{1/2}(x)dx \right)^2}{v^2\,(1-\varrho)^2}$$

Lecture outline

D. J. Bertsimas and G. J. van Ryzin. A stochastic and dynamic vehicle routing problem in the Euclidean plane. *Operations Research*, 39:601–615, 1991

D. J. Bertsimas and G. J. van Ryzin. Stochastic and dynamic vehicle routing with general interarrival and service time distributions. *Advances in Applied Probability*, 25:947–978, 1993

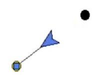
M. Pavone, E. Frazzoli, and F. Bullo. Distributed and adaptive algorithms for vehicle routing in a stochastic and dynamic environment. *IEEE Transactions on Automatic Control*, May 2010. (Submitted, Apr 2009) to appear

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

$$\lim_{\lambda \to 0^+} \overline{T}_{\text{SQM}} / \overline{T}^* = 1$$

Proof sketch

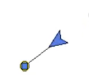
- As $\lambda \to 0^+$, $\mathbb{P}\,[\text{demand generated when system is empty}] \to 1$
- $\Rightarrow$ all demands generated with the vehicle at p^*
- $\Rightarrow \overline{T}_{\text{SQM}} = \mathbb{E}_{\varphi}[\|X - p^*\|]/v + \bar{s}$

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

$$\lim_{\lambda \to 0^+} \overline{T}_{\text{SQM}} / \overline{T}^* = 1$$

Proof sketch

- As $\lambda \to 0^+$, $\mathbb{P}\,[\text{demand generated when system is empty}] \to 1$
- $\Rightarrow$ all demands generated with the vehicle at p^*
- $\Rightarrow \overline{T}_{\text{SQM}} = \mathbb{E}_{\varphi}[\|X - p^*\|]/v + \bar{s}$

An optimal spatially-unbiased heavy-load policy

Unbiased TSP (UTSP)

Partition $\mathcal{Q}$ into r subregions $\mathcal{Q}_k$ with $\int_{\mathcal{Q}_k} \varphi(x)dx = 1/r$. Then:

1: within each subregion form sets of size n/r
2: deposit sets in a queue
3: service sets FCFS by following a TSP tour

Optimize over n.

Optimality of UTSP policy

$$\lim_{\varrho \to 1^-} \overline{T}_{\text{UTSP}}(r) / \overline{T}^*_{\text{U}} \leq 1 + 1/r$$

Proof

Proof ($r = 1$)

- idea: reduction to GI/G/1 queue
- jth set viewed as jth customer: arrival and service times are i.i.d.!
- inter-arrival distribution is Erlang of order n
- expected service time is $n\,\bar{s} + \beta_{\text{TSP}}\sqrt{n}\int_{\mathcal{Q}} \varphi^{1/2}(x)dx/v$
- standard results give upper bound on the wait in queue for a set
- then easy to find upper bound for individual demands

Comments

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 φ

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:

$$\frac{\lambda}{n}\Big(n\,\bar{s} + \beta_{\mathrm{TSP}}\sqrt{n}\int_{\mathcal{Q}} \varphi^{1/2}(x)dx/v\Big) < 1$$

- then one should **a priori** select:

$$n > \lambda^2 \beta_{\mathrm{TSP}}^2 \left[\int_{\mathcal{Q}} \varphi^{1/2}(x)\,dx\right]^2 / (v^2\,(1-\varrho)^2)$$

- $\Rightarrow$ wrong selection of n might lead to instability or unacceptable deterioration in performance

Divide & Conquer policy

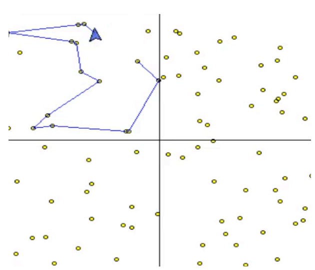
Divide & Conquer (DC)

Partition $\mathcal{Q}$ into r subregions $\mathcal{Q}_k$ with $\int_{\mathcal{Q}_k} \varphi(x)dx = 1/r$. Then:

1: while no customers, move to empirical median $\tilde{p}^*$

2: while customers waiting

1. move to subregion $\mathcal{Q}_k$
2. service all demands in $\mathcal{Q}_k$ by following a TSP tour
3. $k \leftarrow k+1$ (modulo r)



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

Implementation:

- NP-hard computation, but effective heuristics

Adaptation: the policy does not require knowledge of

1. vehicle velocity v, environment $\mathcal{Q}$
2. arrival rate λ
3. expected on-site service $\bar{s}$

Performance:

1. in light load, delay is optimal
2. in heavy load, delay is optimal
3. stable in any load condition

optimal and adaptive
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 "jth customer"
- focus on dynamical system

$$\mathbb{E}[n_{i+1}] \leq \lambda\, \mathbb{E}\Big[\sum_{q=1}^{n_i} s_q + \mathrm{TSP}(n_i)\Big]$$
$$\leq \lambda \left(\bar{s}\; \mathbb{E}[n_i] + \beta_{\mathrm{TSP}} \int_{\mathcal{Q}} \varphi^{1/2}(x)dx\, \sqrt{\mathbb{E}[n_i]}/v \right)$$

- upper bound trajectories with the trajectories of **virtual** dynamical system

$$z_{i+1} = \varrho\, z_i + (\lambda/v)\, \beta_{\mathrm{TSP}} \int_{\mathcal{Q}} \varphi^{1/2}(x)dx\, \sqrt{z_i}$$

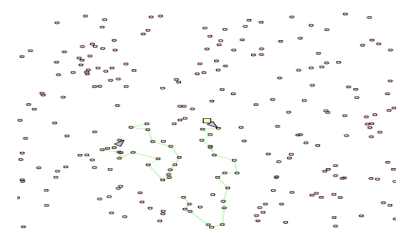
- $\overline{T}_{\mathrm{DC}} \leq \lim_{i \to +\infty} z_i/\lambda$

Receding-Horizon policy

Receding-Horizon (RH)

For $\eta \in (0, 1]$, single agent performs:

```
while no customers, move to empirical median p̃*
while customers waiting
   1 compute TSP tour through current demands
   2 service η-fraction of path
```



RH policy

Implementation:

- NP-hard computation, but effective heuristics

Adaptation: the policy does not require knowledge of

1. vehicle velocity v, environment $\mathcal{Q}$
2. arrival rate λ and spatial density function φ
3. expected on-site service $\bar{s}$

Performance:

1. in light load, delay is optimal
2. in heavy load, delay is within a multiplicative factor from optimal
3. multiplicative factor depends upon φ and is conjectured to equal 2

adaptive to all parameters

Lecture outline

Dynamic Vehicle Routing for Robotic Networks
Lecture #4: The multi-vehicle DVR problem

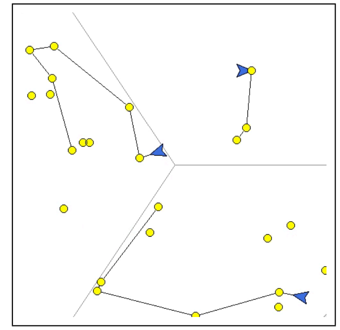
Francesco Bullo[1] Emilio Frazzoli[2] Marco Pavone[2]
Ketan Savla[2] Stephen L. Smith[2]

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

[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

Load balancing in DVR via territory partitioning



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

Lecture outline

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

Territory partitioning is ... art

Ocean Park Paintings, by Richard Diebenkorn (1922-1993)

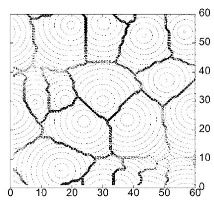
Territory partitioning: optimality and behaviors

DESIGN of performance metrics

1. how to cover a region with n minimum-radius overlapping disks?
2. how to design a minimum-distortion (fixed-rate) vector quantizer?
3. 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?



4. what if each robot goes to "center" of own dominance region?
5. what if each robot moves away from closest vehicle?

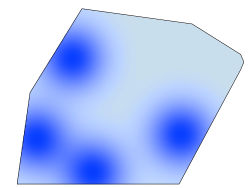
Multi-center functions

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 $v = \{v_1, \ldots, v_n\}$

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



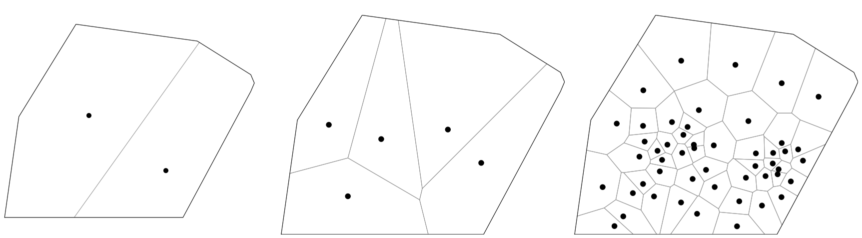
- $\varphi : \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$ density
- $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$ penalty function

F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks.* Applied Mathematics Series. Princeton University Press, 2009. Available at http://www.coordinationbook.info

Optimal partitioning

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

$$V_i(p) = \{x \in \mathcal{Q} | \ \|x - p_i\| \leq \|x - p_j\|, \ \forall j \neq i\}$$
$$= \mathcal{Q} \bigcap_j (\text{half plane between } i \text{ and } j, \text{ containing } i)$$

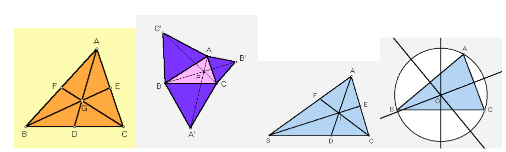


Descartes 1644, Dirichlet 1850, Voronoi 1908, Thiessen 1911, Fortune 1986 (sweepline algorithm $O(n \log(n))$)

Optimal centering (for region v with density φ)

function of p	minimizer = center
$p \mapsto \int_v \|x - p\| \varphi(x) dx$	**median** (or **Fermat–Weber point**)
$p \mapsto \int_v \|x - p\|^2 \varphi(x) dx$	**centroid** (or **center of mass**)
$p \mapsto \text{area}(v \cap \text{disk}(p, r))$	**r-area center**
$p \mapsto$ radius of largest disk centered at p enclosed inside v	**incenter**
$p \mapsto$ radius of smallest disk centered at p enclosing v	**circumcenter**

From online Encyclopedia of Triangle Centers



How to compute the median of a convex set

For convex planar set $\mathcal{Q}$ with strictly positive density φ,

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

1. $\mathcal{H}_{\text{FW}}$ is strictly convex
2. the global minimum point is in $\mathcal{Q}$ and is called *median* of $\mathcal{Q}$
3. compute median via gradient flow with

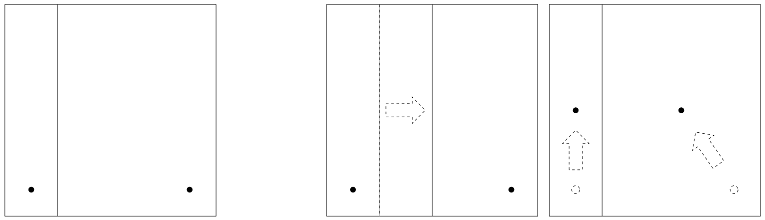
$$\frac{d}{dp}\mathcal{H}_{\text{FW}}(p) = \int_{\mathcal{Q}} \frac{p - x}{\|p - x\|} \varphi(x) dx$$

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

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

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

$$\frac{\partial \mathcal{H}}{\partial p_i}(p) = \int_{V_i(p)} \frac{\partial}{\partial p_i} f\left(\|x - p_i\|\right) \varphi(x) dx + \int_{\partial V_i(p)} f\left(\|x - p_i\|\right) \langle n_i(x), \frac{\partial x}{\partial p_i} \rangle \varphi(x) dx$$

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

$$\begin{aligned}\frac{\partial \mathcal{H}}{\partial p_i}(p) = &\int_{V_i(p)} \frac{\partial}{\partial p_i} f\left(\|x - p_i\|\right)\varphi(x)dx \\ &+ \int_{\partial V_i(p)} f\left(\|x - p_i\|\right)\langle n_i(x), \frac{\partial x}{\partial p_i}\rangle\varphi(x)dx \\ &+ \underbrace{\sum_{j \text{ neigh } i} \int_{\partial V_j(p)\cap\partial V_i(p)} f\left(\|x - p_j\|\right)\langle n_{ji}(x), \frac{\partial x}{\partial p_i}\rangle\varphi(x)dx}_{\text{contrib from neighbors}}\end{aligned}$$

Gradient algorithm for multicenter function

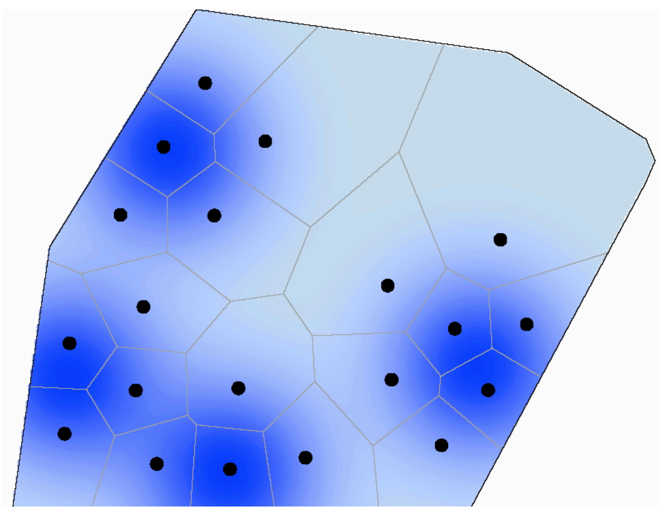
After assuming v is Voronoi partition,

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

For f smooth, note simplifications for boundary terms

$$\begin{aligned}\frac{\partial \mathcal{H}}{\partial p_i}(p) = &\int_{V_i(p)} \frac{\partial}{\partial p_i} f\left(\|x - p_i\|\right)\varphi(x)dx \\ &+ \int_{\partial V_i(p)} f\left(\|x - p_i\|\right)\langle n_i(x), \frac{\partial x}{\partial p_i}\rangle\varphi(x)dx \\ &- \int_{\partial V_i(P)} f\left(\|x - p_i\|\right)\langle n_i(x), \frac{\partial x}{\partial p_i}\rangle\varphi(x)dx\end{aligned}$$

Example optimal partition



Lecture outline

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

D. J. Bertsimas and G. J. van Ryzin. Stochastic and dynamic vehicle routing with general interarrival and service time distributions. *Advances in Applied Probability*, 25:947–978, 1993

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**

Light-load lower bound

Multi - Median

- minimizer $p^* = \{p_1^*, \dots, p_m^*\}$ of

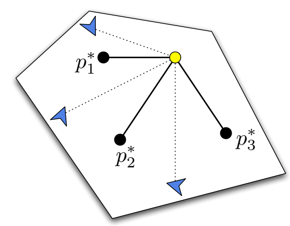
$$p \mapsto \mathbb{E}_\varphi[\min_i \|X - p_i\|] = \sum_{i=1}^{m} \int_{V_i} \|x - p_i\| \varphi(x) dx$$

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

For all policies π: $\overline{T}_\pi \geq \mathbb{E}_\varphi[\min_i \|X - p_i^*\|]/v + \bar{s}$

Proof sketch:

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



Heavy-load lower bound

Heavy-load lower bound

spatially-unbiased policies: $$\overline{T}_\pi \geq \frac{\beta_{\mathrm{TSP}}^2}{2} \frac{\lambda \left(\int_Q \varphi^{1/2}(x) dx\right)^2}{m^2 \, v^2 \, (1-\varrho)^2} \quad \text{as } \varrho \to 1^-$$

spatially-biased policies: $$\overline{T}_\pi \geq \frac{\beta_{\mathrm{TSP}}^2}{2} \frac{\lambda \left(\int_Q \varphi^{2/3}(x) dx\right)^3}{m^2 \, v^2 \, (1-\varrho)^2} \quad \text{as } \varrho \to 1^-$$

Proof sketch (for unbiased policies):

- Recall inter-demand distance $\overline{D} \geq \beta_{\mathrm{TSP}} \frac{\int_Q \varphi^{1/2}(x) dx}{\sqrt{2 \overline{N}}}$, as $\varrho \to 1^-$
- for stability with m vehicles:

$$\bar{s} + \frac{\overline{D}}{v} \leq \frac{m}{\lambda} \quad \Longrightarrow \quad \bar{s} + \beta_{\mathrm{TSP}} \frac{\int_Q \varphi^{1/2}(x) dx}{v \sqrt{2 \overline{N}}} \leq m/\lambda$$

- $\overline{N} = \lambda \overline{W}$ and $\overline{T} = \overline{W} + \bar{s} \quad \Longrightarrow \quad \overline{T}^* \geq \frac{\beta_{\mathrm{TSP}}^2}{2} \frac{\lambda \left(\int_Q \varphi^{1/2}(x) dx\right)^2}{m^2 \, v^2 \, (1-\varrho)^2}$

Heavy-load lower bound

Heavy-load lower bound

spatially-unbiased policies: $$\overline{T}_\pi \geq \frac{\beta_{\mathrm{TSP}}^2}{2} \frac{\lambda \left(\int_Q \varphi^{1/2}(x) dx\right)^2}{m^2 \, v^2 \, (1-\varrho)^2} \quad \text{as } \varrho \to 1^-$$

spatially-biased policies: $$\overline{T}_\pi \geq \frac{\beta_{\mathrm{TSP}}^2}{2} \frac{\lambda \left(\int_Q \varphi^{2/3}(x) dx\right)^3}{m^2 \, v^2 \, (1-\varrho)^2} \quad \text{as } \varrho \to 1^-$$

Proof sketch (for unbiased policies):

- Recall inter-demand distance $\overline{D} \geq \beta_{\mathrm{TSP}} \frac{\int_Q \varphi^{1/2}(x) dx}{\sqrt{2 \overline{N}}}$, as $\varrho \to 1^-$
- for stability with m vehicles:

$$\bar{s} + \frac{\overline{D}}{v} \leq \frac{m}{\lambda} \quad \Longrightarrow \quad \bar{s} + \beta_{\mathrm{TSP}} \frac{\int_Q \varphi^{1/2}(x) dx}{v \sqrt{2 \overline{N}}} \leq m/\lambda$$

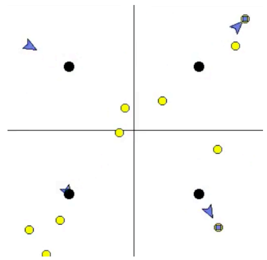
- $\overline{N} = \lambda \overline{W}$ and $\overline{T} = \overline{W} + \bar{s} \quad \Longrightarrow \quad \overline{T}^* \geq \frac{\beta_{\mathrm{TSP}}^2}{2} \frac{\lambda \left(\int_Q \varphi^{1/2}(x) dx\right)^2}{m^2 \, v^2 \, (1-\varrho)^2}$

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 vehicles service demands in FCFS order
3: each vehicle returns to p_k^* after each service is completed



Proof sketch of optimality

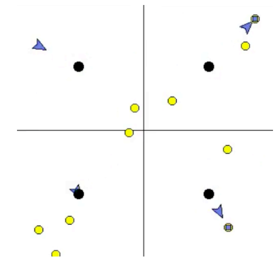
- As $\lambda \to 0^+$, $\mathbb{P}$[demand generated when system is empty] $\to 1$
- $\Rightarrow$ all demands are generated with the vehicles at p^*

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 vehicles service demands in FCFS order
3: each vehicle returns to p_k^* after each service is completed



Proof sketch of optimality

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

An optimal spatially-unbiased heavy-load policy

Unbiased TSP (UTSP)

Partition $\mathcal{Q}$ into r subregions $\mathcal{Q}_k$ with $\int_{\mathcal{Q}_k} \varphi(x)dx = 1/r$. 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

$$\lim_{\varrho \to 1^-} \overline{T}_{\mathrm{UTSP}}(r)/\overline{T}_{\mathrm{U}}^* \leq 1 + 1/r$$

Proof sketch of optimality (r=1)

- reduction to GI/G/m

An optimal spatially-unbiased heavy-load policy

Unbiased TSP (UTSP)

Partition $\mathcal{Q}$ into r subregions $\mathcal{Q}_k$ with $\int_{\mathcal{Q}_k} \varphi(x)dx = 1/r$. 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

$$\lim_{\varrho \to 1^-} \overline{T}_{\mathrm{UTSP}}(r)/\overline{T}_{\mathrm{U}}^* \leq 1 + 1/r$$

Proof sketch of optimality (r=1)

- reduction to GI/G/m

Lecture outline

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

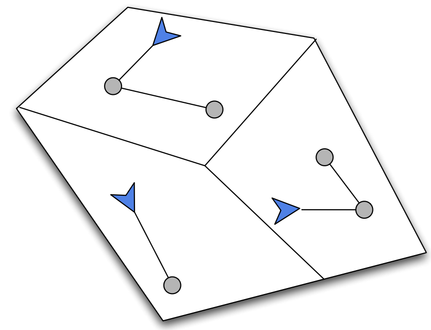
M. Pavone, E. Frazzoli, and F. Bullo. Distributed and adaptive algorithms for vehicle routing in a stochastic and dynamic environment. *IEEE Transactions on Automatic Control*, May 2010. (Submitted, Apr 2009) to appear

Partitioning policies

Definition (π-partitioning policy)

Given m vehicles and single-vehicle policy π:

1. Workspace divided into m subregions
2. One-to-one correspondence vehicles/subregions
3. Each agent executes the single-vehicle policy π within its own subregion



Motivation

Performance:

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

Motivation

Performance:

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

Motivation

Performance:

- light load: problem reduces to locational optimization
- heavy load:
  1. delay of optimal single vehicle policy scales as $\lambda\,|\mathcal{Q}|$
  2. by (equitably) partitioning, delay reduces to $\frac{\lambda}{m}\,\frac{|\mathcal{Q}|}{m} = \frac{\lambda|\mathcal{Q}|}{m^2}$
  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 $\propto \lambda \int_{\mathcal{Q}_k} \varphi(x)dx$
- per-vehicle service capacity is $\propto \lambda \int_{\mathcal{Q}_k} \varphi^{1/2}(x)dx$
- optimal partitioning = **equalizing** per-vehicle workload **and** service capacity

Definition

A partition $\{\mathcal{Q}_k\}_{k=1}^m$ is:

- equitable if $\int_{\mathcal{Q}_k} \varphi(x)dx = \int_{\mathcal{Q}} \varphi(x)dx/m$
- simultaneously equitable if
  1. $\int_{\mathcal{Q}_k} \varphi(x)dx = \int_{\mathcal{Q}} \varphi(x)dx/m$, and
  2. $\int_{\mathcal{Q}_k} \varphi^{1/2}(x)dx = \int_{\mathcal{Q}} \varphi^{1/2}(x)dx/m$

Simultaneously equitable partitions exist for any $\mathcal{Q}$ and φ

(S. Bespamyatnikh, D. Kirkpatrick, and J. Snoeyink, 2000)

Optimal partitioning in heavy load

Intuition

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

Definition

A partition $\{\mathcal{Q}_k\}_{k=1}^m$ is:

- equitable if $\int_{\mathcal{Q}_k} \varphi(x)dx = \int_{\mathcal{Q}} \varphi(x)dx/m$
- simultaneously equitable if
  1. $\int_{\mathcal{Q}_k} \varphi(x)dx = \int_{\mathcal{Q}} \varphi(x)dx/m$, and
  2. $\int_{\mathcal{Q}_k} \varphi^{1/2}(x)dx = \int_{\mathcal{Q}} \varphi^{1/2}(x)dx/m$

Simultaneously equitable partitions exist for any $\mathcal{Q}$ and φ

(S. Bespamyatnikh, D. Kirkpatrick, and J. Snoeyink, 2000)

Optimal partitioning in heavy load

Intuition

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

Definition

A partition $\{\mathcal{Q}_k\}_{k=1}^m$ is:

- equitable if $\int_{\mathcal{Q}_k} \varphi(x)dx = \int_{\mathcal{Q}} \varphi(x)dx/m$
- simultaneously equitable if
  1. $\int_{\mathcal{Q}_k} \varphi(x)dx = \int_{\mathcal{Q}} \varphi(x)dx/m$, and
  2. $\int_{\mathcal{Q}_k} \varphi^{1/2}(x)dx = \int_{\mathcal{Q}} \varphi^{1/2}(x)dx/m$

Simultaneously equitable partitions exist for any $\mathcal{Q}$ and φ

(S. Bespamyatnikh, D. Kirkpatrick, and J. Snoeyink, 2000)

Optimal partitioning in heavy load

Theorem

Given single-vehicle optimal policy π^*, a π^*-partitioning policy using a simultaneously equitable partition is an optimal unbiased policy

Proof sketch

- $\mathbb{P}\,[\text{demand arrives in } \mathcal{Q}_k] = \int_{\mathcal{Q}_k} \varphi(x)\, dx = 1/m$
- arrival rate in region k: $\lambda_k = \lambda/m$
- $\Rightarrow \varrho_k = \lambda_k \bar{s} = \lambda \bar{s}/m = \varrho < 1 \Rightarrow$ system is stable
- conditional density for region k: $\varphi(x)/\left(\int_{\mathcal{Q}_k} \varphi(x)\, dx\right) = m\, \varphi(x)$
- $\overline{T} = \sum_{k=1}^{m} \left(\int_{\mathcal{Q}_k} \varphi(x)\, dx \ \frac{\beta_{\mathrm{TSP}}^2}{2} \frac{\lambda_k}{v^2\,(1-\varrho_k)^2} \left[\int_{\mathcal{Q}_k} \sqrt{\frac{\varphi(x)}{\int_{\mathcal{Q}_k} \varphi(x)\, dx}} dx \right]^2 \right)$
  $= \sum_{k=1}^{m} \ \frac{1}{m}\, \overline{T}_{\pi^*} \frac{1}{m^2}$

Comments

If $\{\mathcal{Q}_k\}_{k=1}^m$ 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 $\{\mathcal{Q}_k\}_{k=1}^m$ is only equitable wrt to φ...

- per-vehicle service capacity is unbalanced $\Rightarrow$ policy stable but **not optimal**
- guaranteed to be within m of optimal unbiased performance

Comments

If $\{\mathcal{Q}_k\}_{k=1}^m$ 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 $\{\mathcal{Q}_k\}_{k=1}^m$ is only equitable wrt to φ...

- per-vehicle service capacity is unbalanced $\Rightarrow$ policy stable but **not optimal**
- guaranteed to be within m of optimal unbiased performance

FB, EF, MP, KS, SLS (UCSB, MIT) Dynamic Vehicle Routing (Lecture 4/8) 29jun10 @ Baltimore, ACC 24 / 27

Special cases

Case $\bar{s} = 0$:

- stability not an issue:

$$\underbrace{\lambda}_{\text{generation rate}} - \underbrace{m \cdot \frac{n}{\text{TSPlength}(n)}}_{\text{service rate}} = \text{demand growth rate}$$

- since $\text{TSPlength}(n) \propto \sqrt{n} \Rightarrow$ stability for all λ, 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

Special cases

Case $\bar{s} = 0$:

- stability not an issue:

$$\underbrace{\lambda}_{\text{generation rate}} \quad - \quad \underbrace{m \cdot \frac{n}{\mathrm{TSPlength}(n)}}_{\text{service rate}} \quad = \quad \text{demand growth rate}$$

- since $\mathrm{TSPlength}(n) \propto \sqrt{n} \Rightarrow$ stability for all λ, 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

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*	
11:00-11:45pm	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

Dynamic Vehicle Routing for Robotic Networks Lecture #5: Extensions to vehicle networks and distributed algorithms

Francesco Bullo[1] Emilio Frazzoli[2] Marco Pavone[2]
Ketan Savla[2] Stephen L. Smith[2]

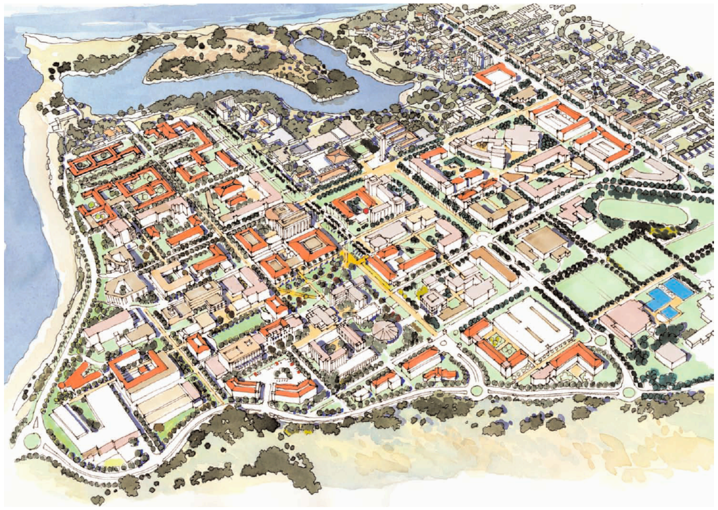
[1]CCDC
University of California, Santa Barbara
bullo@engineering.ucsb.edu

[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

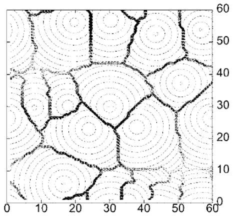
Territory partitioning via *centralized space planning*



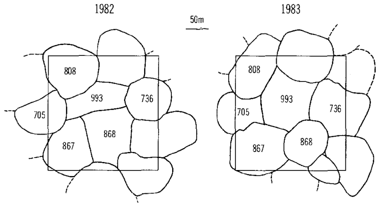
UCSB Campus Development Plan, 2008

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

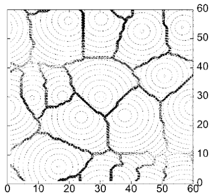
Territory partitioning: behaviors and optimality

DESIGN of performance metrics

1. how to cover a region with n minimum-radius overlapping disks?
2. how to design a minimum-distortion (fixed-rate) vector quantizer?
3. 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?



4. what if each robot goes to "center" of own dominance region?
5. what if each robot moves away from closest vehicle?

Lecture outline

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

References

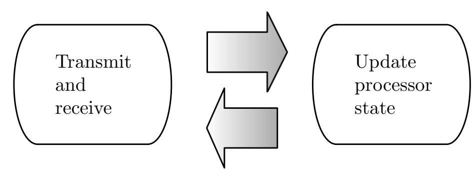
1. I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999
2. N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1997
3. D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997
4. S. Martínez, F. Bullo, J. Cortés, and E. Frazzoli. On synchronous robotic networks – Part I: Models, tasks and complexity. *IEEE Transactions on Automatic Control*, 52(12):2199–2213, 2007

Objective

1. meaningful + tractable model
2. 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



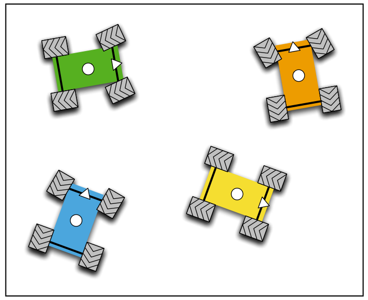
Distributed algorithm for a network of processors consists of

1. $W^{[i]}$, the **processor state set**
2. $\mathbb{A}$, the **communication alphabet**
3. $\mathrm{stf}^{[i]} : W^{[i]} \times \mathbb{A}^n \to W^{[i]}$, the **state-transition map**
4. $\mathrm{msg}^{[i]} : W^{[i]} \to \mathbb{A}$, the **message-generation map**

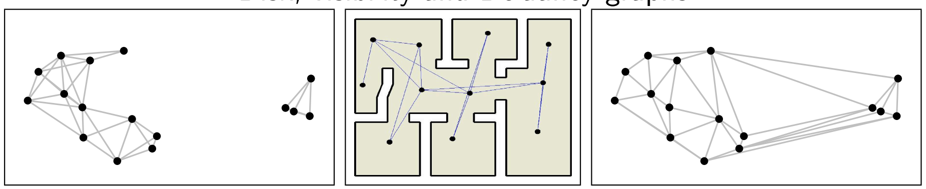
Robotic network

A **robotic network** is

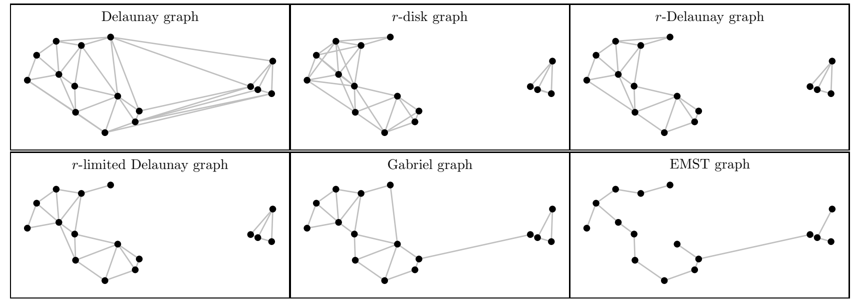
1. **set of robots** moving in space $\mathcal{Q}$
2. **interaction graph**



Disk, visibility and Delauney graphs



Communication models for robotic networks



Relevant graphs

1. fixed, directed, balanced
2. switching
3. proximity/geometric or state-dependent
4. random, random geometric (packet losses)

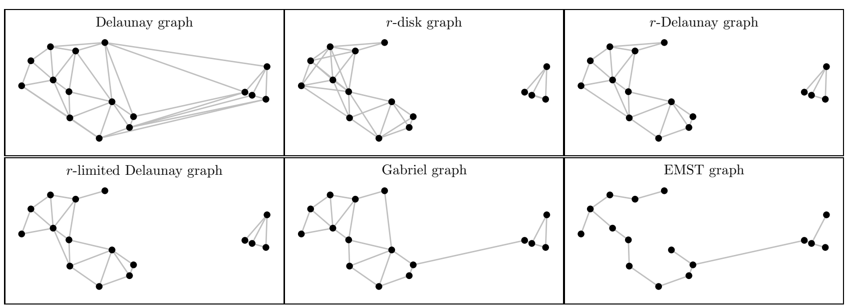
Message model

1. message
2. packet/bits

Sensing model

1. absolute coords other robots
2. absolute coords environment boundary

Communication models for robotic networks



Relevant graphs

1. fixed, directed, balanced
2. switching
3. proximity/geometric or state-dependent
4. random, random geometric (packet losses)

Message model

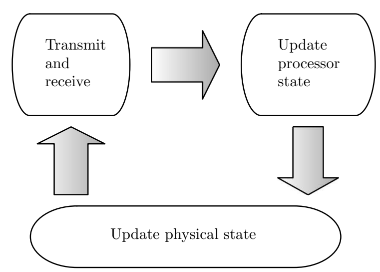
1. message
2. packet/bits

Sensing model

1. absolute coords other robots
2. absolute coords environment boundary

Synchronous control and communication

1. **communication schedule** $\mathbb{T} = \{t_\ell\}_{\ell \in \mathbb{N}_0} \subset \mathbb{R}_{\geq 0}$
2. **communication alphabet** $\mathbb{A}$
3. set of values for **processor vars** W
4. **message-generation function** $\text{msg} : \mathbb{T} \times \mathcal{Q} \times W \to \mathbb{A}$
5. **state-transition functions** $\text{stf} : \mathbb{T} \times W \times \mathbb{A}^N \to W$
6. **control function** $\text{ctrl} : \mathbb{R}_{\geq 0} \times \mathcal{Q} \times W \times \mathbb{A}^N \to U$



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

M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo. Equitable partitioning policies for mobile robotic networks. *IEEE Transactions on Automatic Control*, 2010. (Submitted Dec 2008 and Aug 2009) to appear

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

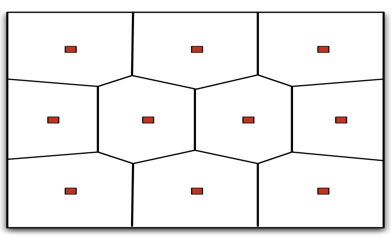
Spatially-distributed policies for DVR

Key idea

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

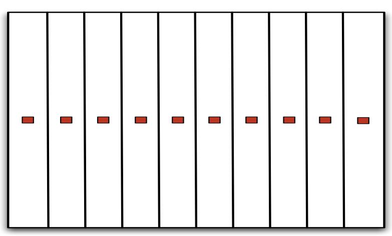
Light load

Optimal pre-positioning
⇒ median Voronoi diagrams



Heavy load

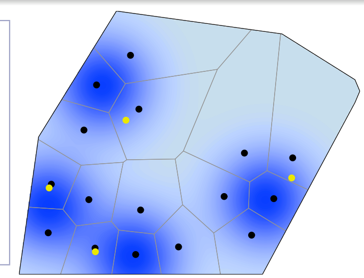
Workload balance
⇒ equitable partitions

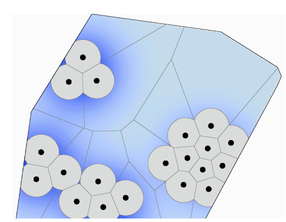


Median Voronoi diagrams (and beyond) with synchronous proximity-graphs communication

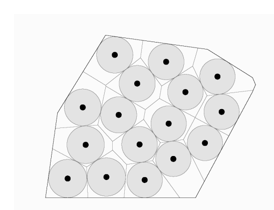
Voronoi+centering law

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

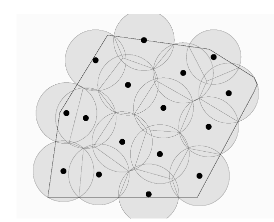




Area-center



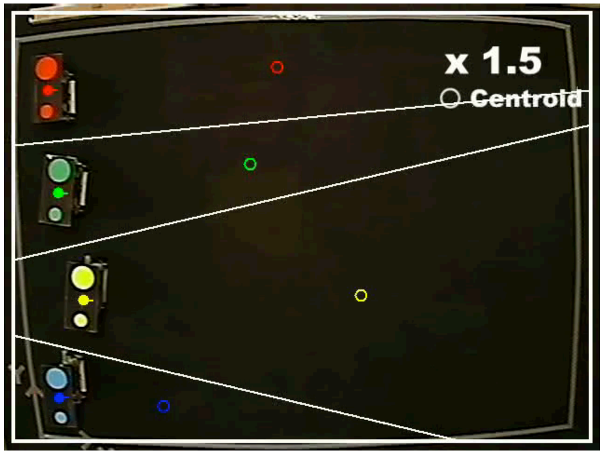
Incenter



Circumcenter

S. Martínez, J. Cortés, and F. Bullo. Motion coordination with distributed information. *IEEE Control Systems Magazine*, 27(4):75–88, 2007

Experimental Partitioning

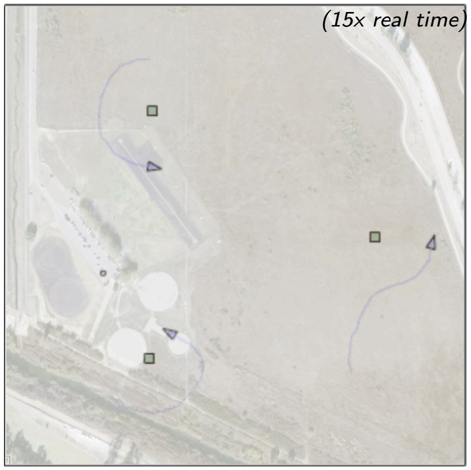


Takahide Goto, Takeshi Hatanaka, Masayuki Fujita
Tokyo Institute of Technology

Experimental Partitioning

Mac Schwager, Brian Julian, Daniela Rus
Distributed Robots Laboratory, MIT

Hardware-in-the-loop Partitioning and DVR for UAVs



John J. Enright, Chung Hsieh, Emilio Frazzoli
ARES Group, MIT and UCLA

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 $\{V_1, \dots, V_m\}$ generated by points $(p_1, \dots, p_m)$:

$$V_i = \{x \in Q \mid \|x - p_i\|^2 \leqslant \|x - p_j\|^2, \forall j \neq i\}$$

In general, an equitable Voronoi Diagram fails to exist...

Equitable and median Voronoi diagrams with synchronous proximity-graphs communication

"Ambitious" goal:

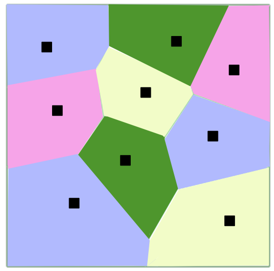
Distributed algorithm to partition the workspace according to:

1. median Voronoi diagram (relevant in light-load)
2. equitable (relevant in heavy load)

Voronoi Diagrams

Voronoi partition $\{V_1, \dots, V_m\}$ generated by points $(p_1, \dots, p_m)$:

$$V_i = \{x \in \mathcal{Q} \mid \|x - p_i\|^2 \leqslant \|x - p_j\|^2, \forall j \neq i\}$$



In general, an equitable Voronoi Diagram fails to exist...

Equitable and median Voronoi diagrams with synchronous proximity-graphs communication

"Ambitious" goal:

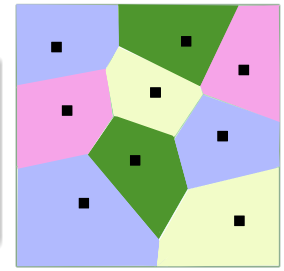
Distributed algorithm to partition the workspace according to:

1. median Voronoi diagram (relevant in light-load)
2. equitable (relevant in heavy load)

Voronoi Diagrams

Voronoi partition $\{V_1, \dots, V_m\}$ generated by points $(p_1, \dots, p_m)$:

$$V_i = \{x \in \mathcal{Q} \mid \|x - p_i\|^2 \leqslant \|x - p_j\|^2, \forall j \neq i\}$$



In general, an equitable Voronoi Diagram fails to exist...

Partitioning using Power Diagrams

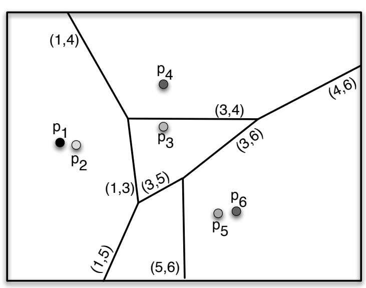
Power distance

- $p = (p_1, \dots, p_m)$ collection of points in $\mathcal{Q} \subset \mathbb{R}^2$
- 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 $\{V_1, \dots, V_m\}$ generated by weighted points $\big((p_1, w_1), \dots, (p_m, w_m)\big)$:

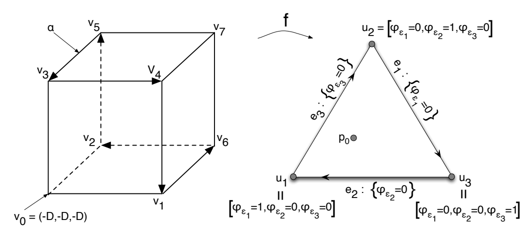
- $V_i = \{x \in \mathcal{Q} \mid \|x - p_i\|^2 - w_i \leqslant \|x - p_j\|^2 - w_j, \forall j \neq i\}$



Existence theorem for Power diagrams

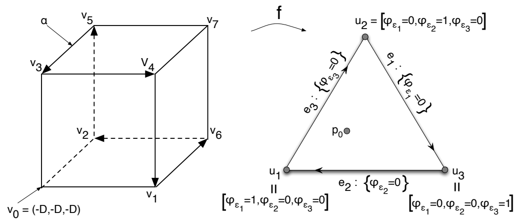
Existence theorem

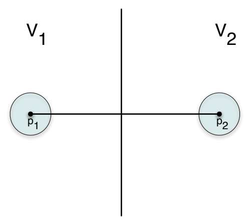
Let $p = (p_1, \dots, p_m)$ be the positions of $m \geq 1$ distinct points in $\mathcal{Q}$. Then there exist weights $(w_1, \dots, w_m)$ such that the corresponding Power diagram is equitable with respect to φ



Existence theorem

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

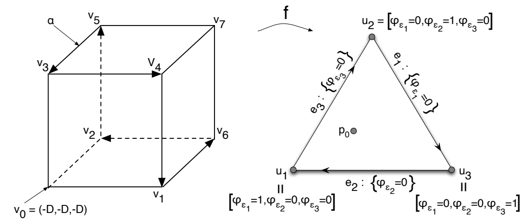


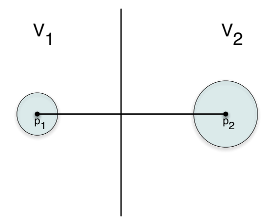


Existence theorem for Power diagrams

Existence theorem

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



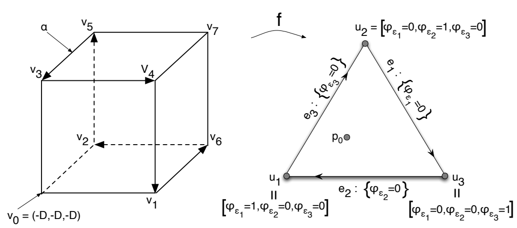


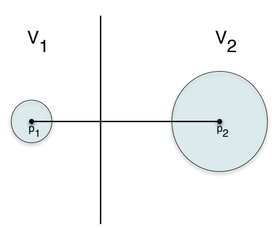
FB, EF, MP, KS, SLS (UCSB, MIT) Dynamic Vehicle Routing (Lecture 5/8) 29jun10 @ Baltimore, ACC 20 / 44

Existence theorem for Power diagrams

Existence theorem

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



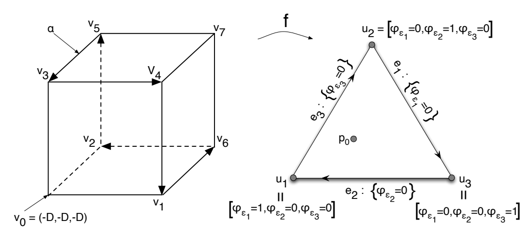


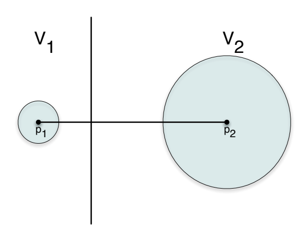
FB, EF, MP, KS, SLS (UCSB, MIT) Dynamic Vehicle Routing (Lecture 5/8) 29jun10 @ Baltimore, ACC 20 / 44

Existence theorem for Power diagrams

Existence theorem

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





FB, EF, MP, KS, SLS (UCSB, MIT) Dynamic Vehicle Routing (Lecture 5/8) 29jun10 @ Baltimore, ACC 20 / 44

Gradient descent law for equitable partitioning

- w_i locally controlled by vehicle i
- locational optimization function

$$\mathcal{H}(w) \doteq \sum_{i=1}^{m}\Big(\int_{V_i(w)} \varphi(x)dx\Big)^{-1} = \sum_{i=1}^{m} |V_i(w)|_{\varphi}^{-1}$$

- spatially-distributed gradient: $\frac{\partial \mathcal{H}}{\partial w_i} = \sum_{j \in N_i} \alpha_{ij}^{\varphi}\Big(\frac{1}{|V_j|_{\varphi}^2} - \frac{1}{|V_i|_{\varphi}^2}\Big)$

Gradient law for equitable partitioning

```
At each comm round:
1: acquire neighbors' positions
2: compute own dominance region
```
3: $w_i \leftarrow w_i - \gamma \frac{\partial \mathcal{H}}{\partial w_i}$

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

$$\dot{w}_i = -\frac{\partial \mathcal{H}}{\partial w_i} - w_i, \qquad \frac{\partial \mathcal{H}}{\partial p_i} \cdot \dot{p}_i - \frac{\partial \mathcal{H}}{\partial w_i} w_i = 0$$

Motion toward the median:

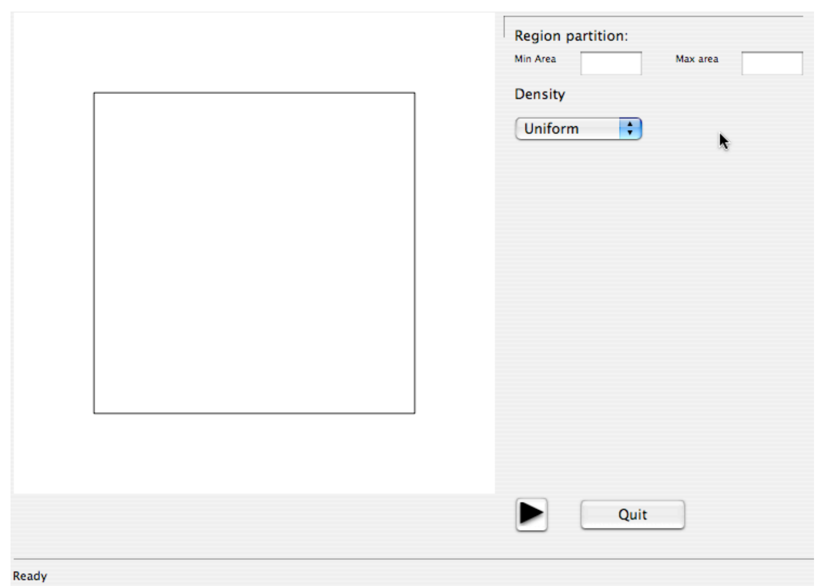
- basic idea: add a term that enforces computation of the median
- gradient term for computation of the median:

$$\frac{\partial \mathcal{H}_{\mathrm{FW}}}{\partial p_i} = \int_{V_i} \frac{p_i - x}{\|p_i - x\|}\varphi(x)dx$$

- modify the gradient descent law as

$$\dot{w}_i = -\frac{\partial \mathcal{H}}{\partial w_i}, \qquad \dot{p}_i = \frac{\partial \mathcal{H}_{\mathrm{FW}}}{\partial p_i}\ \psi\Big(\frac{\partial \mathcal{H}}{\partial p_i}, \frac{\partial \mathcal{H}_{\mathrm{FW}}}{\partial p_i}\Big)$$

Simulation



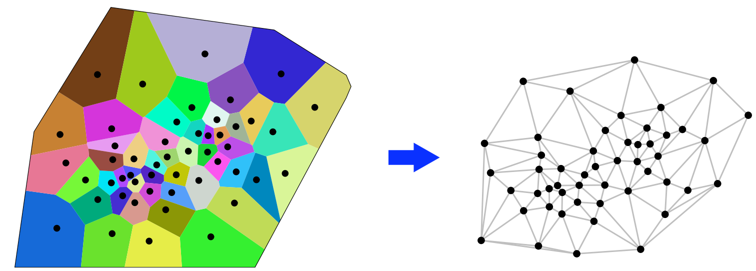
Lecture outline

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

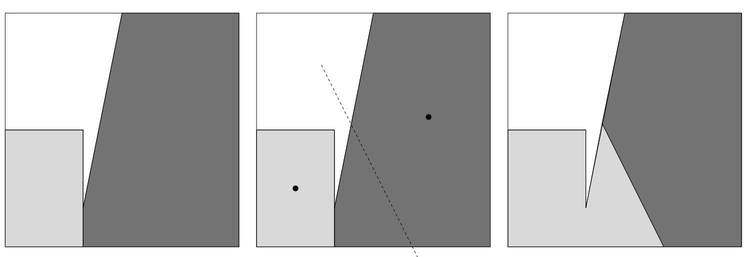


Minimalist coordination

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

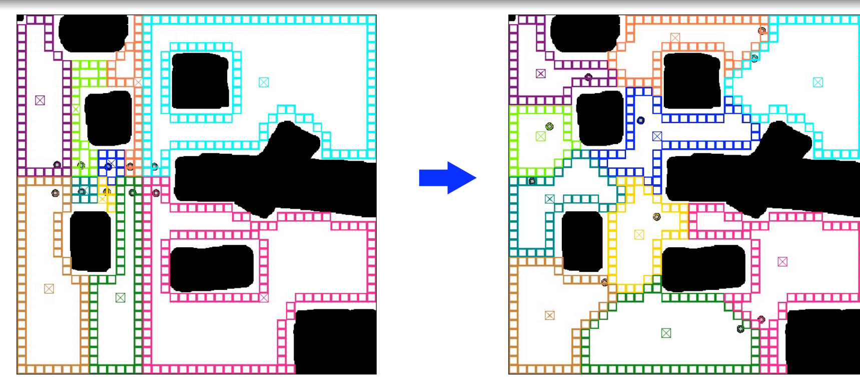
Gossip (asynchronous pair-wise) partitioning policy

1. Random communication between two regions
2. Compute two centers
3. Compute bisector of centers
4. Partition two regions by bisector



F. Bullo, R. Carli, and P. Frasca. Gossip coverage control for robotic networks: Dynamical systems on the the space of partitions. *SIAM Review*, January 2010. Submitted

Indoor example implementation



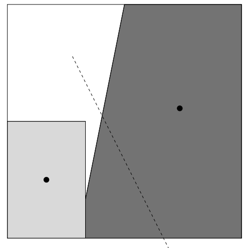
- Player/Stage platform
- realistic robot models in discretized environments
- integrated wireless network model & obstacle-avoidance planner

J. W. Durham, R. Carli, P. Frasca, and F. Bullo. Discrete partitioning and coverage control with gossip communication. In *ASME Dynamic Systems and Control Conference*, Hollywood, CA, October 2009

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

Lyapunov function for peer-to-peer territory partitioning

$$\mathcal{H}(v) = \sum_{i=1}^{n} \int_{v_i} f(\| \mathrm{center}(v_i) - q \|)\phi(q)dq$$

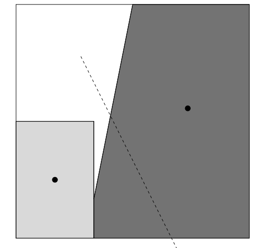


1. state space is not finite-dimensional
   - non-convex disconnected polygons
   - arbitrary number of vertices
2. peer-to-peer map is not deterministic, ill-defined and discontinuous
   - two regions could have same centers

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

Lyapunov function for peer-to-peer territory partitioning

$$\mathcal{H}(v) = \sum_{i=1}^{n} \int_{v_i} f(\| \mathrm{center}(v_i) - q \|)\phi(q)dq$$



1. state space is not finite-dimensional
   - non-convex disconnected polygons
   - arbitrary number of vertices
2. 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 ℓ, the set v is collections of n subsets of Q, $\{v_1, \dots, v_n\}$, such that

1. $v_1 \cup \dots \cup v_n = Q$,
2. $\mathrm{interior}(v_i) \cap \mathrm{interior}(v_j) = \emptyset$ if $i \neq j$, and
3. each v_i is union of ℓ convex sets

Given sets A and B, symmetric distance is:

$$d_\Delta(A, B) = \mathrm{area}\Big((A \cup B) \setminus (A \cap B)\Big)$$

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:

1. $W \subset X$ compact and positively invariant for each T_i
2. $U : W \to \mathbb{R}$ decreasing along each T_i
3. U and T_i are continuous on W
4. there exists probability $p \in]0, 1[$ such that, for all indices $i \in I$ and times ℓ, we have $\quad \mathrm{Prob}\left[x_{\ell+1} = T_i(x_\ell) \,|\, \mathrm{past}\right] \geq p$

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

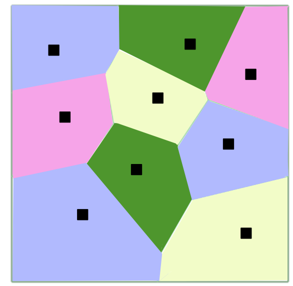
A. Arsie, K. Savla, and E. Frazzoli. Efficient routing algorithms for multiple vehicles with no explicit communications. *IEEE Transactions on Automatic Control*, 54(10):2302–2317, 2009

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

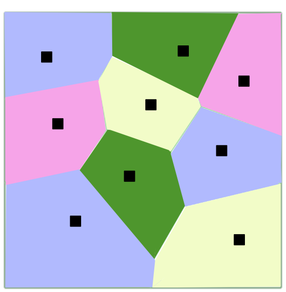
- Explicit agent-to-agent communication
- Needs knowledge of φ

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 φ

Partitioning with no explicit inter-vehicle communication

Inspiration: Distributed MacQueen algorithm

- Pick any m generator points $(p_1, \dots, p_m) \in \mathcal{Q}^m$
- Iteratively sample points q_j according to probability density function φ
- At each iteration j:
  - Assign the sampled point to the nearest generator $i^*(q_j) \in \{1, \dots, m\}$
  - update the position of generator i^* as

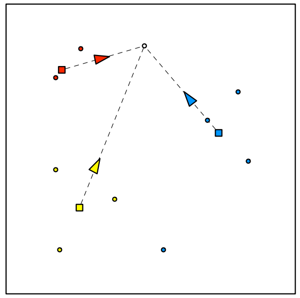
$$p_{i^*} = \frac{(\#\text{pts assigned in past})\, p_{i^*} + q_j}{\#\text{pts assigned in past} + 1}$$

Algorithms

No sensor policy

For all time t, each vehicle moves towards:

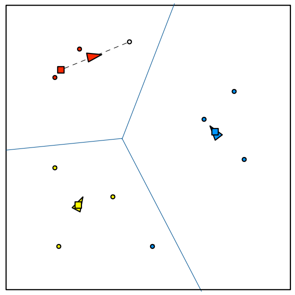
- the nearest outstanding task; else,
- the (nearest) point minimizing the average distance to tasks *serviced in the past*



Sensor-based policy

For all time t, each vehicle moves towards:

- the nearest among outstanding tasks that is closest to it than other vehicles; else,
- the (nearest) point minimizing the average distance to tasks *serviced in the past*

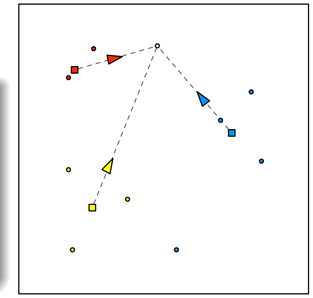


Algorithms

No sensor policy

For all time t, each vehicle moves towards:

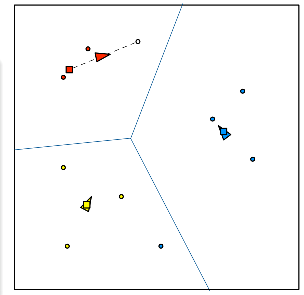
- the nearest outstanding task; else,
- the (nearest) point minimizing the average distance to tasks *serviced in the past*



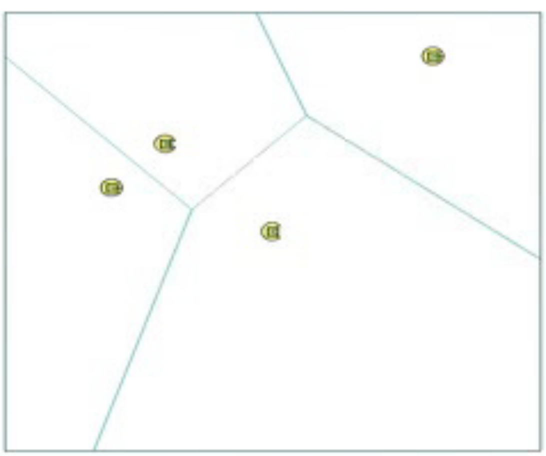
Sensor-based policy

For all time t, each vehicle moves towards:

- the nearest among outstanding tasks that is closest to it than other vehicles; else,
- the (nearest) point minimizing the average distance to tasks *serviced in the past*



Illustration

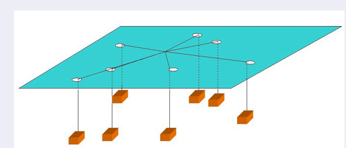


Differences with the MacQueen algorithm

- 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

$$\mathrm{FW}_i = \mathrm{argmin}_{p_i \in \mathcal{Q}} \sum_{q \in \text{past tasks}_i} \|q - p_i\|$$

$$\mathrm{Mean}_i = \frac{1}{|\text{past tasks}_i|} \sum_{q \in \text{past tasks}_i} q$$



- No simple recursion like the MacQueen algorithm $\longrightarrow$ need 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 $(p_1(t), \dots, p_m(t))$

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:
  - $C_i(t) := \{y \in \mathcal{Q} \mid \|\sum_{q \in \text{past tasks}_i} \text{vers}(y-q)\| \leq 1\}$
  - 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

Analysis of the algorithm

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

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:
  - $C_i(t) := \{y \in \mathcal{Q} \mid \|\sum_{q \in \text{past tasks}_i} \text{vers}(y-q)\| \leq 1\}$
  - 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

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

Coverage as a geometric game

Strategies

- $p = (p_1, \dots, p_m) \in \mathcal{Q}^m$
- 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 diam($\mathcal{Q}$) time, whichever occurs first.
- Utility function: expected time spent alone at the next task location

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

Coverage as a geometric game

Strategies

- $p = (p_1, \dots, p_m) \in \mathcal{Q}^m$
- 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 $\text{diam}(\mathcal{Q})$ 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

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

$\dot{p}_i = \frac{\partial}{\partial p_i}\mathcal{U}_i(p) = -\int_{V_i(p)} \frac{p_i - q}{\|p_i - q\|}\varphi(q)dq \implies$ gradient descent policy

Limited information

- No knowledge of φ
- No inter-agent communication

Approximations

- Empirical Utility Maximization: $p_i(t) = \text{argmax}_{x\in\mathcal{Q}} \sum_{q\sim\varphi} R_i(x, p_{-i}, q)$
- $\hat{R}_i(x, p_{-i}, q) = \text{diam}(\mathcal{Q}) - \|x - q\|$ 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

$\dot{p}_i = \frac{\partial}{\partial p_i}\mathcal{U}_i(p) = -\int_{V_i(p)} \frac{p_i - q}{\|p_i - q\|}\varphi(q)dq \implies$ gradient descent policy

Limited information

- No knowledge of φ
- No inter-agent communication

Approximations

- Empirical Utility Maximization: $p_i(t) = \text{argmax}_{x\in\mathcal{Q}} \sum_{q\sim\varphi} R_i(x, p_{-i}, q)$
- $\hat{R}_i(x, p_{-i}, q) = \text{diam}(\mathcal{Q}) - \|x - q\|$ 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

$\dot{p}_i = \frac{\partial}{\partial p_i}\mathcal{U}_i(p) = -\int_{V_i(p)} \frac{p_i - q}{\|p_i - q\|}\varphi(q)dq \implies$ gradient descent policy

Limited information

- No knowledge of φ
- No inter-agent communication

Approximations

- Empirical Utility Maximization: $p_i(t) = \text{argmax}_{x\in\mathcal{Q}} \sum_{q\sim\varphi} R_i(x, p_{-i}, q)$
- $\hat{R}_i(x, p_{-i}, q) = \text{diam}(\mathcal{Q}) - \|x - q\|$ 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

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*	
11:00-11:45pm	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

Dynamic Vehicle Routing for Robotic Networks
Lecture #6: Different Demand Models

Francesco Bullo[1] Emilio Frazzoli[2] Marco Pavone[2]
Ketan Savla[2] Stephen L. Smith[2]

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

[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

Lecture outline

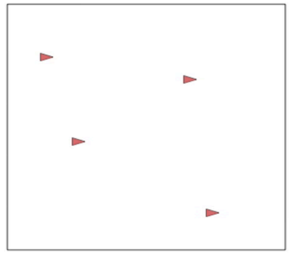
M. Pavone and E. Frazzoli. Dynamic vehicle routing with stochastic time constraints. In *IEEE Int. Conf. on Robotics and Automation*, Anchorage, AK, May 2010

M. Pavone, N. Bisnik, E. Frazzoli, and V. Isler. A stochastic and dynamic vehicle routing problem with time windows and customer impatience. *ACM/Springer Journal of Mobile Networks and Applications*, 14(3):350–364, 2009

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

Problem formulation

Problem statement

Solve problem OPT:

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

Well-posedness

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

Problem formulation

Problem statement

Solve problem OPT:

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

Well-posedness

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

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

Lower bound

OPT is lower bounded by:

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

Devised algorithms to solve $\underline{OPT}$

Lower bound

Intuition for lower bound:

$$\mathbb{P}\left[W_j < G_j\right] \leq \mathbb{P}\left[\min_{k\in\{1,\dots,m\}} \frac{\|X_j - X_k\|}{v} < G_j\right] \leq \sup_{(p_1,\dots,p_m)\in\mathcal{Q}^m} \underbrace{\mathbb{P}\left[\min_{k\in\{1,\dots,m\}} \frac{\|X_j - p_k\|}{v} < G_j\right]}_{\doteq \mathcal{H}(p_1,\dots,p_m)}$$

Lower bound

OPT is lower bounded by:

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

Devised algorithms to solve $\underline{OPT}$

Lower bound

Intuition for lower bound:

$$\mathbb{P}\left[W_j < G_j\right] \leq \mathbb{P}\left[\min_{k\in\{1,\dots,m\}} \frac{\|X_j - X_k\|}{v} < G_j\right] \leq \sup_{(p_1,\dots,p_m)\in\mathcal{Q}^m} \underbrace{\mathbb{P}\left[\min_{k\in\{1,\dots,m\}} \frac{\|X_j - p_k\|}{v} < G_j\right]}_{\doteq \mathcal{H}(p_1,\dots,p_m)}$$

Lower bound

OPT is lower bounded by:

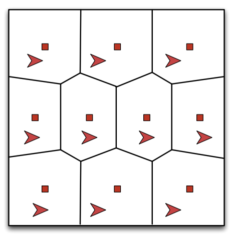
$$\underline{OPT}: \min_{m\in\mathbb{N}_{>0}} m \quad \text{s.t.} \quad \sup_{(p_1,\dots,p_m)\in\mathcal{Q}^m} \mathcal{H}(p_1,\dots,p_m) \geq \phi^{\mathrm{d}}$$

Devised algorithms to solve $\underline{OPT}$

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

Nearest Depot Assignment (NDA) policy

Compute maximum of $\mathcal{H}$: $(\bar{p}_1,\dots,\bar{p}_m)$.
Then:
1: $\bar{p}_k$ is depot of kth vehicle
2: nearest-depot assignment
3: FCFS service



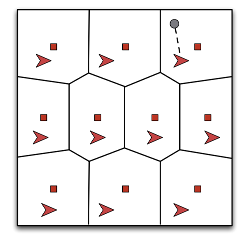
Proof sketch:

- 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 $\mathcal{H}$: $(\bar{p}_1,\dots,\bar{p}_m)$.
Then:
1: $\bar{p}_k$ is depot of kth vehicle
2: nearest-depot assignment
3: FCFS service



Proof sketch:

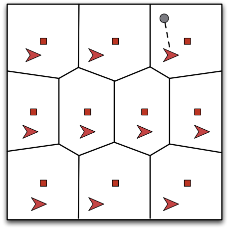
- as usual, as $\lambda \to 0^+$, the problem reduces to optimal pre-positioning

NDA policy (optimal as $\lambda \longrightarrow 0$)

Nearest Depot Assignment (NDA) policy

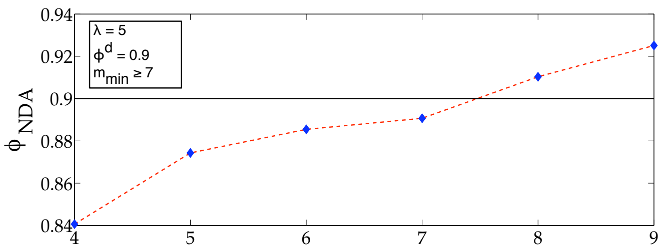
Compute maximum of $\mathcal{H}$: $(\bar{p}_1, \dots, \bar{p}_m)$.
Then:

1: $\bar{p}_k$ is depot of kth vehicle
2: nearest-depot assignment
3: FCFS service



Proof sketch:

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

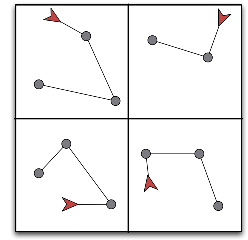


Batch policy

Batch (B) policy

Partition $\mathcal{Q}$ 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

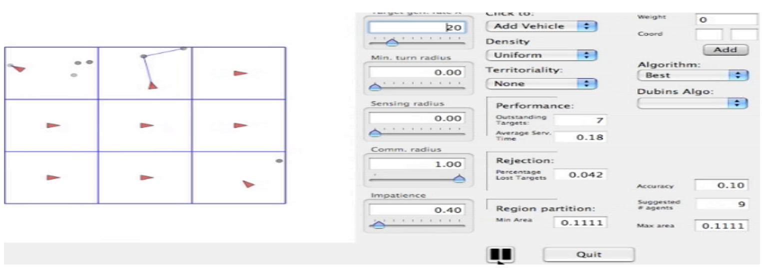
- if s=0: $m_B = \min\left\{ m \,\middle|\, \sup_{\theta \in \mathbb{R}_{>0}} (1 - F_G(\theta))(1 - \frac{\lambda \cdot \text{const}}{\theta m^2}) \geq \phi^{\text{d}} \right\}$
- with time windows: $m_B/m^* \leq 3.78$, when λ large and $\phi^d \to 1^-$

Characterization of batch policy

Proof sketch (m=1):

- upper bound expected length of TSP tour with const $\cdot \lambda/m^2$, via control-theoretical methods
- use Markov's ineq to lower bound:

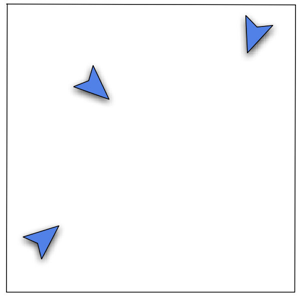
$$\begin{aligned} \mathbb{P}[W < G] &\geq \mathbb{P}[W < G | 2\,\text{TSP} < \theta]\mathbb{P}[2\,\text{TSP} < \theta] \\ &\geq (1 - F_G(\theta))(1 - \mathbb{E}[2\,\text{TSP}]/\theta) \end{aligned}$$



Lecture outline

Demands with priority levels

- m vehicles
- n classes of demands
  - $1 =$ highest priority
  - $n =$ lowest priority
- Poisson arrivals $\lambda_1, \dots, \lambda_n$
- locations uniformly distributed
  can extend to non-uniform φ



Steady-state expected system-time $\overline{T}_1, \dots, \overline{T}_n$

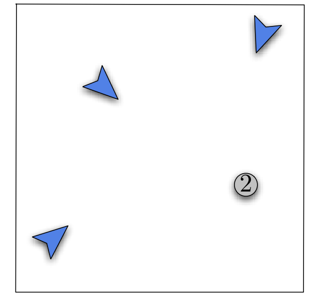
Goal for vehicles

Minimize $c_1 \overline{T}_1 + \dots + c_n \overline{T}_n$ $\qquad (\uparrow c_i \Rightarrow \uparrow \text{priority of class } i)$

S. L. Smith, M. Pavone, F. Bullo, and E. Frazzoli. Dynamic vehicle routing with priority classes of stochastic demands. *SIAM Journal on Control and Optimization*, 48(5):3224–3245, 2010

Demands with priority levels

- m vehicles
- n classes of demands
  - $1 =$ highest priority
  - $n =$ lowest priority
- Poisson arrivals $\lambda_1, \dots, \lambda_n$
- locations uniformly distributed
  can extend to non-uniform φ



Steady-state expected system-time $\overline{T}_1, \dots, \overline{T}_n$

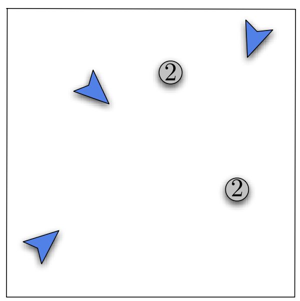
Goal for vehicles

Minimize $c_1 \overline{T}_1 + \dots + c_n \overline{T}_n$ $\qquad (\uparrow c_i \Rightarrow \uparrow \text{priority of class } i)$

S. L. Smith, M. Pavone, F. Bullo, and E. Frazzoli. Dynamic vehicle routing with priority classes of stochastic demands. *SIAM Journal on Control and Optimization*, 48(5):3224–3245, 2010

Demands with priority levels

- m vehicles
- n classes of demands
  - $1 =$ highest priority
  - $n =$ lowest priority
- Poisson arrivals $\lambda_1, \dots, \lambda_n$
- locations uniformly distributed
  can extend to non-uniform φ



Steady-state expected system-time $\overline{T}_1, \dots, \overline{T}_n$

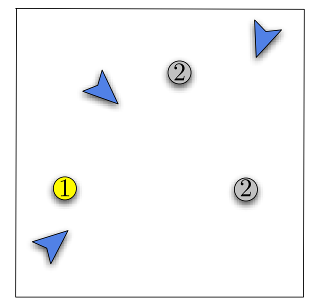
Goal for vehicles

Minimize $c_1 \overline{T}_1 + \dots + c_n \overline{T}_n$ $\qquad (\uparrow c_i \Rightarrow \uparrow \text{priority of class } i)$

S. L. Smith, M. Pavone, F. Bullo, and E. Frazzoli. Dynamic vehicle routing with priority classes of stochastic demands. *SIAM Journal on Control and Optimization*, 48(5):3224–3245, 2010

Demands with priority levels

- m vehicles
- n classes of demands
  - $1 =$ highest priority
  - $n =$ lowest priority
- Poisson arrivals $\lambda_1, \dots, \lambda_n$
- locations uniformly distributed
  can extend to non-uniform φ



Steady-state expected system-time $\overline{T}_1, \dots, \overline{T}_n$

Goal for vehicles

Minimize $c_1 \overline{T}_1 + \dots + c_n \overline{T}_n$ $\qquad (\uparrow c_i \Rightarrow \uparrow \text{priority of class } i)$

S. L. Smith, M. Pavone, F. Bullo, and E. Frazzoli. Dynamic vehicle routing with priority classes of stochastic demands. *SIAM Journal on Control and Optimization*, 48(5):3224–3245, 2010

Literature Review

Classic Priority Queueing

L. Kleinrock. *Queueing Systems. Volume II: Computer Applications*. Wiley, New York, 1976

E. G. Coffman Jr. and I. Mitrani. A characterization of waiting time performance realizable by single-server queues. *Operations Research*, 28(3):810–821, 1980

Related Combinatorial Problems

A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan. The minimum latency problem. In *ACM Symposium on the Theory of Computing*, pages 163–171, Montreal, Canada, 1994

M. Z. Spivey and W. B. Powell. The dynamic assignment problem. *Transportation Science*, 38(4):399–419, 2004

A. Blum, S. Chawla, D. R. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward TSP. *SIAM Journal on Computing*, 37(2):653–670, 2007

Load Factor and Stability

Stable: Queue remains bounded

Define **load factor** as

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

- λ_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

1. Light load $\varrho \to 0^+$
2. 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 $\overline{T}^*_c$ = optimal value of cost $c_1 \overline{T}_1 + \cdots + c_n \overline{T}_n$.

Lower bound for every policy

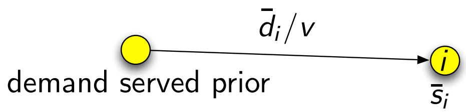
$$\overline{T}^*_c \geq \frac{\beta_{\mathrm{TSP}} |Q|}{2m^2 v^2 (1-\varrho)^2} \sum_{\alpha=1}^{n} \left(c_\alpha + 2 \sum_{j=\alpha+1}^{n} c_j \right) \lambda_\alpha$$

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 α
- travel distance is $r_\alpha \bar{d}_\alpha$



- For stability: $\sum_{i=1}^{n} \lambda_i \left(r_i \bar{d}_i/v + \bar{s}_i\right) < m$
- Can bound travel distance as

$$\bar{d}_\alpha \geq \frac{\beta_{\mathrm{TSP}}}{\sqrt{2}} \sqrt{\frac{|Q|}{\sum_i r_i \overline{N}_i}}$$

- generates a linear program with $2^n - 1$ constraints, one for each combination $\{r_1, \dots, r_n\}$
- solution to LP is largest lower bound

Separate Queues Policy

Input: Probability distribution $\mathbf{p} = [p_1, \dots, p_n]$.

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 $\mathbf{p}$
2: Service all demands of selected class following TSP
3: Repeat

Policy performance optimized over $\mathbf{p}$.

Separate Queues Performance

Heavy load performance

For the SQ policy,

$$\frac{\overline{T}_{c,\mathrm{SQ}}}{\overline{T}_c^*} \leq 2n^2$$

as $\varrho \to 1^-$.

- n = number of classes
- independent of $\varrho, c, \bar{s}, \lambda$

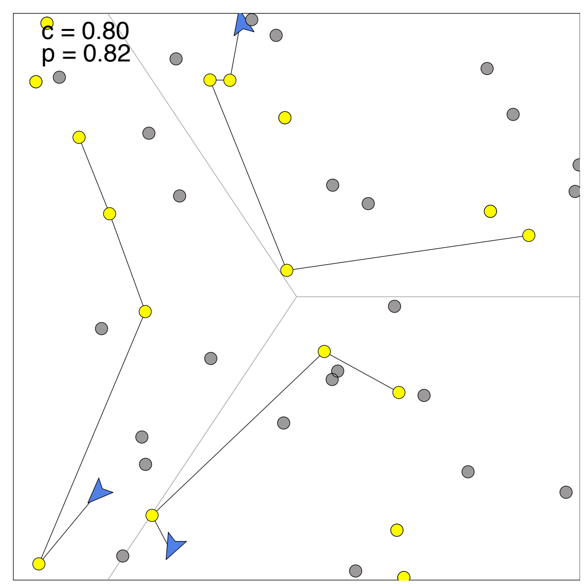
Heuristic Improvements:

1. Receding horizon: service only a fraction η of TSP
2. when following TSP, service newly arrived demands within ϵ of TSP.

$$\epsilon \sqrt{\frac{\mu |Q|}{\sum_{\alpha=1}^{n} \overline{N}_\alpha}},$$

where μ 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 $c_2 = 0.2$
- $\mathbf{p} = [0.82, 0.18]$

Proof idea for upper bound

- In heavy-load, shortest path through N points:

$$= \beta_{\mathrm{TSP}} \sqrt{|Q| N} \quad \text{with prob. 1 (BHH theorem)}$$

- Study expected # of outstanding demands at each iteration

$$N_i(t+1) \leq f\big(N_1(t), \dots, N_m(t), \mathbf{p}, \lambda, \bar{s}\big)$$

- Function f has a linear part plus a sub-linear part
- Bound evolution by stable linear system for all $\varrho < 1$

$$\mathcal{N}(t+1) = A(\mathbf{p}, \lambda, \bar{s})\mathcal{N}(t) + B(\mathbf{p}, \lambda, \bar{s})$$

- Allows computation of $\limsup_{t \to +\infty} \mathcal{N}_i(t)$
- 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

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*	
11:00-11:45pm	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

Dynamic Vehicle Routing for Robotic Networks
Lecture #7: Vehicle Models

Francesco Bullo[1] Emilio Frazzoli[2] Marco Pavone[2]
Ketan Savla[2] Stephen L. Smith[2]

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

[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

Outline of the lecture

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-Shepps 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

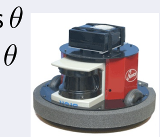
$$\begin{aligned}\dot{x} &= \cos\theta\\ \dot{y} &= \sin\theta\\ \dot{\theta} &= \omega\end{aligned}$$
$|\omega| \leq 1/\rho$

Reeds-Shepp car

$$\begin{aligned}\dot{x} &= v\cos\theta\\ \dot{y} &= v\sin\theta\\ \dot{\theta} &= \omega\end{aligned}$$
$v \in \{-1, 1\}$
$|\omega| \leq 1/\rho$

Differential drive

$$\begin{aligned}\dot{x} &= \tfrac{1}{2}(\omega_l + \omega_r)\cos\theta\\ \dot{y} &= \tfrac{1}{2}(\omega_l + \omega_r)\sin\theta\\ \dot{\theta} &= \tfrac{1}{\rho}(\omega_r - \omega_l)\end{aligned}$$
$|\omega_l| \leq 1$; $|\omega_r| \leq 1$



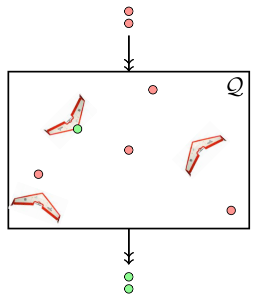
Double integrator

$\ddot{x} = u$
$\|\dot{x}\| \leq 1$
$\|u\| \leq 1$

DTRP formulation

Problem setup

- m identical vehicles in $\mathcal{Q}$
- Spatio-temporal Poisson process: rate λ and uniform spatial density
- On-site service time $s = 0$



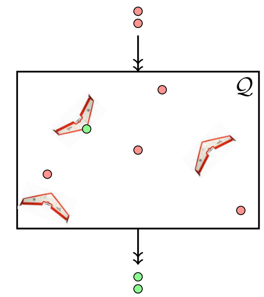
Objective

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

DTRP formulation

Problem setup

- m identical vehicles in $\mathcal{Q}$
- Spatio-temporal Poisson process: rate λ and uniform spatial density
- On-site service time $s = 0$



Objective

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

Stabilizability

- $\underbrace{\lambda}_{\text{task generation rate}} \quad - \quad \underbrace{m \cdot \frac{n}{\text{TSPlength}(n)}}_{\text{task service rate}} \quad = \quad \text{task growth rate}$

  n: # outstanding tasks
- TSPlength(n) strictly sub-linear $\implies$ stability $\forall \lambda, m$
- Euclidean 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

- $\underbrace{\lambda}_{\text{task generation rate}} \quad - \quad \underbrace{m \cdot \frac{n}{\text{TSPlength}(n)}}_{\text{task service rate}} \quad = \quad \text{task growth rate}$

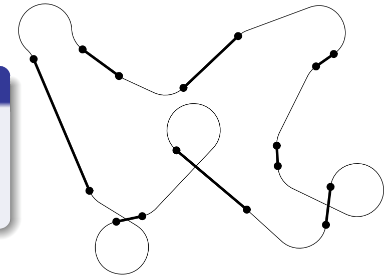
  n: # outstanding tasks
- TSPlength(n) strictly sub-linear $\implies$ stability $\forall \lambda, m$
- Euclidean TSPlength$(n) = \Theta(n^{1/2})$ (Beardwood et. al. '59)
- Euclidean TSP based path planning heuristic $\implies$ $O(n)$
- Traveling salesperson problems for differential vehicles.

Outline of the lecture

Traveling Salesperson Problem

Problem Statement

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

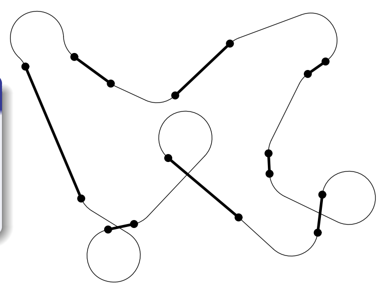


- NP-hardness a consequence of the NP-hardness of the Euclidean TSP.
- Does the cost of this TSP increase SUBLINEARLY with n?
- Is there a polynomial-time algorithm that returns a tour of length $o(n)$??
- What is the quality of the solution?

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

1. K. Savla, E. Frazzoli, and F. Bullo. Traveling Salesperson Problems for the Dubins vehicle. *IEEE Transactions on Automatic Control*, 53(6):1378–1391, 2008
2. K. Savla, F. Bullo, and E. Frazzoli. Traveling Salesperson Problems for a double integrator. *IEEE Transactions on Automatic Control*, 54(4):788–793, 2009
3. J. J. Enright, K. Savla, E. Frazzoli, and F. Bullo. Stochastic and dynamic routing problems for multiple UAVs. *AIAA Journal of Guidance, Control, and Dynamics*, 34(4):1152–1166, 2009
4. J. J. Enright and E. Frazzoli. The stochastic Traveling Salesman Problem for the Reeds-Shepp car and the differential drive robot. In *IEEE Conf. on Decision and Control*, pages 3058–3064, San Diego, CA, December 2006
5. K. Savla and E. Frazzoli. On endogenous reconfiguration for mobile robotic networks. In *Workshop on Algorithmic Foundations of Robotics*, Guanajuato, Mexico, December 2008
6. M. Pavone, K. Savla, and E. Frazzoli. Sharing the load. *IEEE Robotics and Automation Magazine*, 16(2):52–61, 2009
7. F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. L. Smith. Dynamic vehicle routing for robotic systems. *Proceedings of the IEEE*, May 2010. Submitted
8. S. Rathinam, R. Sengupta, and S. Darbha. A resource allocation algorithm for multi-vehicle systems with non holonomic constraints. *IEEE Transactions on Automation Sciences and Engineering*, 4(1):98–104, 2007
9. J. Le Ny, E. Feron, and E. Frazzoli. On the curvature-constrained traveling salesman problem. *IEEE Transactions on Automatic Control*, 2009. to appear
10. S. Itani. *Dynamic Systems and Subadditive Functionals*. PhD thesis, Massachusetts Institute of Technology, 2009

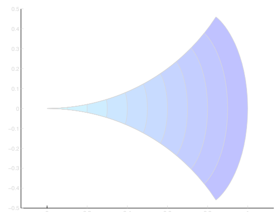
Stochastic TSP: A nearest-neighbor lower bound

Outline of the calculations

- Calculate (an upper bound on) expected distance from an arbitrary vehicle configuration to closest point, δ^*
  - Calculate (an upper bound on) the area of the set reachable with a path of length δ, $\mathcal{R}_\delta$.
  - $\Pr(\delta^* \geq \delta) \geq \max\{0, 1 - n|\mathcal{R}_\delta|/|\mathcal{Q}|\}$
- Expected length of the tour cannot be less than n times $\mathbb{E}[\delta^*]$

Example: Dubins vehicle

- $|\mathcal{R}_\delta| = \frac{\delta^3}{3\rho}$
- $\mathbb{E}[\delta^*] = \frac{3}{4}\left(\frac{3\rho|\mathcal{Q}|}{n}\right)^{1/3}$.
- $\lim_{n\to\infty} \frac{\mathbb{E}[\mathrm{TSP}(n)]}{n^{2/3}} \geq \frac{3}{4}(3\rho|\mathcal{Q}|)^{1/3}$.



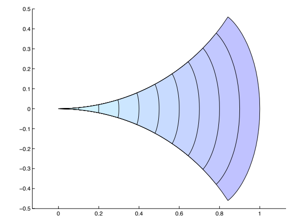
Stochastic TSP: A nearest-neighbor lower bound

Outline of the calculations

- Calculate (an upper bound on) expected distance from an arbitrary vehicle configuration to closest point, δ^*
  - Calculate (an upper bound on) the area of the set reachable with a path of length δ, $\mathcal{R}_\delta$.
  - $\Pr(\delta^* \geq \delta) \geq \max\{0, 1 - n|\mathcal{R}_\delta|/|\mathcal{Q}|\}$
- Expected length of the tour cannot be less than n times $\mathbb{E}[\delta^*]$

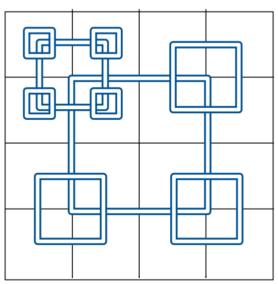
Example: Dubins vehicle

- $|\mathcal{R}_\delta| = \frac{\delta^3}{3\rho}$
- $\mathbb{E}[\delta^*] = \frac{3}{4}\left(\frac{3\rho|\mathcal{Q}|}{n}\right)^{1/3}$.
- $\lim_{n\to\infty} \frac{\mathbb{E}[\mathrm{TSP}(n)]}{n^{2/3}} \geq \frac{3}{4}(3\rho|\mathcal{Q}|)^{1/3}$.



Towards an upper bound: tiling based algorithms

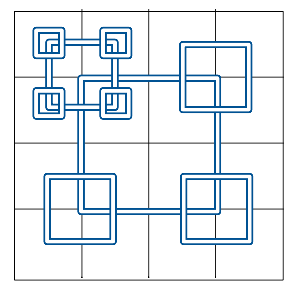
- The way the ETSP tours are constructed relies on the scaling properties of tours: the length of the tour scales as the coordinates of the points.



- 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

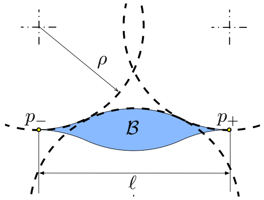
Towards an upper bound: tiling based algorithms

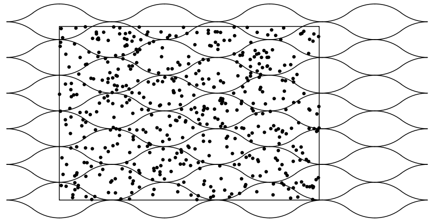
- The way the ETSP tours are constructed relies on the scaling properties of tours: the length of the tour scales as the coordinates of the points.



- 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

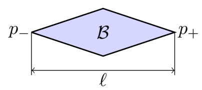




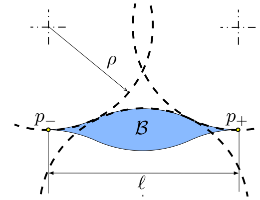
Bead properties

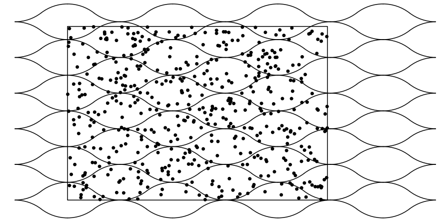
- Length$(p_-, q, p_+) \leq \ell + o(\ell^2)$ for all $q \in \mathcal{B}$
- Width: $w(\ell) = \frac{\ell^2}{8\rho} + o(\ell^3)$
- The beads tile the plane
- Useful for Dubins vehicle, Reeds-Shepp car and double integrator

- Diamond-like cell for differential drive



Bead construction

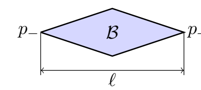




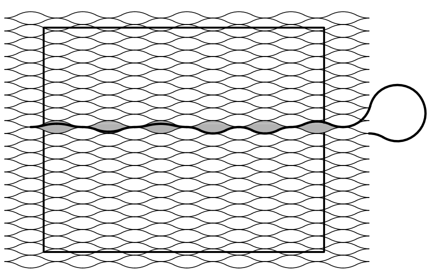
Bead properties

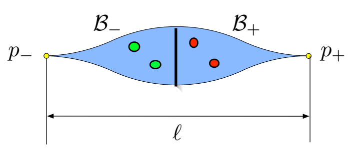
- Length$(p_-, q, p_+) \leq \ell + o(\ell^2)$ for all $q \in \mathcal{B}$
- Width: $w(\ell) = \frac{\ell^2}{8\rho} + o(\ell^3)$
- The beads tile the plane
- Useful for Dubins vehicle, Reeds-Shepp car and double integrator

- Diamond-like cell for differential drive



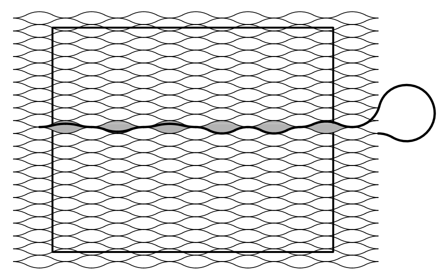
The single-sweep tiling algorithm

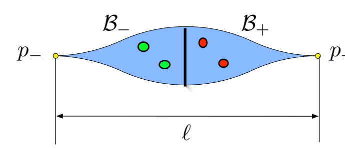




- Tile the region with beads
- Sweep the bead rows, while servicing all the targets in every bead as follows:
  - Service every task q in $\mathcal{B}_-$ using the "$p_- \to q \to p_-$" protocol
  - Move from p_- to p_+
  - Service every task q in $\mathcal{B}_+$ using the "$p_+ \to q \to p_+$" protocol

The single-sweep tiling algorithm



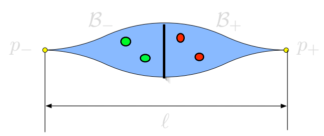


- Tile the region with beads
- Sweep the bead rows, while servicing all the targets in every bead as follows:
  - Service every task q in $\mathcal{B}_-$ using the "$p_- \to q \to p_-$" protocol
  - Move from p_- to p_+
  - Service every task q in $\mathcal{B}_+$ using the "$p_+ \to q \to p_+$" protocol

Analysis of the single-sweep tiling algorithm

Path length calculations

$TSP(n) =$ (bead row length + move to next bead row) $\times$ # bead rows + move to service each task $\times$ # tasks + tour closure length



- For a Reeds-Shepp car, as $\ell \to 0$:

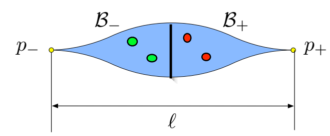
$$TSP(n) \leq \left(\sqrt{|\mathcal{Q}|} + \ell/2\right)\frac{\sqrt{|\mathcal{Q}|}}{w(\ell)/2} + \ell n + 2\left(\sqrt{|\mathcal{Q}|} + \rho\pi\right)$$
$$\leq 16\rho\frac{|\mathcal{Q}|}{\ell^2} + 8\rho\frac{\sqrt{|\mathcal{Q}|}}{\ell} + \ell n + 2\left(\sqrt{|\mathcal{Q}|} + \rho\pi\right) \left(\because w(\ell) \approx \frac{\ell^2}{8\rho}\right)$$

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

Analysis of the single-sweep tiling algorithm

Path length calculations

$TSP(n) =$ (bead row length + move to next bead row) $\times$ # bead rows + move to service each task $\times$ # tasks + tour closure length



- For a Reeds-Shepp car, as $\ell \to 0$:

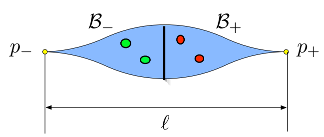
$$TSP(n) \leq \left(\sqrt{|\mathcal{Q}|} + \ell/2\right)\frac{\sqrt{|\mathcal{Q}|}}{w(\ell)/2} + \ell n + 2\left(\sqrt{|\mathcal{Q}|} + \rho\pi\right)$$
$$\leq 16\rho\frac{|\mathcal{Q}|}{\ell^2} + 8\rho\frac{\sqrt{|\mathcal{Q}|}}{\ell} + \ell n + 2\left(\sqrt{|\mathcal{Q}|} + \rho\pi\right) \left(\because w(\ell) \approx \frac{\ell^2}{8\rho}\right)$$

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

Analysis of the single-sweep tiling algorithm

Path length calculations

$TSP(n) =$ (bead row length + move to next bead row) $\times$ # bead rows + move to service each task $\times$ # tasks + tour closure length



- For a Reeds-Shepp car, as $\ell \to 0$:

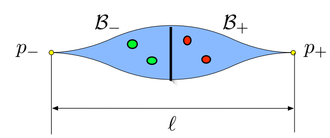
$$TSP(n) \leq \left(\sqrt{|\mathcal{Q}|} + \ell/2\right)\frac{\sqrt{|\mathcal{Q}|}}{w(\ell)/2} + \ell n + 2\left(\sqrt{|\mathcal{Q}|} + \rho\pi\right)$$
$$\leq 16\rho\frac{|\mathcal{Q}|}{\ell^2} + 8\rho\frac{\sqrt{|\mathcal{Q}|}}{\ell} + \ell n + 2\left(\sqrt{|\mathcal{Q}|} + \rho\pi\right) \left(\because w(\ell) \approx \frac{\ell^2}{8\rho}\right)$$

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

Analysis of the single-sweep tiling algorithm

Path length calculations

$TSP(n) =$ (bead row length + move to next bead row) $\times$ # bead rows + move to service each task $\times$ # tasks + tour closure length



- For a Dubins vehicle, as $\ell \to 0$:

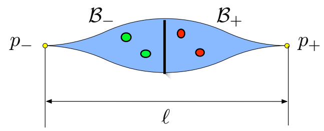
$$TSP(n) \leq \left(\sqrt{|\mathcal{Q}|} + \frac{w(\ell)}{2} + \kappa\right)\frac{\sqrt{|\mathcal{Q}|}}{w(\ell)/2} + (\ell + \kappa)n + 2\sqrt{|\mathcal{Q}|} + \kappa$$
$$\leq 16\rho\frac{|\mathcal{Q}|}{\ell^2} + \sqrt{|\mathcal{Q}|} + 16\kappa\frac{\sqrt{|\mathcal{Q}|}}{\ell^2} + \ell n + \underline{\boldsymbol{\kappa n}} + 2\sqrt{|\mathcal{Q}|} + \kappa$$

- The κn term grows linearly in n for all ℓ $\implies$ TSP$(n) = O(n)$

Analysis of the single-sweep tiling algorithm

Path length calculations

$TSP(n) =$ (bead row length + move to next bead row) $\times$ # bead rows + move to service each task $\times$ # tasks + tour closure length



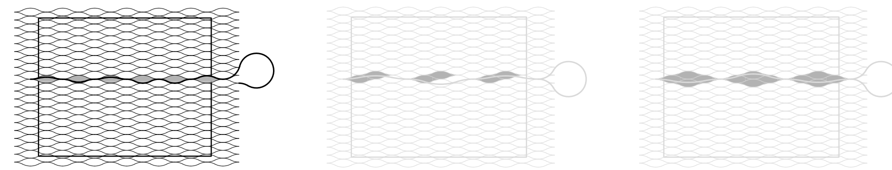
- For a Dubins vehicle, as $\ell \to 0$:

$$\begin{aligned} TSP(n) &\leq \left(\sqrt{|\mathcal{Q}|} + \frac{w(\ell)}{2} + \kappa\right)\frac{\sqrt{|\mathcal{Q}|}}{w(\ell)/2} + (\ell + \kappa)n + 2\sqrt{|\mathcal{Q}|} + \kappa \\ &\leq 16\rho\frac{|\mathcal{Q}|}{\ell^2} + \sqrt{|\mathcal{Q}|} + 16\kappa\frac{\sqrt{|\mathcal{Q}|}}{\ell^2} + \ell n + \underline{\kappa \mathbf{n}} + 2\sqrt{|\mathcal{Q}|} + \kappa \end{aligned}$$

- The κn term grows linearly in n for all $\ell \implies \text{TSP}(n) = O(n)$

The recursive sweep tiling algorithm

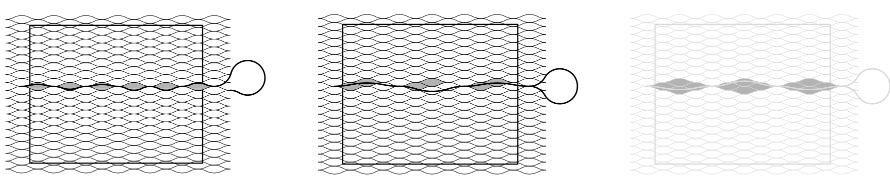
- Tile $\mathcal{Q}$ with beads such that: $\frac{|\mathcal{B}|}{|\mathcal{Q}|} = \frac{1}{2n}$ (i.e., $\ell \sim n^{-1/3}$)
- Sweep the bead rows, visiting one target per non-empty bead.
- Iterate, using at the i-th phase a "meta-bead" composed of 2^{i-1} beads.
- After $\log n$ phases, visit the outstanding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



Phase 1 Phase 2 Phase 3

The recursive sweep tiling algorithm

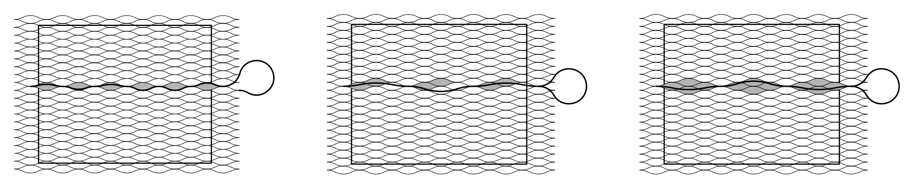
- Tile $\mathcal{Q}$ with beads such that: $\frac{|\mathcal{B}|}{|\mathcal{Q}|} = \frac{1}{2n}$ (i.e., $\ell \sim n^{-1/3}$)
- Sweep the bead rows, visiting one target per non-empty bead.
- Iterate, using at the i-th phase a "meta-bead" composed of 2^{i-1} beads.
- After $\log n$ phases, visit the outstanding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



Phase 1 Phase 2 Phase 3

The recursive sweep tiling algorithm

- Tile $\mathcal{Q}$ with beads such that: $\frac{|\mathcal{B}|}{|\mathcal{Q}|} = \frac{1}{2n}$ (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.



Phase 1 Phase 2 Phase 3

Analysis of the recursive algorithm

- Theorem: For a Dubins vehicle, with probability one,

$$\limsup_{n\to\infty} \frac{\mathrm{TSP}(n)}{n^{2/3}} \le 24\sqrt[3]{\rho|\mathcal{Q}|}\left(1+\frac{7}{3}\pi\frac{\rho}{\sqrt{|\mathcal{Q}|}}\right)$$

Outline of the proof

- $\Pr(\lim_{n\to\infty} \#$ tasks remaining after phase $i^* > 24\log n) = 0$
- Path length calculations:
  - Phase 1 path length $O\left(\frac{1}{\ell^2}\right) = O\left(n^{2/3}\right) \quad (\because \quad \ell \sim n^{-1/3})$
  - 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 $O(\log n)$

Analysis of the recursive algorithm

- Theorem: For a Dubins vehicle, with probability one,

$$\limsup_{n\to\infty} \frac{\mathrm{TSP}(n)}{n^{2/3}} \le 24\sqrt[3]{\rho|\mathcal{Q}|}\left(1+\frac{7}{3}\pi\frac{\rho}{\sqrt{|\mathcal{Q}|}}\right)$$

Outline of the proof

- $\Pr(\lim_{n\to\infty} \#$ tasks remaining after phase $i^* > 24\log n) = 0$
- Path length calculations:
  - Phase 1 path length $O\left(\frac{1}{\ell^2}\right) = O\left(n^{2/3}\right) \quad (\because \quad \ell \sim n^{-1/3})$
  - 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 $O(\log n)$

FB, EF, MP, KS, SLS (UCSB, MIT) Dynamic Vehicle Routing (Lecture 7/8) 29jun10 @ Baltimore, ACC 17 / 36

Summary of TSPs

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

Stabilizability of the DTRP

- $\underbrace{\lambda}_{\text{task generation rate}} \quad - \quad \underbrace{m\cdot\frac{n}{\mathrm{TSP}(n)}}_{\text{task service rate}} \quad = \quad \text{task growth rate}$

  n: # outstanding tasks

- $\mathbb{E}[\mathrm{TSP}(n)] \in \Theta(n^{2/3}) \implies$ trivial receding horizon TSP-based policies are stable for the DTRP for all λ and m

Summary of TSPs

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

Stabilizability of the DTRP

- $\underbrace{\lambda}_{\text{task generation rate}} \quad - \quad \underbrace{m\cdot\frac{n}{\mathrm{TSP}(n)}}_{\text{task service rate}} \quad = \quad \text{task growth rate}$

  n: # outstanding tasks

- $\mathbb{E}[\mathrm{TSP}(n)] \in \Theta(n^{2/3}) \implies$ trivial receding horizon TSP-based policies are stable for the DTRP for all λ and m

FB, EF, MP, KS, SLS (UCSB, MIT) Dynamic Vehicle Routing (Lecture 7/8) 29jun10 @ Baltimore, ACC 18 / 36

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_π be the number of outstanding tasks at steady-state under stable policy π
- Calculate (an upper bound on) expected distance from an arbitrary vehicle configuration to closest among n_π 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

- $\mathbb{E}[\delta^*(n_\pi)] = \frac{3}{4}\left(\frac{3\rho|\mathcal{Q}|}{n_\pi}\right)^{1/3}$
- Steady state+ Little's formula: $\frac{\lambda}{m} = \frac{4}{3}\left(\frac{\lambda T_\pi}{3\rho|\mathcal{Q}|}\right)^{1/3}$
- $\liminf_{\frac{\lambda}{m}\to+\infty} \overline{T}^* \frac{m^3}{\lambda^2} \geq \frac{81}{64}\rho|\mathcal{Q}|$

The heavy load case: nearest neighbor lower bound

Outline of the calculations

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

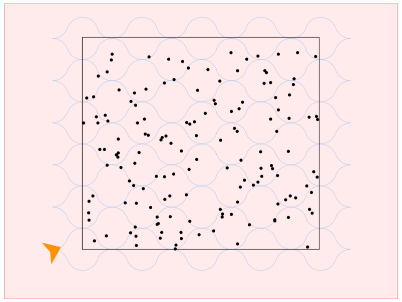
Example: Dubins vehicle

- $\mathbb{E}[\delta^*(n_\pi)] = \frac{3}{4}\left(\frac{3\rho|\mathcal{Q}|}{n_\pi}\right)^{1/3}$
- Steady state+ Little's formula: $\frac{\lambda}{m} = \frac{4}{3}\left(\frac{\lambda T_\pi}{3\rho|\mathcal{Q}|}\right)^{1/3}$
- $\liminf_{\frac{\lambda}{m}\to+\infty} \overline{T}^* \frac{m^3}{\lambda^2} \geq \frac{81}{64}\rho|\mathcal{Q}|$

The multiple sweep tiling algorithm

The single vehicle version

1. Tile $\mathcal{Q}$ with beads of length $\ell = c/\lambda$
2. Update outstanding task list
3. Execute single sweep tiling algorithm
4. Goto 2.



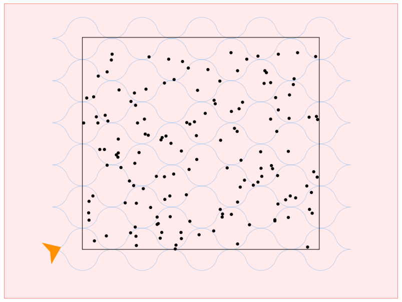
The multi-vehicle version

- Divide $\mathcal{Q}$ 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

1. Tile $\mathcal{Q}$ with beads of length $\ell = c/\lambda$
2. Update outstanding task list
3. Execute single sweep tiling algorithm
4. Goto 2.



The multi-vehicle version

- Divide $\mathcal{Q}$ 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 $\lambda_{\mathcal{B}} = \lambda \frac{|\mathcal{B}|}{|\mathcal{Q}|}$
- The vehicle visits each bead with at a rate no smaller than $\mu_{\mathcal{B}} \approx (\text{single sweep path length})^{-1}$
- The system time is no greater than the system time for the corresponding M/D/1 queue: $\overline{T}^* \leq \frac{1}{\mu_{\mathcal{B}}} \left(1 + \frac{1}{2} \frac{\lambda_{\mathcal{B}}}{\mu_{\mathcal{B}} - \lambda_{\mathcal{B}}}\right)$
- Optimize over ℓ

Example: Dubins vehicle

- $\lambda_{\mathcal{B}} = \frac{\ell^3 \lambda}{16 \rho |\mathcal{Q}|}$; $\mu_{\mathcal{B}} \geq \frac{\ell^2 m}{16 \rho |\mathcal{Q}|} \left(1 + \frac{7}{3}\pi \frac{\rho}{\sqrt{|\mathcal{Q}|}}\right)^{-1}$
- $\limsup_{\frac{\lambda}{m} \to +\infty} \overline{T}^* \frac{m^3}{\lambda^2} \leq 71 \rho |\mathcal{Q}| \left(1 + \frac{7}{3}\pi \frac{\rho}{\sqrt{|\mathcal{Q}|}}\right)^3$

Analysis of the multiple sweep algorithm

General protocol

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

Example: Dubins vehicle

- $\lambda_{\mathcal{B}} = \frac{\ell^3 \lambda}{16 \rho |\mathcal{Q}|}$; $\mu_{\mathcal{B}} \geq \frac{\ell^2 m}{16 \rho |\mathcal{Q}|} \left(1 + \frac{7}{3}\pi \frac{\rho}{\sqrt{|\mathcal{Q}|}}\right)^{-1}$
- $\limsup_{\frac{\lambda}{m} \to +\infty} \overline{T}^* \frac{m^3}{\lambda^2} \leq 71 \rho |\mathcal{Q}| \left(1 + \frac{7}{3}\pi \frac{\rho}{\sqrt{|\mathcal{Q}|}}\right)^3$

Outline of the lecture

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?

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 $q \in \mathcal{Q}$ is lower bounded by $\|q - p\|$
- A simple lower bound on $\overline{T}^*$ is obtained by relaxing differential constraints
- $\overline{T}^* \geq \mathcal{H}_m^*(\mathcal{Q})$
- $\mathcal{H}_m^*(\mathcal{Q}) = \Theta\left(\frac{1}{\sqrt{m}}\right)$

The Median Circling (MC) Policy

Assign "virtual" generators to each agent. All agents do the following, in parallel (possibly asynchronously):

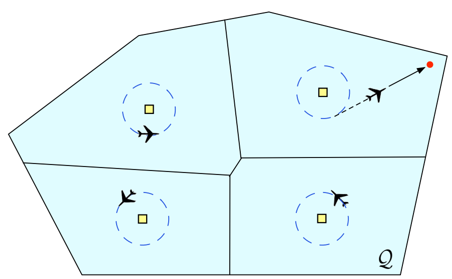
- Update the generator position according to a gradient descent law.
- Service targets in own region, returning to a "loitering circle" of radius 2.91ρ centered on their generator position when done

- We have

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

- Furthermore,

$$\lim_{\mathcal{H}_m^* \to +\infty, \lambda/m \to 0^+} \frac{T_{\mathrm{MC}}}{\overline{T}^*} = 1.$$



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 2.91ρ centered on their generator position when done

- We have

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

- Furthermore,

$$\lim_{\mathcal{H}_m^* \to +\infty, \lambda/m \to 0^+} \frac{T_{\mathrm{MC}}}{\overline{T}^*} = 1.$$

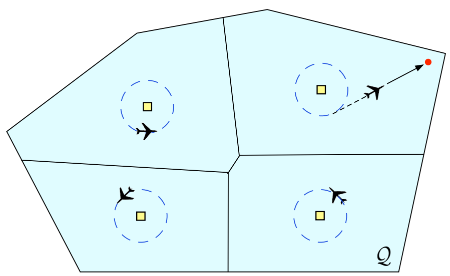
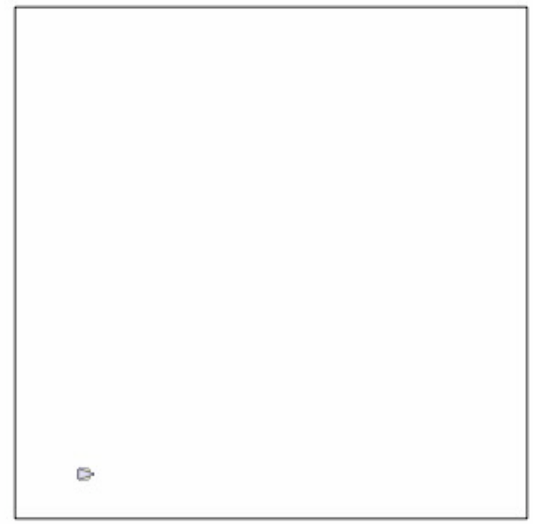


Illustration of the MC policy

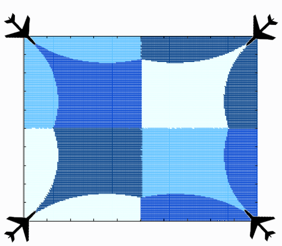


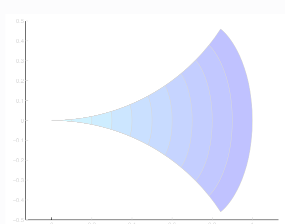
Tighter lower bound using differential constraints

General protocol

- Consider a "frozen moment in time"
- Consider the "modified Voronoi" diagram of the vehicles.
- Relaxation: approximate vehicle Voronoi region by their reachable sets
- Optimize over the vehicle configurations

Example: Dubins vehicle





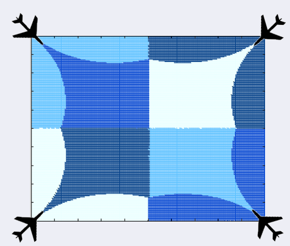
- For $m \geq m_{\mathrm{crit}}$, $\overline{T}^* \geq \frac{k_1(|\mathcal{Q}|,\rho)}{m^{1/3}}$

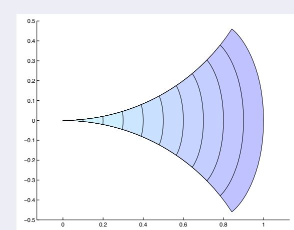
Tighter lower bound using differential constraints

General protocol

- Consider a "frozen moment in time"
- Consider the "modified Voronoi" diagram of the vehicles.
- Relaxation: approximate vehicle Voronoi region by their reachable sets
- Optimize over the vehicle configurations

Example: Dubins vehicle

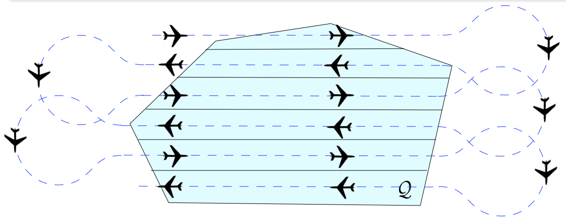


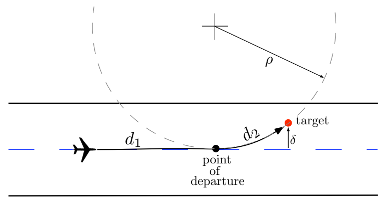


- For $m \geq m_{\mathrm{crit}}$, $\overline{T}^* \geq \frac{k_1(|\mathcal{Q}|,\rho)}{m^{1/3}}$

The Strip Loitering (SL) policy

- Divide the environment $\mathcal{Q}$ into strips of width $\min\left\{\frac{k_2(\mathcal{Q},\rho)}{m^{2/3}}, 2\rho\right\}$
- 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.

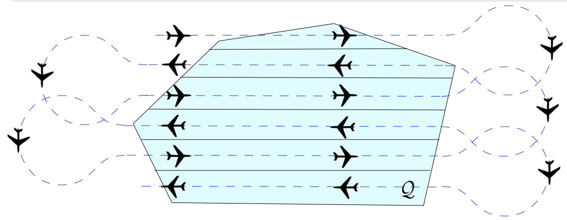


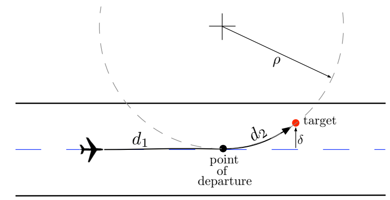


$$\lim_{m\to+\infty} T_{\mathrm{SL}}\, m^{1/3} \le k_3(\mathcal{Q},\rho), \quad \text{and} \quad \lim_{m\to+\infty} \frac{T_{\mathrm{SL}}}{T^*} \le k_4(\mathcal{Q},\rho).$$

The Strip Loitering (SL) policy

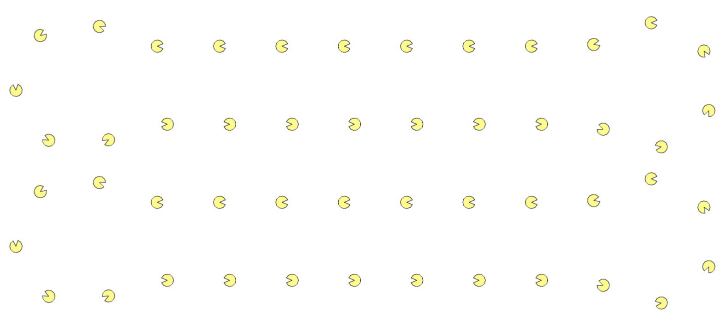
- Divide the environment $\mathcal{Q}$ into strips of width $\min\left\{\frac{k_2(\mathcal{Q},\rho)}{m^{2/3}}, 2\rho\right\}$
- Design a closed loitering path that bisects the strips. All vehicles move along this path, equally spaced, with dynamic regions of responsibility.
- Each vehicle services targets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





$$\lim_{m\to+\infty} T_{\mathrm{SL}}\, m^{1/3} \le k_3(\mathcal{Q},\rho), \quad \text{and} \quad \lim_{m\to+\infty} \frac{T_{\mathrm{SL}}}{T^*} \le k_4(\mathcal{Q},\rho).$$

Illustration of the SL policy



Outline of the lecture

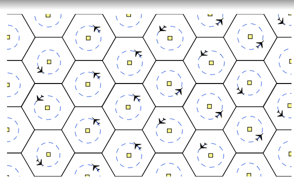
Phase transition in the light load

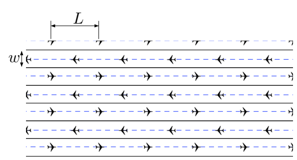
- We have two policies: Median Circling (MC), and Strip Loitering (SL). Which is better?
- Define the non-holonomic density $d_\rho = \frac{\rho^2 m}{|\mathcal{Q}|}$.
  - MC is optimal when $d_\rho \to 0$,
  - SL is within a constant factor of the optimal as $d_\rho \to +\infty$.
- **phase transition:** the optimal organization changes from territorial (MC) to gregarious (SL) depending on the non-holonomic density of the agents.

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 $d_\rho = \frac{\rho^2 m}{|\mathcal{Q}|}$.
  - MC is optimal when $d_\rho \to 0$,
  - SL is within a constant factor of the optimal as $d_\rho \to +\infty$.
- **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



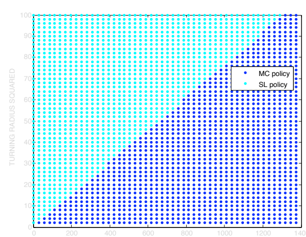


- Ignoring boundary conditions (e.g., consider the unbounded plane), we can compare the coverage cost for the two policies analytically:

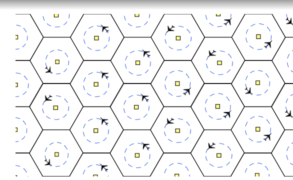
$$T_{\mathrm{SL}} < T_{\mathrm{MC}} \quad \Leftrightarrow \quad d_\rho > 0.0587$$

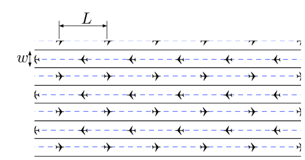
(i.e., transition occurs when the area of the dominance region is about 4-5 times the area of the minimum turning radius circle).

- Simulation results yield $d_\rho^{\mathrm{crit}} \approx 0.0759$ (within a factor 1.3 of the analytical result).



Estimate of the critical density



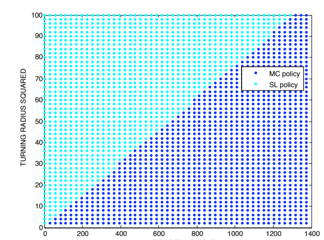


- Ignoring boundary conditions (e.g., consider the unbounded plane), we can compare the coverage cost for the two policies analytically:

$$T_{\mathrm{SL}} < T_{\mathrm{MC}} \quad \Leftrightarrow \quad d_\rho > 0.0587$$

(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 $d_\rho^{\mathrm{crit}} \approx 0.0759$ (within a factor 1.3 of the analytical result).



Dynamic Vehicle Routing Summary

	Euclidean vehicle	Dubins vehicle, Reeds-Shepp car Double integrator, Differential drive
$\mathbb{E}$[TSP Length] $(n \to \infty)$	$\Theta(n^{\frac{1}{2}})$	$\Theta(n^{\frac{2}{3}})$
$\overline{T}^*$ $(\frac{\lambda}{m} \to \infty)$	$\Theta(\frac{\lambda}{m^2})$	$\Theta(\frac{\lambda^2}{m^3})$
$\overline{T}^*$ $(\frac{\lambda}{m} \to 0, \frac{m}{\lvert\mathcal{Q}\rvert} \to 0)$	$\Theta(m^{-\frac{1}{2}})$	$\Theta(m^{-\frac{1}{2}})$
$\overline{T}^*$ $(\frac{\lambda}{m} \to 0, \frac{m}{\lvert\mathcal{Q}\rvert} \to \infty)$	$\Theta(m^{-\frac{1}{2}})$	$\Theta(m^{-\frac{1}{3}})$

Lecture outline

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*	
11:00-11:45pm	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

Dynamic Vehicle Routing for Robotic Networks
Lecture #8: Different Task Models

Francesco Bullo[1] Emilio Frazzoli[2] Marco Pavone[2]
Ketan Savla[2] Stephen L. Smith[2]

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

[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 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

Literature Review

Scaling laws in Robotic Networks

V. Sharma, M. Savchenko, E. Frazzoli, and P. Voulgaris. Transfer time complexity of conflict-free vehicle routing with no communications. *International Journal of Robotics Research*, 26(3):255–272, 2007

S. L. Smith and F. Bullo. Monotonic target assignment for robotic networks. *IEEE Transactions on Automatic Control*, 54(9):2042–2057, 2009

Throughput vs Delay in Wireless Networks

G. Sharma, R. Mazumdar, and N. Shroff. Delay and capacity trade-offs in mobile ad hoc networks: A global perspective. In *IEEE Conf. on Computer Communications*, pages 1–12, Barcelona, Spain, April 2006

A. El Gamal, J. Mammen, B. Prabhakar, and D. Shah. Optimal throughput-delay scaling in wireless networks. Part I: The fluid model. *IEEE Transactions on Information Theory*, 52(6):2568–2592, 2006

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

B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*, volume 21 of *Algorithmics and Combinatorics*. Springer, 4 edition, 2007

Lecture outline

1. Dynamic Team Forming
2. Three Policies
3. Analysis of Policies

Dynamic Team Forming

Set of services $\{r_1, \dots, r_k\}$.

Vehicle properties:

- k different vehicle types.
- Vehicle type $j \in \{1, \dots, k\}$, can provide only service r_j.

Task (demand) model:

- Poisson and Uniform arrivals
- Each task requires a subset of services in $\{r_1, \dots, r_k\}$.
- $\mathcal{K}$ different types of tasks
- Tasks of type α arrive at rate λ_α
- Task completed when required vehicles **simultaneously** spend on-site service time at location.

S. L. Smith and F. Bullo. The dynamic team forming problem: Throughput and delay for unbiased policies. *Systems & Control Letters*, 58(10-11):709–715, 2009

Load Factor and Stability

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

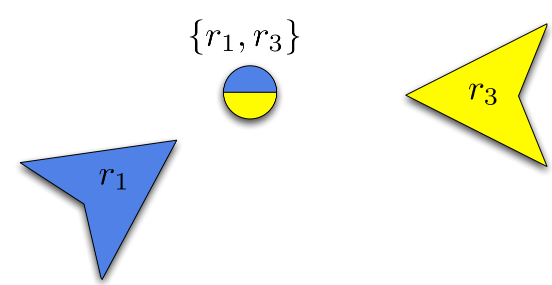
Necessary stability condition:

$$[R_1 \; \cdots \; R_{\mathcal{K}}] \begin{bmatrix} \lambda_1 \bar{s}_1 \\ \vdots \\ \lambda_{\mathcal{K}} \bar{s}_{\mathcal{K}} \end{bmatrix} < \begin{bmatrix} m_1 \\ \vdots \\ m_k \end{bmatrix}$$

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 $j \in \{1, 2, 3, 4\}$.
- $\mathcal{K} = 6$ 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 λ_α, and

$$R_\alpha = [1\;0\;1\;0]^\top.$$

Lecture outline

1. Dynamic Team Forming
2. **Three Policies**
   - Complete Team
   - Task-Specific Team Policy
   - Scheduled Task-Specific Team Policy
3. Analysis of Policies

Task-Type Unbiased Policies

For a policy π:

- System time of each task-type $\overline{T}_{\pi,1}, \dots, \overline{T}_{\pi,\mathcal{K}}$
- Feasible set of system times are subset of $\mathbb{R}^{\mathcal{K}}$
- Optimization space similar to priority queues, but with teaming

To simplify, consider **task-type unbiased policies**

$$\overline{T}_{\pi,1} = \overline{T}_{\pi,2} = \cdots = \overline{T}_{\pi,\mathcal{K}} =: \overline{T}_{\pi}$$

and the optimization: $\inf_{\pi} \overline{T}_{\pi}$.

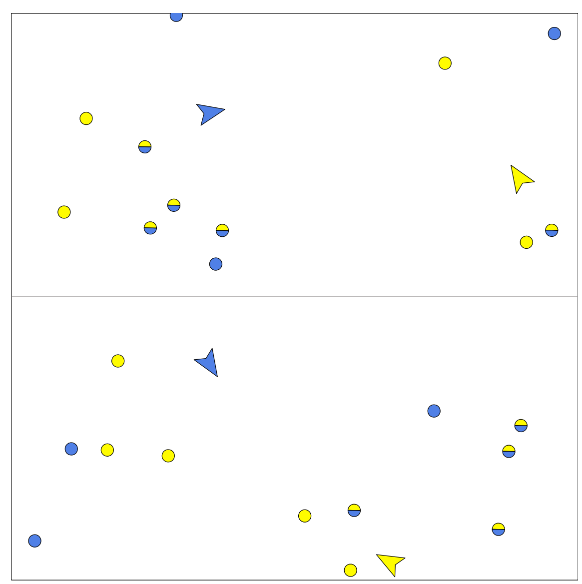
Policy 1: Complete Team Policy

Complete Team Policy

1: Form $\min\{m_1, \dots m_k\}$ 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

Policy 1: Complete Team Policy



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

Policy 2: Task-specific Team Policy

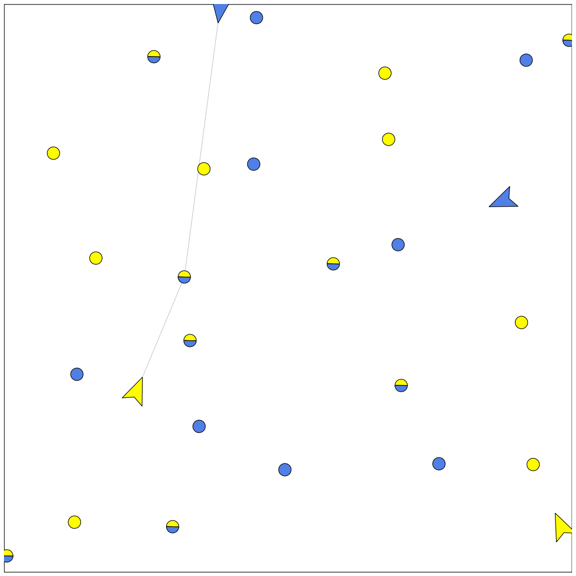
- m_j vehicles provide service r_j.
- r_j appears in $e_j^\top [R_1 \ \cdots \ R_\mathcal{K}] \mathbf{1}_\mathcal{K}$ task types.
- If $m_j \geq e_j^\top [R_1 \ \cdots \ R_\mathcal{K}] \mathbf{1}_\mathcal{K} \Rightarrow$ enough vehicles of type j to create dedicated team for each task type.
- Create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

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

Task-Specific Team Policy

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

Policy 2: Task-Specific Team Policy



- task types: $\{y\}, \{b\}, \{y, b\}$
- two vehicles of each type
- y, b each appear in two task-types

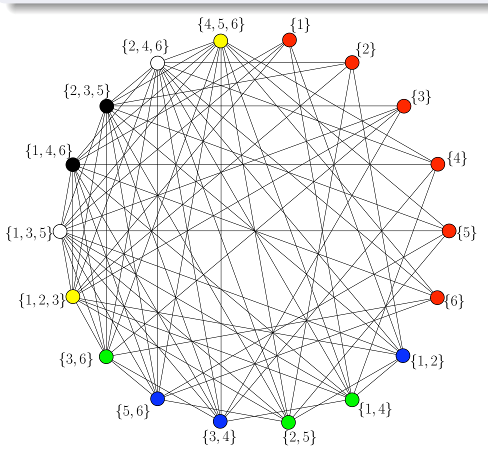
$$m_{\text{TST}} = 1$$

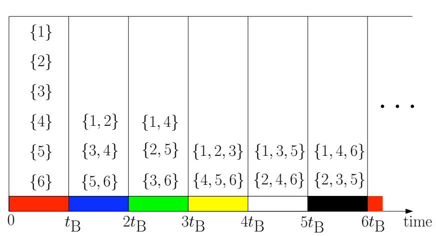
Policy 3: Preliminary Result

Definition (Service schedule)

A partition of task types into L time slots, such that:

- each type appears in exactly one time slot, and
- task types in each time slot are pairwise disjoint.





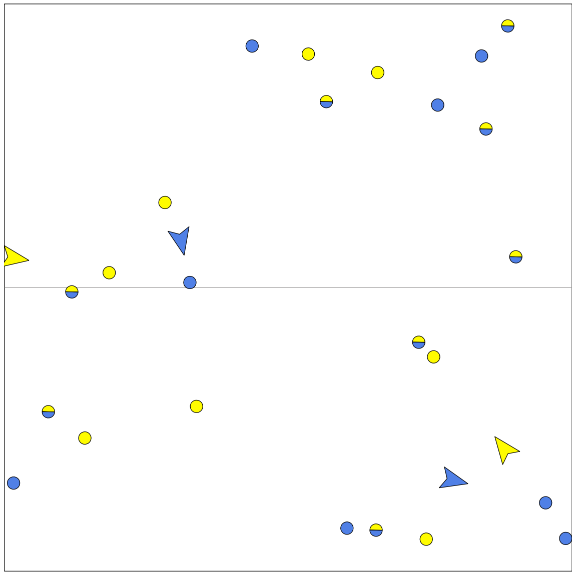
Policy 3: Scheduled Task-Specific Team Policy

Scheduled Task-specific team policy

Partition $\mathcal{Q}$ 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:
  1. Divide robots into teams to form required task types.
  2. 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: $\{y, b\}$

Lecture outline

Assumptions for Analysis

Assumptions:

1. $m_i = m/k$ for each vehicle type i.
2. $\lambda_\alpha = \lambda/\mathcal{K}$ for each task-type α.
3. on-site service has mean $\bar{s}$ and is upper bounded by $s_{\max}$.
4. $p\mathcal{K}$ of the $\mathcal{K}$ 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

$$B_m \mapsto \begin{cases} \max\left\{\overline{T}_{\min}, \dfrac{\overline{T}_{\mathrm{ord}}(B_m/B_{\mathrm{crit}})}{(1 - B_m/B_{\mathrm{crit}})^2}\right\}, & \text{if } B_m < B_{\mathrm{crit}}, \\ +\infty, & \text{if } B_m \geq B_{\mathrm{crit}}. \end{cases}$$

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

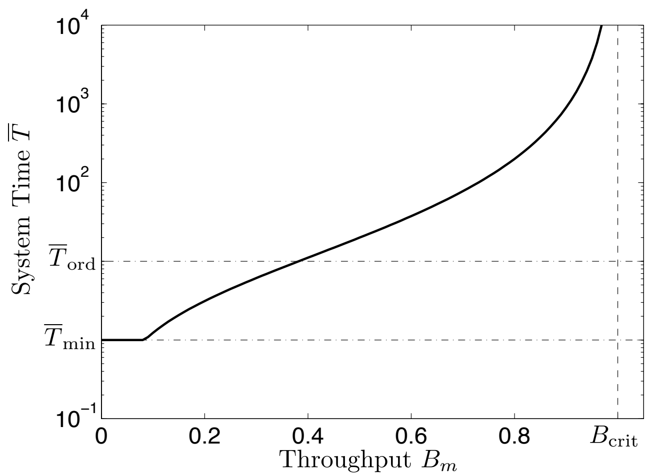
Example (Single vehicle DVR)

$B_{\mathrm{crit}} = 1/\bar{s}$

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

$\overline{T}_{\mathrm{ord}} \approx C(\int_Q \varphi^{1/2}(x)dx)^2/v^2$ (heavy load numerator)

Throughput vs System Time Profile



System Time for each Policy

	$\overline{T}_{\mathrm{min}}$	$\overline{T}_{\mathrm{ord}}$	B_{crit}
Lower bound ($\overline{T}^*$)	$\sqrt{k}$	k	$\frac{1}{pk\bar{s}}$
Complete Team	$\sqrt{k}$	k	$\frac{1}{k\bar{s}}$
Task-Specific	$\sqrt{pk\mathcal{K}}$	$pk\mathcal{K}$	$\frac{1}{pk\bar{s}}$
Scheduled Task-Specific	$L\sqrt{k}$	Lk	$\frac{\mathcal{K}}{s_{\mathrm{max}}Lk}$

where $L \in [p\mathcal{K}, \mathcal{K}]$

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

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*	
11:00-11:45pm	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