Dynamic Vehicle Routing for Robotic Systems

Francesco Bullo Emilio Frazzoli Marco Pavone Ketan Savla Stephen L. Smith

***Abstract*—Recent years have witnessed great advancements in the science and technology of autonomy, robotics and networking. This paper surveys recent concepts and algorithms for dynamic vehicle routing (DVR), that is, for the automatic planning of optimal multi-vehicle routes to perform tasks that are generated over time by an exogenous process. We consider a rich variety of scenarios relevant for robotic applications. We begin by reviewing the basic DVR problem: demands for service arrive at random locations at random times and a vehicle travels to provide on-site service while minimizing the expected wait time of the demands. Next, we treat different multi-vehicle scenarios based on different models for demands (e.g., demands with different priority levels and impatient demands), vehicles (e.g., motion constraints, communication and sensing capabilities), and tasks. The performance criterion used in these scenarios is either the expected wait time of the demands or the fraction of demands serviced successfully. In each specific DVR scenario, we adopt a rigorous technical approach that relies upon methods from queueing theory, combinatorial optimization and stochastic geometry. First, we establish fundamental limits on the achievable performance, including limits on stability and quality of service. Second, we design algorithms, and provide provable guarantees on their performance with respect to the fundamental limits.**

I. Introduction

This survey presents a joint algorithmic and queueing approach to the design of cooperative control and task allocation strategies for networks of uninhabited vehicles and robots. The approach enables groups of robots to complete tasks in uncertain and dynamically changing environments, where new task requests are generated in real-time. Applications include surveillance and monitoring missions, as well as transportation networks and automated material handling.

As a motivating example, consider the following scenario: a sensor network is deployed in order to detect suspicious activity in a region of interest. (Alternatively, the sensor network is replaced by a high-altitude sensory-rich aircraft loitering over the region.) In addition to the sensor network, a team of unmanned aerial vehicles (UAVs) is available and each UAV is equipped with close-range high-resolution on-board sensors. Whenever a sensor detects a potential event, a request for close-range observation by one of the UAVs is generated. In response to this request, a UAV visits the location to gather close-range information and investigates the cause of the alarm. Each request for close-range observation might include priority levels or time windows during which the inspection must occur and it might require an on-site service time. In summary, from a control algorithmic viewpoint, each time a new request arises, the UAVs need to decide which vehicle will inspect that location and along which route. Thus, the problem is to design algorithms that enable real-time task allocation and vehicle routing.

Accordingly, this paper surveys allocation and routing algorithms that typically blend ideas from receding-horizon resource allocation, distributed optimization, combinatorics and control. The key novelty in our approach is the simultaneous introduction of stochastic, combinatorial and queueing aspects in the distributed coordination of robotic networks.

Static vehicle routing: In the recent past, considerable efforts has been devoted to the problem of how to cooperatively assign and schedule demands for service that are defined over an extended geographical area [1], [2], [3], [4], [5]. In these papers, the main focus is in developing distributed algorithms that operate with knowledge about the demands locations and with limited communication between robots. However, the underlying mathematical model is *static*, in that no new demands arrive over time. Thus, the centralized version of the problem fits within the framework of the static vehicle routing problem (see [6] for a thorough introduction to this problem, known in the operations research literature as the Vehicle Routing Problem (VRP)), whereby: (i) a team of m vehicles is required to service a set of n demands in a 2-dimensional space; (ii) each demand requires a certain amount of on-site service; (iii) the goal is to compute a set of routes that optimizes the cost of servicing (according to some quality of service metric) the demands. In general, most of the available literature on routing for robotic networks focuses on static environments and does not properly account for scenarios in which dynamic, stochastic and adversarial events take place.

Dynamic vehicle routing: The problem of planning routes through service demands that arrive *during* a mission execution is known as the "dynamic vehicle routing problem" (abbreviated as the DVR problem). See Figure 1 for an illustration of DVR. There are two key differences between static and dynamic vehicle routing problems. First, planning algorithms should actually provide *policies* (in contrast to pre-planned routes) that prescribe how the routes should evolve as a function of those inputs that evolve in real-time. Second, dynamic demands (i.e., demands that vary over time) add *queueing phenomena* to the combinatorial nature of vehicle

Submitted to the *Proceedings of the IEEE* in May 2010, revised in February 2011. This research was partially supported by AFOSR award FA 8650-07-2-3744, ARO MURI award W911NF-05-1-0219, NSF awards ECCS-0705451 and CMMI-0705453, and ARO award W911NF-11-1-0092.

F. Bullo is with the Center for Control, Dynamical Systems and Computation and with the Department of Mechanical Engineering, University of California, Santa Barbara, CA 93106 (bullo@engineering.ucsb.edu).

E. Frazzoli is with the Laboratory for Information and Decision Systems, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139 (frazzoli@mit.edu).

M. Pavone is with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109 (marco.pavone@jpl.nasa.gov).

K. Savla is with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 (ksavla@mit.edu).

S. L. Smith is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo ON, N2L 3G1 Canada (stephen.smith@uwaterloo.ca).

The authors are listed in alphabetical order.

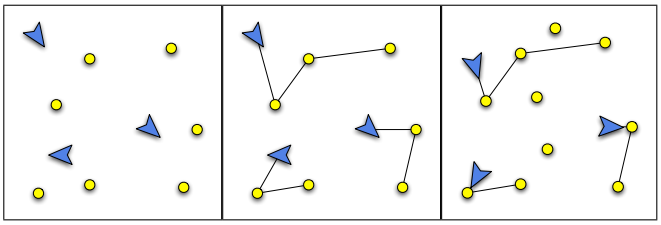

Fig. 1. An illustration of *dynamic vehicle routing for a robotic system*. From panel #1 to #2: vehicles are assigned to customers and select routes. Panel #3: the DVR problem is how to re-allocate and re-plan routes when new customers appear.

routing. In such a dynamic setting, it is natural to focus on steady-state performance instead of optimizing the performance for a single demand. Additionally, system stability in terms of the number of waiting demands is an issue to be addressed.

Algorithmic queueing theory for DVR: The objective of this work is to present a joint algorithmic and queueing approach to the design of cooperative control and task allocation strategies for *networks of uninhabited vehicles* required to operate in dynamic and uncertain environments. This approach is based upon the pioneering work of Bertsimas and Van Ryzin [7], [8], [9], who introduced queueing methods to solve the simplest DVR problem (a vehicle moves along straight lines and visits demands whose time of arrival, location and on-site service are stochastic; information about demand location is communicated to the vehicle upon demand arrival); see also the earlier related work [10].

Starting with these works [7], [8], [9] and integrating ideas from dynamics, combinatorial optimization, teaming, and distributed algorithms, we have recently developed a systematic approach to tackle complex dynamic routing problems for robotic networks. We refer to this approach as "algorithmic queueing theory" for dynamic vehicle routing. The power of algorithmic queueing theory stems from the wide spectrum of aspects, critical to the routing of robotic networks, for which it enables a rigorous study; specific examples taken from our work in the past few years include complex models for the demands such as time constraints [11], [12], service priorities [13], and translating demands [14], problems concerning robotic implementation such as adaptive and decentralized algorithms [15], [16], complex vehicle dynamics [17], [18], limited sensing range [19], and team forming [20], and even integration of humans in the design space [21].

Survey content: In this work we provide a detailed account of algorithmic queueing theory for DVR, with an emphasis on robotic applications. We start in Section II by reviewing the possible approaches to dynamic vehicle routing problems. Then, in Section III, we describe the foundations of algorithmic queueing theory, which lie on the aforementioned works of Bertsimas and Van Ryzin. In the following four sections we discuss some of our recent efforts in applying algorithmic queueing theory to realistic dynamic routing problems for robotic systems.

Specifically, in Section IV we present routing policies for DVR problems that (i) are spatially distributed, scalable to large networks, and adaptive to network changes, (ii) have remarkably good performance guarantees in both the light-load regime (i.e., when the arrival rate for the demands is small) and in the heavy-load regime (i.e., when the arrival rate for the demands is large). Here, by network changes we mean changes in the number of vehicles, the arrival rate of demands, and the characterization of the on-site service requirement.

In Section V we discuss time-constrained and prioritized service. For time-constrained DVR problems, we establish upper and lower bounds on the optimal number of vehicles for a given level of service quality (defined as the desired fraction of demands that must receive service within the deadlines). Additionally, we rigorously characterize two service policies: in light load the DVR problem with time constraints is closely related to a particular facility location problem, and in moderate and heavy load, static vehicle routing methods, such as solutions of traveling salesman problems, can provide good performance. We then study DVR problems in which demands have an associated level of priority (or importance). The problem is characterized by the number of different priority classes n and their relative levels of importance. We provide lower bounds on the optimal performance and a service policy which is guaranteed to perform within a factor $2n^2$ of the optimal in the heavy-load.

We then study the implications of vehicle motion constraints in Section VI. We focus on the Dubins vehicle, namely, a nonholonomic vehicle that is constrained to move along paths of bounded curvature without reversing direction. For m Dubins vehicles, the DVR problem with arrival rate λ and with uniform spatial distribution has the following properties: the system time is (i) of the order λ^2/m^3 in heavy-load, (ii) of the order $1/\sqrt{m}$ in the light-load if the vehicle density is small, and of the order $1/\sqrt[3]{m}$ in the light-load if the density of the vehicles is high.

In Section VII we discuss the case when vehicles are heterogeneous, each capable of providing a specific type of service. Each demand may require several different services, implying that collaborative teams of vehicles must be formed to service a demand. We present three simple policies for this problem. For each policy we show that there is a broad class of system parameters for which the policy's performance is within a constant factor of the optimal.

Finally, in Section VIII we summarize other recent results in DVR and draw our conclusions.

II. Algorithmic Approaches to DVR Problems

In this section we review possible approaches to DVR problems and motivate our proposed algorithmic queueing theory approach.

A. On the Adaptation of Queueing Policies and Static Methods

A naive, yet reasonable approach to DVR problems would be to adapt classic queueing policies to spatial queueing systems. However, perhaps surprisingly, this adaptation is not at all straightforward. For example, policies based on a First-Come First-Served discipline, whereby tasks are fulfilled in the order in which they arrive, are unable to stabilize the system for all possible task arrival rates, in the sense that under such

policies the average number of tasks grows over time without bound [7, page 608].

A second possibility is to combine static routing methods (e.g., nearest neighbor or VRP-like methods) and sequential re-optimization algorithms. This approach, indeed, will be at the core of most of the policies we present in this work. However, the joint selection of a static routing method and of the re-optimization horizon in presence of robot and task constraints (e.g., differential motion constraints or task priorities) makes the application of this approach far from trivial. For example, one can show that an erroneous selection of the re-optimization horizon can lead to pathological scenarios where no task receives service [22]. Likewise, direct application of VRP-like methods might lead to infeasible paths for vehicles with differential motion constraints. Finally, performance criteria in dynamic settings commonly differ from those of the corresponding static problems (e.g., in a dynamic setting, the wait for service delivery might be a more important factor than the total vehicle travel cost).

The general conclusion is that DVR problems require *ad hoc* routing algorithms together with tailored performance analyses. There are currently two main algorithmic approaches that allow *both* a rigorous synthesis and a performance analysis of routing algorithms for DVR problems; we review these two approaches next.

B. Online Algorithms

An online algorithm is one that operates based on input information given up to the current time. Thus, these algorithms are designed to operate in scenarios where the entire input is not known at the outset, and new pieces of the input should be incorporated as they become available. The distinctive feature of the online algorithm approach is the method that is used to evaluate the performance of online algorithms, which is called *competitive analysis* [23]. In competitive analysis, the performance of an online algorithm is compared to the performance of a corresponding offline algorithm (i.e., an algorithm that has *a priori* knowledge of the entire input) in the worst case scenario. Specifically, an online algorithm is c-competitive if its cost on *any* problem instance is at most c times the cost of an optimal offline algorithm:

$$\text{Cost}_{\text{online}}(I) \leq c\, \text{Cost}_{\text{optimal offline}}(I), \quad \forall \text{ problem instances } I.$$

In the recent past, dynamic vehicle routing problems have been studied in this framework, under the name of the online traveling repairman problem [24], [25], [26].

While the online algorithm approach applied to DVR has led to numerous results and interesting insights, it leaves some questions unanswered, especially in the context of robotic networks. First, competitive analysis is a *worst-case* analysis, hence, the results are often overly pessimistic for normal problem instances. Moreover, in many applications there is some probabilistic problem structure (e.g., distribution of the inter-arrival times, spatial distribution of future demands, distribution of on-site service times etc.), that can be advantageously exploited by the vehicles. In online algorithms, this additional information is not taken into account. Second, competitive analysis is used to bound the performance relative to the optimal offline algorithm, and thus it does not give an absolute measure of performance. In other words, an optimal online algorithm is an algorithm with minimum "cost of causality" in the worst-case scenario, but not necessarily with the minimum worst-case cost. Finally, many important real-world constraints for DVR, such as time windows, priorities, differential constraints on vehicle's motion and the requirement of teams to fulfill a demand "have so far proved to be too complex to be considered in the online framework" [27, page 206]. Some of these drawbacks have been recently addressed by [28] where a combined stochastic and online approach is proposed for a general class of combinatorial optimization problems and is analyzed under some technical assumptions.

This discussion motivates an alternative approach for DVR in the context of robotic networks, based on probabilistic modeling, and average-case analysis.

C. Algorithmic Queueing Theory

Algorithmic queueing theory embeds the dynamic vehicle routing problem within the framework of queueing theory and overcomes most of the limitations of the online algorithm approach; in particular, it allows to take into account several real-world constraints, such as time constraints and differential constraints on vehicles' dynamics. We call this approach *algorithmic queueing theory* since its objective is to *synthesize* an efficient control policy, whereas in traditional queueing theory the objective is usually to *analyze* the performance of a specific policy. Here, an efficient policy is one whose *expected* performance is either optimal or optimal within a constant factor.[1] Algorithmic queueing theory basically consists of the following steps:

(i) queueing model of the robotic system and analysis of its structure;
(ii) establishment of fundamental limitations on performance, independent of algorithms; and
(iii) design of algorithms that are either optimal or constant-factor away from optimal, possibly in specific asymptotic regimes.

Finally, the proposed algorithms are evaluated via numerical, statistical and experimental studies, including Monte-Carlo comparisons with alternative approaches.

In order to make the model tractable, customers are usually considered "statistically independent" and their arrival process is assumed stationary (with possibly unknown parameters). Because these assumptions can be unrealistic in some scenarios, this approach has its own limitations. The aim of this paper is to show that algorithmic queueing theory, despite these disadvantages, is a very useful framework for the design of routing algorithms for robotic networks and a valuable complement to the online algorithm approach.

[1]The expected performance of a policy is the expected value of the performance over all possible inputs (i.e., demand arrival sequences). A policy performs within a constant factor κ of the optimal if the ratio between the policy's expected performance and the optimal expected performance is upper bounded by κ.

III. Algorithmic Queueing Theory for DVR

In this section we describe algorithmic queueing theory. We start with a short review of some fundamental concepts from the locational optimization literature, and then we introduce the general approach.

A. Preliminary Tools

The Euclidean Traveling Salesman Problem (in short, TSP) is formulated as follows: given a set D of n points in $\mathbb{R}^d$, find a minimum-length tour (i.e., a closed path that visits all points exactly once) of D. More properties of the TSP tour can be found in Section A of the Appendix. In this paper, we will present policies that require real-time solutions of TSPs over possibly large point sets; this can indeed be achieved by using efficient approximation algorithms presented in Section B of the Appendix.

Let $\mathcal{Q} \subset \mathbb{R}^2$ be a bounded, convex set (the following concepts can be similarly defined in higher dimensions). Let $P = (p_1, \ldots, p_m)$ be an array of m distinct points in $\mathcal{Q}$. The *Voronoi diagram* of $\mathcal{Q}$ generated by P is an array of sets, denoted by $\mathcal{V}(P) = (V_1(P), \ldots, V_m(P))$, defined by

$$V_i(P) = \{x \in \mathcal{Q} | \, \|x - p_i\| \leq \|x - p_j\|, \forall j \in \{1, \ldots, m\}\},$$

where $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^2$. We refer to P as the set of *generators* of $\mathcal{V}(P)$, and to $V_i(P)$ as the Voronoi cell or the region of dominance of the ith generator.

The expected distance between a random point q, generated according to a probability density function $\varphi : \mathcal{Q} \to \mathbb{R}_{\geq 0}$, and the closest point in P is given by

$$H_m(P, \mathcal{Q}) := \mathbb{E}\left[\min_{k \in \{1,\ldots,m\}} \|p_k - q\|\right].$$

The function H_m is known in the locational optimization literature as the continuous Weber function or the continuous multi-median function; see [29], [30] and the references therein. The m-median of the set $\mathcal{Q}$ with density φ is the global minimizer

$$P_m^*(\mathcal{Q}) = \arg\min_{P \in \mathcal{Q}^m} H_m(P, \mathcal{Q}).$$

We let $H_m^*(\mathcal{Q}) = H_m(P_m^*(\mathcal{Q}), \mathcal{Q})$ be the global minimum of H_m. The set of critical points of H_m contains all arrays $(p_1, \ldots, p_m)$ with distinct entries and with the property that each point p_k is simultaneously the generator of the Voronoi cell $V_k(P)$ and the median of $V_k(P)$. We refer to such Voronoi diagrams as *median Voronoi diagrams*. It is possible to show that a median Voronoi diagram always exists for any bounded convex domain $\mathcal{Q}$ and density φ. More properties of the multi-median function are discussed in Section C of the Appendix.

B. Queueing Model for DVR

Here we review the model known in the literature as the m-vehicle Dynamic Traveling Repairman Problem (m-DTRP) and introduced in [7], [8].

Consider m vehicles free to move, at a constant speed v, within $\mathbb{R}^2$. The extension to higher-dimensional setups is straightforward unless otherwise noted. On the other hand, constraints on the motion of the vehicles (e.g., obstacles) require in general non-trivial extensions of our approach, and our results do not necessarily hold.

Demands are generated in a bounded and convex set $\mathcal{Q}$, called the *environment*, according to a homogeneous (i.e., time-invariant) spatio-temporal Poisson process, with time intensity $\lambda \in \mathbb{R}_{>0}$, and spatial density $\varphi : \mathcal{Q} \to \mathbb{R}_{>0}$. In other words, demands arrive to $\mathcal{Q}$ according to a Poisson process with intensity λ, and their locations $\{X_j;\ j \geq 1\}$ are i.i.d. (i.e., *independent* and *identically distributed*) and distributed according to a density φ whose support is $\mathcal{Q}$. Many results in this paper extend to the case in which $\mathcal{Q}$ is not convex, and we refer the interested reader to our full-length papers for more details.

A demand's location becomes known (is realized) at its arrival epoch; thus, at time t we know with *certainty* the locations of demands that arrived prior to time t, but future demand locations form an i.i.d. sequence. The density φ satisfies:

$$\mathbb{P}\left[X_j \in S\right] = \int_S \varphi(x)\, dx \quad \forall S \subseteq \mathcal{Q}, \quad \text{and} \quad \int_{\mathcal{Q}} \varphi(x)\, dx = 1.$$

At each demand location, vehicles spend some time $s \geq 0$ in on-site service that is i.i.d. and generally distributed with finite first and second moments denoted by $\bar{s} > 0$ and $\overline{s^2}$. A realized demand is removed from the system after one of the vehicles has completed its on-site service. We define the *load factor* $\varrho := \lambda \bar{s}/m$.

The system time of demand j, denoted by T_j, is defined as the elapsed time between the arrival of demand j and the time one of the vehicles completes its service. The waiting time of demand j, W_j, is defined by $W_j = T_j - s_j$. The steady-state system time is defined by $\overline{T} := \limsup_{j \to \infty} \mathbb{E}\left[T_j\right]$. A policy for routing the vehicles is said to be *stable* if the expected number of demands in the system is uniformly bounded at all times. A necessary condition for the existence of a stable policy is that $\varrho < 1$; we shall assume $\varrho < 1$ throughout the paper. When we refer to *light-load* conditions, we consider the case $\varrho \to 0^+$, in the sense that $\lambda \to 0^+$; when we refer to *heavy-load* conditions, we consider the case $\varrho \to 1^-$, in the sense that $\lambda \to (m/\bar{s})^-$.

Let $\mathcal{P}$ be the set of all causal, stable, and time-invariant routing policies and $\overline{T}_\pi$ be the system time of a particular policy $\pi \in \mathcal{P}$. The m-DTRP is then defined as the problem of finding a policy $\pi^* \in \mathcal{P}$ (if one exists) such that

$$\overline{T}^* := \overline{T}_{\pi^*} = \inf_{\pi \in \mathcal{P}} \overline{T}_\pi.$$

In general, it is difficult to characterize the optimal achievable performance $\overline{T}^*$ and to compute the optimal policy π^* for arbitrary values of the problem parameters λ, m, etc. It is instead possible and useful to consider particular ranges of parameter values and, specifically, asymptotic regimes such as the light-load and the heavy-load regimes. For the purpose of characterizing asymptotic performance, we briefly review some useful notation. For $f, g : \mathbb{N} \to \mathbb{R}$, $f \in O(g)$ (respectively, $f \in \Omega(g)$) if there exist $N_0 \in \mathbb{N}$ and $k \in \mathbb{R}_{>0}$ such that $|f(N)| \leq k|g(N)|$ for all $N \geq N_0$ (respectively, $|f(N)| \geq k|g(N)|$ for all $N \geq N_0$). If $f \in O(g)$ and $f \in \Omega(g)$, then the notation $f \in \Theta(g)$ is used.

C. Lower Bounds on the System Time

As in many queueing problems, the analysis of the DTRP problem for all the values of the load factor ϱ in $(0, 1)$ is difficult. In [7], [8], [9], [31], lower bounds for the optimal steady-state system time are derived for the light-load case (i.e., $\varrho \to 0^+$), and for the heavy-load case (i.e., $\varrho \to 1^-$). Subsequently, policies are designed for these two limiting regimes, and their performance is compared to the lower bounds.

For the light-load case, a tight lower bound on the system time is derived in [8]. In the light-load case, the lower bound on the system time is strongly related to the solution of the m-median problem:

$$\overline{T}^* \geq \frac{1}{v} H_m^*(\mathcal{Q}) + \bar{s}, \qquad \text{as } \varrho \to 0^+. \tag{1}$$

The bound is tight: there exist policies whose system times, in the limit $\varrho \to 0^+$, attain this bound; we present such asymptotically optimal policies for the light-load case below.

Two lower bounds exist for the heavy-load case [9], [31] depending on whether one is interested in *biased* policies or *unbiased* policies.

Definition III.1 (Spatially biased and unbiased policies)**.** *Let X be the location of a randomly chosen demand and W be its wait time. A policy π is said to be*

(i) spatially unbiased *if for every pair of sets $\mathcal{S}_1$, $\mathcal{S}_2 \subseteq \mathcal{Q}$*

$$\mathbb{E}\left[W \,|X \in \mathcal{S}_1\right] = \mathbb{E}\left[W \,|X \in \mathcal{S}_2\right]; \quad \textit{and}$$

(ii) spatially biased *if there exist sets $\mathcal{S}_1$, $\mathcal{S}_2 \subseteq \mathcal{Q}$ such that*

$$\mathbb{E}\left[W \,|X \in \mathcal{S}_1\right] > \mathbb{E}\left[W \,|X \in \mathcal{S}_2\right].$$

Within the class of spatially *unbiased* policies in $\mathcal{P}$, the optimal system time is lower bounded by

$$\overline{T}_{\mathrm{U}}^* \geq \frac{\beta_{\mathrm{TSP},2}^2}{2} \frac{\lambda \left(\int_{\mathcal{Q}} \varphi^{1/2}(x)dx\right)^2}{m^2\, v^2\, (1-\varrho)^2} \qquad \text{as } \varrho \to 1^-, \tag{2}$$

where $\beta_{\mathrm{TSP},2} \simeq 0.7120 \pm 0.0002$ (for more detail on the constant $\beta_{\mathrm{TSP},2}$, we refer the reader to Appendix A).

Within the class of spatially *biased* policies in $\mathcal{P}$, the optimal system time is lower bounded by

$$\overline{T}_{\mathrm{B}}^* \geq \frac{\beta_{\mathrm{TSP},2}^2}{2} \frac{\lambda \left(\int_{\mathcal{Q}} \varphi^{2/3}(x)dx\right)^3}{m^2\, v^2\, (1-\varrho)^2} \qquad \text{as } \varrho \to 1^-. \tag{3}$$

Both bounds (2) and (3) are tight: there exist policies whose system times, in the limit $\varrho \to 1^-$, attain these bounds; therefore the inequalities in (2) and (3) could indeed be replaced by equalities. We present asymptotically optimal policies for the heavy-load case below. It is shown in [9] that the lower bound in equation (3) is always less than or equal to the lower bound in equation (2) for all densities φ.

We conclude with some remarks. First, it is possible to show (see [9], Proposition 1) that a *uniform* spatial density function leads to the *worst possible performance* and that any deviation from uniformity in the demand distribution will strictly lower the optimal mean system time in both the unbiased and biased case. Additionally, allowing biased service results in a strict reduction of the optimal expected system time for any non-uniform density φ. Finally, when the density is uniform there is nothing to be gained by providing biased service.

D. Centralized and Ad-Hoc Policies

In this section we present centralized, ad-hoc policies that are *either* optimal in light-load *or* optimal in heavy-load. Here, we say that a policy is ad-hoc if it performs "well" only for a limited range of values of ϱ. In light-load, the SQM policy provides optimal performance (i.e., $\lim_{\varrho\to 0^+} \overline{T}_{\mathrm{SQM}}/\overline{T}^* = 1$):

> **The m Stochastic Queue Median (SQM) Policy [8]** — Locate one vehicle at each of the m median locations for the environment $\mathcal{Q}$. When demands arrive, assign them to the vehicle corresponding to the nearest median location. Have each vehicle service its respective demands in First-Come, First-Served (FCFS) order returning to its median after each service is completed.

This policy, although optimal in light-load, has two characteristics that limit its application to robotic networks: First, it quickly becomes unstable as the load increases, i.e., there exists $\varrho_c < 1$ such that for all $\varrho > \varrho_c$ the system time $\overline{T}_{\mathrm{SQM}}$ is infinite (hence, this policy is *ad-hoc*). Second, a central entity needs to compute the m-median locations and assign them to the vehicles (hence, from this viewpoint the policy is centralized).

In heavy-load, the UTSP policy provides optimal unbiased performance (i.e., $\lim_{\varrho\to 1^-} \overline{T}_{\mathrm{UTSP}}/\overline{T}_{\mathrm{U}}^* = 1$):

> **The Unbiased TSP (UTSP) Policy [9]** — Let r be a fixed positive, large integer. From a central point in the interior of $\mathcal{Q}$, subdivide the service region into r wedges $\mathcal{Q}_1, \ldots, \mathcal{Q}_r$ such that $\int_{\mathcal{Q}_k} \varphi(x)dx = 1/r$, $k \in \{1, \ldots, r\}$. Within each subregion, form sets of demands of size n/r (n is a design parameter). As sets are formed, deposit them in a queue and service them FCFS with the first available vehicle by forming a TSP on the set and following it in an arbitrary direction. Optimize over n (see [9] for details).

It is possible to show that, as $\varrho \to 1^-$,

$$\overline{T}_{\mathrm{UTSP}} \leq \left(1 + \frac{m}{r}\right) \frac{\beta_{\mathrm{TSP},2}^2}{2} \frac{\lambda \left(\int_{\mathcal{Q}} \varphi^{1/2}(x)dx\right)^2}{m^2\, v^2\, (1-\varrho)^2}; \tag{4}$$

thus, letting $r \to \infty$, the lower bound in (2) is achieved.

The same paper [9] presents an optimal biased policy. This policy, called Biased TSP (BTSP) Policy, relies on an even finer partition of the environment and requires φ to be piecewise constant.

Although both the UTSP and the BTSP policies are optimal within their respective classes, they have two characteristics that limit their application to robotic networks: First, in the UTSP policy, to ensure stability, n should be chosen so that (see [9], page 961)

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

therefore, to ensure stability over a wide range of values of ϱ, the system designer is forced to select a *large* value for n. However, if during the execution of the policy the load factor turns out to be only moderate, demands have to wait for an excessively large set to be formed, and the overall system performance deteriorates significantly. Similar considerations hold for the BTSP policy. Hence, these two policies are *ad-hoc*. Second, both policies require a centralized data structure (the demands' queue is shared by the vehicles); hence, both policies are centralized.

Remark III.2 (System time bounds in heavy-load with zero service time)**.** *If $\bar{s} = 0$, then the heavy-load regime is defined as $\lambda/m \to +\infty$, and all the performance bounds we provide in this and in the next two sections hold by simply substituting $\varrho = 0$. For example, equation* (2) *reads*

$$\overline{T}_U^* \geq \frac{\beta_{\mathrm{TSP},2}^2}{2} \frac{\lambda \left(\int_{\mathcal{Q}} \varphi^{1/2}(x) dx \right)^2}{m^2 v^2} \qquad \text{as } \lambda/m \to +\infty. \ \square$$

IV. Routing for Robotic Networks: Decentralized and Adaptive Policies

In this section we first discuss routing algorithms that are both adaptive and amenable to decentralized implementation; then, we present a decentralized and adaptive routing algorithm that does not require any explicit communication between the vehicles while still being optimal in the light-load case.

A. Decentralized and Adaptive Policies

Here, we say that a policy is adaptive if it performs "well" for every value of ϱ in the range $[0, 1)$. A candidate decentralized and adaptive control policy is the simple Nearest Neighbor (NN) policy: at each service completion epoch, each vehicle chooses to visit next the closest unserviced demand, if any, otherwise it stops at the current position. Because of the dependencies among the inter-demand travel distances, the analysis of the NN policy is difficult and no rigorous results have been obtained so far [7]; in particular, there are no rigorous results about its stability properties. Simulation experiments show that the NN policy performs like a biased policy and is not optimal in the light-load case or in the heavy-load case [7], [9]. Therefore, the NN policy *lacks* provable performance guarantees (in particular about stability), and does *not* seem to achieve optimal performance in light-load or in heavy-load.

In [15], we study decentralized and adaptive routing policies that are optimal in light-load and that are optimal unbiased algorithms in heavy-load. The key idea we pursue is that of *partitioning policies*:

Definition IV.1 (Partitioning policies)**.** *Given a policy π for the 1-DTRP and m vehicles, a π-partitioning policy is a family of multi-vehicle policies such that*

(i) *the environment $\mathcal{Q}$ is partitioned into m openly disjoint subregions $\mathcal{Q}_k$, $k \in \{1, \ldots, m\}$, whose union is $\mathcal{Q}$,*

(ii) *one vehicle is assigned to each subregion (thus, there is a one-to-one correspondence between vehicles and subregions),*

(iii) *each vehicle executes the single-vehicle policy π in order to service demands that fall within its own subregion.*

Because Definition IV.1 does not specify how the environment is actually partitioned, it describes a *family* of policies (one for each partitioning strategy) for the m-DTRP. The SQM policy, which is optimal in light-load, is indeed a partitioning policy whereby $\mathcal{Q}$ is partitioned according to a median Voronoi diagram and each vehicle executes inside its own Voronoi region the policy "service FCFS and return to the median after each service completion." Moreover, specific partitioning policies, which will be characterized in Theorem IV.2, are optimal or within a constant factor of the optimal in heavy-load.

In the following, given two functions $\varphi_j : \mathcal{Q} \to \mathbb{R}_{>0}$, $j \in \{1, 2\}$, with $\int_{\mathcal{Q}} \varphi_j(x)\, dx = c_j$, an m-partition (i.e., a partition into m subregions) is *simultaneously* equitable with respect to φ_1 and φ_2 if $\int_{Q_i} \varphi_j(x)\, dx = c_j/m$ for all $i \in \{1, \ldots, m\}$ and $j \in \{1, 2\}$. Theorem 12 in [32] shows that, given two such functions φ_j, $j \in \{1, 2\}$, there *always* exists an m-partition that is simultaneously equitable with respect to φ_1 and φ_2, and whose subregions Q_i are convex. Then, the following results characterize the optimality of two classes of partitioning policies [15].

Theorem IV.2 (Optimality of partitioning policies)**.** *Assume π^* is a single-vehicle, unbiased optimal policy in the heavy-load regime (i.e., $\varrho \to 1^-$). For m vehicles,*

(i) *a π^*-partitioning policy based on an m-partition which is simultaneously equitable with respect to φ and $\varphi^{1/2}$ is an optimal unbiased policy in heavy-load.*

(ii) *a π^*-partitioning policy based on an m-partition which is equitable with respect to φ does not achieve, in general, the optimal unbiased performance, however it is always within a factor m of it in heavy-load.*

The above results lead to the following strategy: First, for the 1-DTRP, one designs an adaptive and unbiased (in heavy-load) control policy with provable performance guarantees. Then, by using *decentralized* algorithms for environment partitioning, such as those recently developed in [33], one extends such single-vehicle policy to a decentralized and adaptive multi-vehicle policy.

Consider, first, the single vehicle case.

> **The single-vehicle Divide & Conquer (DC) Policy** — Compute an r-partition $\{\mathcal{Q}_k\}_{k=1}^r$ of $\mathcal{Q}$ that is simultaneously equitable with respect to φ and $\varphi^{1/2}$. Let $\tilde{P}_1^*$ be the point minimizing the sum of distances to demands serviced in the past (if no points have been visited in the past, $\tilde{P}_1^*$ is set to be a random point in $\mathcal{Q}$), and let D be the set of outstanding demands waiting for service. If $D = \emptyset$, move to $\tilde{P}_1^*$. If, instead, $D \neq \emptyset$, randomly choose a $k \in \{1, \ldots, r\}$ and move to subregion $\mathcal{Q}_k$; compute the TSP tour through all demands in subregion $\mathcal{Q}_k$ and service all demands in $\mathcal{Q}_k$ by following this TSP tour. If $D \neq \emptyset$ repeat the service process in subregion $k + 1$ (modulo r).

This policy is unbiased in heavy-load. In particular, if $r \to$

$+\infty$, the policy (i) is optimal in light-load and achieves optimal unbiased performance in heavy-load, and (ii) is stable in every load condition. It is possible to show that with $r = 10$ the DC policy is already guaranteed to be within 10% of the optimal (for unbiased policies) performance in heavy-load. If, instead, $r = 1$, the policy (i) is optimal in light-load and within a factor 2 of the optimal unbiased performance in heavy-load, (ii) is stable in every load condition, and (iii) its implementation does *not* require the knowledge of φ. This last property implies that, remarkably, when $r = 1$, the DC policy adapts to *all* problem data (both ϱ and φ). It is worth noting that when $r = 1$ and φ is constant over $\mathcal{Q}$ the DC policy is similar to the generation policy presented in [34].

The optimality of the SQM policy and Theorem IV.2(i) *suggest* the following decentralized and adaptive multi-vehicle version of the DC policy:

(i) compute an m-median of $\mathcal{Q}$ that induces a Voronoi partition that is equitable with respect to φ and $\varphi^{1/2}$,
(ii) assign one vehicle to each Voronoi region,
(iii) each vehicle executes the single-vehicle DC policy in order to service demands that fall within its own subregion, by using the median of the subregion instead of $\tilde{P}_1^*$.

For a given $\mathcal{Q}$ and φ, if there exists an m-median of $\mathcal{Q}$ that induces a Voronoi partition that is equitable with respect to φ and $\varphi^{1/2}$, then the above policy is optimal both in light-load and arbitrarily close to optimality in heavy-load, and stabilizes the system in every load condition. There are two main issues with the above policy, namely (i) existence of an m-median of $\mathcal{Q}$ that induces a Voronoi partition that is equitable with respect to φ and $\varphi^{1/2}$, and (ii) how to compute it. In [33], we showed that for some choices of $\mathcal{Q}$ and φ a median Voronoi diagram that is equitable with respect to φ and $\varphi^{1/2}$ *fails* to exist. Additionally, in [33], we presented a decentralized partitioning algorithm that, for any possible choice of $\mathcal{Q}$ and φ, provides a partition that is equitable with respect to φ and represents a "good" approximation of a median Voronoi diagram (see [33] for details on the metrics that we use to judge "closeness" to median Voronoi diagrams). Moreover, if an m-median of $\mathcal{Q}$ that induces a Voronoi partition that is equitable with respect to φ exists, the algorithm will locally converge to it. This partitioning algorithm is related to the classic Lloyd algorithm from vector quantization theory, and exploits the unique features of power diagrams, a generalization of Voronoi diagrams.

Accordingly, we define the multi-vehicle Divide & Conquer policy as follows.

> **The multi-vehicle Divide & Conquer (m-DC) Policy** — The vehicles run the decentralized partitioning algorithm discussed above (see [33] for more details) and assign themselves to the subregions (this part is indeed a by-product of the algorithm in [33]). Simultaneously, each vehicle executes the single-vehicle DC policy inside its own subregion.

The m-DC policy is within a factor m of the optimal unbiased performance in heavy-load (since the algorithm in [33] *always* provides a partition that is equitable with respect to φ), and stabilizes the system in every load condition. In general, the m-DC policy is only *suboptimal* in light-load; note, however, that the computation of the global minimum of the Weber function H_m (which is non-convex for $m > 1$) is difficult for $m > 1$ (it is NP-hard for the discrete version of the problem); therefore, for $m > 1$, suboptimality has also to be expected from *any* practical implementation of the SQM policy. If an m-median of $\mathcal{Q}$ that induces a Voronoi partition that is equitable with respect to φ exists, the m-DC will locally converge to it, thus we say that the m-DC policy is "locally" optimal in light-load.

Note that, when the density is uniform, a partition that is equitable with respect to φ is also equitable with respect to $\varphi^{1/2}$; therefore, when the density is uniform the m-DC policy is arbitrarily close to optimality in heavy-load (see Theorem IV.2(i)).

The m-DC policy *adapts* to arrival rate λ, expected on-site service $\bar{s}$, and vehicle's velocity v; however, it requires the knowledge of φ.

Tables I and II provide a synoptic view of the results available so far; in particular, our policies are compared with the best unbiased policy available in the literature, i.e., the UTSP policy with $r \to \infty$. In Table I, an asterisk * signals that the result is heuristic. Note that there are currently no results about decentralized and adaptive routing policies that are optimal in light-load and that are optimal *biased* algorithms in heavy-load.

B. A Policy with No Explicit Inter-vehicle Communication

A common theme in cooperative control is the investigation of the effects of different communication and information sharing protocols on the system performance. Clearly, the ability to access more information at each single vehicle cannot decrease the performance level; hence, it is commonly believed that providing better communication among vehicles will improve the system's performance. In [16], we propose a policy for the DVR that does not rely on dedicated communication links between vehicles, but only on the vehicles' knowledge of outstanding demands. An example is when outstanding demands broadcast their location, but vehicles are not aware of one another. We show that, under light load conditions, the inability of vehicles to communicate explicitly does not limit the steady-state performance. In other words, the information contained in the outstanding demands (and hence the effects of others on them) is sufficient to provide, in light load conditions, the same convergence properties attained when vehicles are able to communicate explicitly.

> **The No (Explicit) Communication (NC) Policy** — Let D be the set of outstanding demands waiting for service. If $D = \emptyset$, move to the point minimizing the average distance to demands *serviced in the past* by each vehicle. If there is no unique minimizer, then move to the nearest one. If, instead, $D \neq \emptyset$, move towards the nearest outstanding demand location.

In the NC policy, whenever one or more service requests are outstanding, all vehicles will be pursuing a demand; in particular, when only one service request is outstanding, all

TABLE I
POLICIES FOR THE 1-DTRP

Properties	**DC Policy,** $r \to \infty$	**DC Policy,** $r = 1$	**RH Policy [15]**	**UTSP Policy,** $r \to \infty$
Light-load performance	optimal	optimal	optimal	not optimal
Heavy-load performance	optimal	within 100 % of the optimal	within 100% of the optimal*	optimal
Adaptive to λ, $\bar{s}$, and v	yes	yes	yes	no
Adaptive to φ	no	yes	yes	no

TABLE II
POLICIES FOR THE m-DTRP

Properties	m**-DC Policy,** $r \to \infty$	**UTSP Policy,** $r \to \infty$
Light-load performance	"locally" optimal	not optimal
Heavy-load performance	optimal for uniform φ, within m of optimal unbiased in general	optimal
Adaptive to λ, $\bar{s}$, and v	yes	no
Adaptive to φ	no	no
Distributed	yes	no

vehicles will move towards it. When the demand queue is empty, vehicles will either (i) stop at the current location, if they have visited no demands yet, or (ii) move to their reference point, as determined by the set of demands previously visited.

In [16], we prove that the system time provided by the NC policy converges to a critical point (either a saddle point or a local minimum) of $H_m^*(\mathcal{Q})$ with high probability as $\lambda \to 0^+$. Let us underline that, in general, the achieved critical point strictly depends on the initial positions of the vehicles. We cannot exclude that the algorithm so designed will converge indeed to a saddle point instead of a local minimum. This is due to the fact that the algorithm does not follow the steepest direction of the gradient of the function H_m, but just the gradient with respect to one of the variables. On the other hand, since the algorithm is based on a sequence of demands and at each phase we are trying to minimize a different cost function, it can be proved that the critical points reached by this algorithm are *no worse* than the critical points reached knowing a priori the distribution φ. In [16], we also report results from illustrative numerical experiments that compare the performance of the NC policy with a *sensor-based* policy according to which a demand is considered only by the vehicle whose reference point is closest to the demand location at the time of its generation. We observe that, as λ increases, the performance of the NC policy degrades significantly, almost approaching the performance of a single-vehicle system over an intermediate range of values of λ. Interestingly, this efficiency loss seems to decrease for large λ, and the numerical results suggest that the NC policy recovers a similar performance as the sensor-based policy in the heavy load limit.

Interestingly, the NC policy can be regarded as a learning algorithm in the context of the following game [16]. The service requests are considered as resources and the vehicles as selfish entities. The resources offer rewards in a continuous fashion and the vehicles can collect these rewards by traveling to the resource locations. Every resource offers reward at a unit rate when there is at most one vehicle present at its location and the life of the resource ends as soon as more than one vehicle are present at its location. This setup can be understood to be an extreme form of congestion game, where the resource cannot be shared between vehicles and where the resource expires at the first attempt to share it. The total reward for vehicle i from a particular resource is the time difference between its arrival and the arrival of the next vehicle, if i is the first vehicle to reach the location of the resource, and zero otherwise. The utility function of vehicle i is then defined to be the expected value of reward, where the expectation is taken over the location of the next resource. Hence, the goal of every vehicle is to select their reference location to maximize the expected value of the reward from the next resource. In [16], we prove that the median locations, as a choice for reference positions, are an efficient pure Nash equilibrium for this game. Moreover, we prove that by maximizing their own utility function, the vehicles also maximize the *common global utility function*, which is the negative of the average wait time for service requests.

V. ROUTING FOR ROBOTIC NETWORKS: TIME CONSTRAINTS AND PRIORITIES

In many vehicle routing applications, there are strict service requirements for demands. This can be modeled in two ways. In the first case, demands have (possibly stochastic) deadlines on their waiting times. In the second case, demands have different urgency or "threat" levels, which capture the relative importance of each demand. In this section we study these two related problems and provide routing policies for both scenarios. We discuss hard time constraints in Section V-A and priorities in Section V-B.

In this section we focus only on the case of a uniform spatial density φ. However, the algorithms we present below extend directly to non-uniform density. One simply replaces the equal area partitions with simultaneously equitable (with respect to φ and $\varphi^{1/2}$) partitions, as described for the DC policy in Section IV. The presentation, on the other hand, would become more involved, and thus we restrict our attention to uniform densities.

A. Time Constraints

In [11], [12] we introduced and analyzed DVR with time constraints. Specifically, the setup is the same as that of the m-DTRP, but now each demand j waits for the beginning of its service no longer than a stochastic *patience time* G_j, which is generally distributed according to a distribution function F_G. A vehicle can start the on-site service for the jth demand only within the stochastic time window $[A_j,\, A_j + G_j)$, where A_j is the arrival time of the jth demand. If the on-site service for the jth demand is not started before the time instant $A_j + G_j$, then the jth demand is considered lost; in other words, such demand leaves the system and never returns. If, instead, the on-site service for the jth demand is started before the time instant $A_j + G_j$, then the demand is considered successfully serviced. The waiting time of demand j, denoted again by W_j, is the elapsed time between the arrival of demand j and the time either one of the vehicles starts its service or such demand departs from the system due to impatience, whichever happens first. Hence, the jth demand is considered serviced if and only if $W_j < G_j$. Accordingly, we denote by $\mathbb{P}_\pi\left[W_j < G_j\right]$ the probability that the jth demand is serviced under a routing policy π. The aim is to find the minimum number of vehicles needed to ensure that the steady-state probability that a demand is successfully serviced is larger than a desired value $\phi^{\mathrm{d}} \in (0,\, 1)$, and to determine the policy the vehicles should execute to ensure that such objective is attained.

Formally, define the *success factor* of a policy π as $\phi_\pi := \lim_{j\to+\infty} \mathbb{P}_\pi\left[W_j < G_j\right]$. We identify four types of information on which a control policy can rely: 1) *Arrival time and location*: we assume that the information on arrivals and locations of demands is immediately available to control policies; 2) *On-site service*: the on-site service requirement of demands may either (i) be available, or (ii) be available only through prior statistics, or (iii) not be available to control policies; 3) *Patience time*: the patience time of demands may either (i) be available, or (ii) be available only through prior statistics; 4) *Departure notification*: the information that a demand leaves the system due to impatience may or may not be available to control policies (if the patience time is available, such information is clearly available). Hence, several information structures are relevant. The *least informative* case is when on-site service requirements and departure notifications are not available, and patience times are available only through prior statistics; the *most informative* case is when on-site service requirements and patience times are available.

Given an information structure, we then study the following optimization problem $\mathcal{OPT}$:

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

where $|\pi|$ is the number of vehicles used by π (the existence of the limit $\lim_{j\to\infty} \mathbb{P}_\pi\left[W_j < G_j\right]$ and equivalent formulations in terms of time averages are discussed in [12], [22]). Let m^* denote the optimal cost for the problem $\mathcal{OPT}$ (for a given information structure).

In principle, one should study the problem $\mathcal{OPT}$ for each of the possible information structures. In [12], instead, we considered the following strategy: first, we derived a lower bound that is valid under the most informative information structure (this implies validity under any information structure), then we presented and analyzed two service policies that are amenable to implementation under the least informative information structure (this implies implementability under any information structure). Such approach gives general insights into the problem $\mathcal{OPT}$.

1) Lower Bound: We next present a lower bound for the optimization problem $\mathcal{OPT}$ that holds under any information structure. Let $P = (p_1, \ldots, p_m)$ and define

$$L_m(P, \mathcal{Q}) := 1 - \frac{1}{|\mathcal{Q}|} \int_{\mathcal{Q}} F_G\left(\min_{k\in\{1,\ldots,m\}} \frac{\|x - x_k\|}{v}\right) dx.$$

Theorem V.1 (Lower bound on $\mathcal{OPT}$)**.** *Under any information structure, the optimal cost for the minimization problem $\mathcal{OPT}$ is lower bounded by the optimal cost for the minimization problem*

$$\begin{aligned} \min_{m\in\mathbb{N}_{>0}} \quad & m \\ \textit{subject to} \quad & \sup_{P\in\mathcal{Q}^m} L_m(P, \mathcal{Q}) \geq \phi^{\mathrm{d}}. \end{aligned} \tag{5}$$

The proof of this lower bound relies on some nearest-neighbor arguments. Algorithms to find the solution to the minimization problem in equation (5) have been presented in [12].

2) The Nearest-Depot Assignment (NDA) Policy: We next present the Nearest-Depot Assignment (NDA) policy, which requires the least amount of information and is optimal in light-load.

> **The Nearest-Depot Assignment (NDA) Policy** — Let $\tilde{P}^*_m(\mathcal{Q}) := \arg\max_{P\in\mathcal{Q}^m} L_m(P, \mathcal{Q})$ (if there are multiple maxima, pick one arbitrarily), and let $\tilde{p}^*_k$ be the location of the depot for the kth vehicle, $k \in \{1, \ldots, m\}$. Assign a newly arrived demand to the vehicle whose depot is the nearest to that demand's location, and let D_k be the set of outstanding demands assigned to vehicle k. If the set D_k is empty, move to $\tilde{p}^*_k$; otherwise, visit demands in D_k in first-come, first-served order, by taking the shortest path to each demand location. Repeat.

In [12] we prove that the NDA policy is optimal in light-load under any information structure. Note that the NDA policy is very similar to the SQM policy described in section III-D; the only difference is that the depot locations are now the maximizers of L_m, instead of the minimizers of H_m.

3) The Batch (B) Policy: Finally, we present the Batch (B) policy, which is well-defined for any information structure, however it is particularly tailored for the *least* informative case and is most effective in moderate and heavy-loads.

> **The Batch (B) Policy** — Partition $\mathcal{Q}$ into m equal area regions $\mathcal{Q}_k$, $k \in \{1, \ldots, m\}$, and assign one vehicle to each region. Assign a newly arrived demand that falls in $\mathcal{Q}_k$ to the vehicle responsible for region k, and let D_k be the set of locations of outstanding demands assigned to vehicle k. For each vehicle-region pair k: if the set D_k is empty, move to the

median (the "depot") of $\mathcal{Q}_k$; otherwise, compute a TSP tour through all demands in D_k and vehicle's current position, and service demands by following the TSP tour, skipping demands that are no longer outstanding. Repeat.

Note that this policy is basically a simplified version of the m-DC policy (with $r = 1$).

The following theorem characterizes the batch policy, under the assumption of zero on-site service, and assuming the least informative information structure.

Theorem V.2 (Vehicles required by batch policy)**.** *Assuming zero on-site service, the batch policy guarantees a success factor at least as large as ϕ^{d} if the number of vehicles is equal to or larger than:*

$$\min\Big\{m\Big|\sup_{\theta\in\mathbb{R}_{>0}}(1-F_G(\theta))(1-2g(m)/\theta)\geq\phi^{\mathrm{d}}\Big\},$$

where $g(m) := \frac{1}{2}\left(\frac{\bar{\beta}^2}{v^2}|\mathcal{Q}|\frac{\lambda}{m^2}+\sqrt{\frac{\bar{\beta}^4}{v^4}|\mathcal{Q}|^2\frac{\lambda^2}{m^4}+8\frac{\bar{\beta}^2}{v^2}|\mathcal{Q}|\frac{1}{m}}\right)$*, and where $\bar{\beta}$ is a constant that depends on the* shape *of the service regions.*

Furthermore, in [11] we show that when (i) the system is in heavy-load, (ii) ϕ^{d} tends to one, and (iii) the deadlines are deterministic, the batch policy requires a number of vehicles that is within a factor 3.78 of the optimal.

B. Priorities

In this section we look at a DVR problem in which demands for service have different levels of importance. The service vehicles must then prioritize, providing a quality of service which is proportional to each demand's importance. We introduced this problem in [13]. Formally, we assume an environment $\mathcal{Q}\subset\mathbb{R}^2$, with area $|\mathcal{Q}|$, and m vehicles traveling in $\mathbb{R}^2$, each with maximum speed v. Demands of type $\alpha\in\{1,\dots,n\}$, called α-demands, arrive in the environment according to a Poisson process with rate λ_α. Upon arrival, demands assume an independently and uniformly distributed location in $\mathcal{Q}$. An α-demand requires on-site service with finite mean $\bar{s}_\alpha$.

For this problem the load factor can be written as

$$\varrho := \frac{1}{m}\sum_{\alpha=1}^{n}\lambda_\alpha\bar{s}_\alpha. \tag{6}$$

The condition $\varrho<1$ is necessary for the existence of a stable policy. For a stable policy π, the *average system time per demand* is

$$\overline{T}_\pi = \frac{1}{\Lambda}\sum_{\alpha=1}^{n}\lambda_\alpha\overline{T}_{\pi,\alpha},$$

where $\Lambda := \sum_{\alpha=1}^{n}\lambda_\alpha$, and $\overline{T}_{\pi,\alpha}$ is the expected system time of α-demands (under routing policy π). The average system time per demand is the standard cost functional for queueing systems with multiple classes of demands. Notice that we can write $\overline{T}_\pi = \sum_{\alpha=1}^{n}c_\alpha\overline{T}_{\pi,\alpha}$ with $c_\alpha = \lambda_\alpha/\Lambda$. Thus, if we aim to assign distinct importance levels, we can model priority among classes by allowing any convex combination of $\overline{T}_{\pi,1},\dots,\overline{T}_{\pi,n}$. If $c_\alpha>\lambda_\alpha/\Lambda$, then the system time of α-demands is being weighted more heavily than in the average case. In other words, the quantity $c_\alpha\Lambda/\lambda_\alpha$ gives the priority of α-demands compared to that given in the average system time case. Without loss of generality we can assume that priority classes are labeled so that

$$\frac{c_1}{\lambda_1}\geq\frac{c_2}{\lambda_2}\geq\cdots\geq\frac{c_n}{\lambda_n}, \tag{7}$$

implying that if $\alpha<\beta$ for some $\alpha,\beta\in\{1,\dots,n\}$, then the priority of α-demands is at least as high as that of β-demands.

The problem is as follows. Consider a set of coefficients $c_\alpha>0$, $\alpha\in\{1,\dots,n\}$, with $\sum_{\alpha=1}^{n}c_\alpha=1$, and satisfying expression (7). Determine the policy π (if it exists) which minimizes the cost

$$\overline{T}_{\pi,c} := \sum_{\alpha=1}^{n}c_\alpha\overline{T}_{\pi,\alpha}.$$

In the light-load case where $\varrho\to0^+$ we can use existing policies to solve the problem. This is summarized in the following remark.

Remark V.3 (Light-load regime)**.** *In light-load, it can be verified that the Stochastic Queue Median policy (see Section III-D) provides optimal performance. That is, the vehicles can simply ignore the priorities and service the demands in the FCFS order, returning to their median locations between each service.* □

1) Lower Bound in Heavy-Load: In this section we present a lower bound on the weighted system time $\overline{T}_{\pi,c}$ for every policy π.

Theorem V.4 (Heavy-load lower bound)**.** *The system time of any policy π is lower bounded by*

$$\overline{T}_{\pi,c}\geq\frac{\beta_{\mathrm{TSP},2}^2|\mathcal{Q}|}{2m^2v^2(1-\varrho)^2}\sum_{\alpha=1}^{n}\Big(c_\alpha+2\sum_{j=\alpha+1}^{n}c_j\Big)\lambda_\alpha, \tag{8}$$

as $\varrho\to1^-$, where $c_1,\dots,c_n$ satisfy expression (7)*.*

Remark V.5 (Lower bound for all $\varrho\in[0,1)$)**.** *Lower bound* (8) *holds only in heavy-load. We can also obtain a lower bound that is valid for all values of ϱ. However, in the heavy-load limit it is less tight than bound* (8)*. Under the labeling in expression* (7)*, this general bound for any policy π is*

$$\overline{T}_{\pi,c}\geq\frac{\gamma^2|\mathcal{Q}|}{m^2v^2(1-\varrho)^2}\sum_{\alpha=1}^{n}\Big(\Big(c_\alpha+2\sum_{j=\alpha+1}^{n}c_j\Big)\lambda_\alpha\Big) - \frac{mc_1}{2\lambda_1}+\sum_{\alpha=1}^{n}c_\alpha\bar{s}_\alpha, \tag{9}$$

where $\varrho\in[0,1)$ and $\gamma=2/(3\sqrt{2\pi})\approx0.266$. □

2) The Separate Queues Policy: In this section we present the Separate Queues (SQ) policy. This policy utilizes a probability distribution $\mathbf{p}=[p_1,\dots,p_n]$, where $p_\alpha>0$ for each $\alpha\in\{1,\dots,n\}$, defined over the priority classes. The distribution $\mathbf{p}$ is a set of parameters to be used to optimize performance.

Separate Queues (SQ) Policy — Partition $\mathcal{Q}$ into m equal area regions and assign one vehicle to

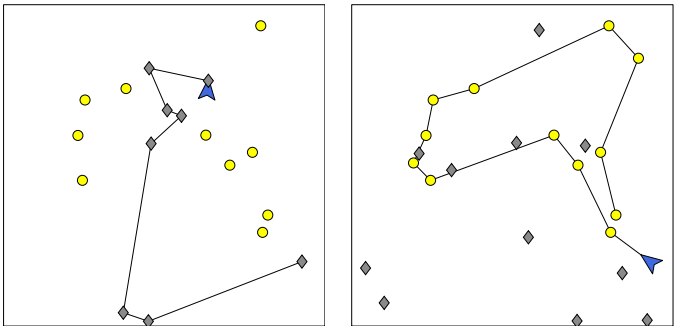

Fig. 2. A representative simulation of the SQ policy for one vehicle and two priority classes. Circle shaped demands are high priority, and diamond shaped are low priority. The vehicle is marked by a chevron shaped object and TSP tour is shown in a solid line. The left-figure shows the vehicle computing a tour through class 2 demands. The right-figure shows the vehicle after completing the class 2 tour and computing a new tour through all outstanding class 1 demands.

each region. For each vehicle, if region contains no demands, then move to median location of region until a demand arrives. Otherwise, select a class according to the distribution $\mathbf{p}$. Compute a TSP tour through all demands in region of the selected class and service all of these demands by following the TSP tour. When tour is completed, repeat by selecting a new class.

Figure 2 shows an illustrative example of the SQ policy. In the first frame the vehicle is servicing only class 2 (diamond shaped) demands, whereas in the second frame, the vehicle is servicing class 1 (circle shaped) demands.

3) Performance of the SQ Policy: By upper bounding the expected steady-state number of demands in each class, we are able to obtain the following expression for the system time of the SQ policy in heavy-load:

$$\overline{T}_{\mathrm{SQ},c} \leq \frac{\beta_{\mathrm{TSP},2}^2 |\mathcal{Q}|}{m^2 v^2 (1-\varrho)^2} \sum_{\alpha=1}^{n} \frac{c_\alpha}{p_\alpha} \left(\sum_{i=1}^{n} \sqrt{\lambda_i p_i} \right)^2 . \qquad (10)$$

Thus, we can minimize this upper bound by appropriately selecting the probability distribution $\mathbf{p} = [p_1, \ldots, p_n]$. With the selection

$$p_\alpha := c_\alpha \quad \text{for each } \alpha \in \{1, \ldots, n\},$$

we obtain the following result.

Theorem V.6 (SQ policy performance)**.** *As $\varrho \to 1^-$, the system time of the SQ policy is within a factor $2n^2$ of the optimal system time. This factor is independent of the arrival rates $\lambda_1, \ldots, \lambda_n$, coefficients $c_1, \ldots, c_n$, service times $\bar{s}_1, \ldots, \bar{s}_n$, and the number of vehicles m.*

In [13], numerical experiments are used to verify the tightness of the upper bound (10), and to compare methods for optimization of the distribution $\mathbf{p}$.

4) Heuristic Improvements: We now present two heuristic improvements on the SQ policy. The first improvement, called the *queue merging heuristic*, is guaranteed to never increase the upper bound on the expected system time, and in certain instances it significantly decreases the upper bound. To motivate the modification, consider the case when all classes have equal priority (i.e., $c_1/\lambda_1 = \cdots = c_n/\lambda_n$), and we use the probability assignment $p_\alpha = c_\alpha$ for each class α. Then, the upper bound for the Separate Queues policy is n times larger than if we (i) ignore priorities, (ii) merge the n classes into a single class, and (iii) run the SQ policy on the merged class (i.e., at each iteration, service all outstanding demands in $\mathcal{Q}$ via the TSP tour).

Motivated by this discussion, we define a *merge configuration* to be a partition of n classes $\{1, \ldots, n\}$ into ℓ sets $C_1, \ldots, C_\ell$, where $\ell \in \{1, \ldots, n\}$. The idea is to run the Separate Queues policy on the ℓ classes, where class $i \in \{1, \ldots, \ell\}$ has arrival rate $\sum_{\alpha \in C_i} \lambda_\alpha$ and convex combination coefficient $\sum_{\alpha \in C_i} c_\alpha$. Given a merge configuration $\{C_1, \ldots, C_\ell\}$, and using the probability assignment $p_i = \sum_{\alpha \in C_i} c_\alpha$ for each class $i \in \{1, \ldots, \ell\}$, the analysis leading to (10) can easily be modified to yield an upper bound of

$$\frac{\beta_{\mathrm{TSP},2}^2 |\mathcal{Q}| \ell}{m^2 v^2 (1-\varrho)^2} \left(\sum_{i=1}^{\ell} \sqrt{\sum_{\alpha \in C_i} c_\alpha \sum_{\beta \in C_i} \lambda_\beta} \right)^2 . \qquad (11)$$

The SQ-policy with merging can be summarized as follows:

> **Separate Queues (SQ) with Merging Policy** — Find the merge configuration $\{C_1, \ldots, C_\ell\}$ which minimizes equation (11). Run the Separate Queues policy on ℓ classes, where class i has arrival rate $\sum_{\alpha \in C_i} \lambda_\alpha$ and convex combination coefficient $\sum_{\alpha \in C_i} c_\alpha$.

Now, to minimize equation (11) in the SQ with Merging policy, one must search over all possible partitions of a set of n elements. The number of partitions is given by the Bell Number and thus search becomes infeasible for more than approximately 10 classes. However, one can also limit the search space in order to increase the number of classes that can be considered as in [13].

The second heuristic improvement for the SQ policy which can be used in implementation is called the *tube heuristic*. The heuristic improvement is as follows:

> **The Tube Heuristic** — When following a tour, service all newly arrived demands that lie within distance $\epsilon > 0$ of the tour.

The idea behind the heuristic is to utilize the fact that some newly arrived demands will be "close" to the demands in the current service batch, and thus can be serviced with minimal additional travel cost. Analysis of the tube heuristic is complicated by the fact that it introduces correlation between demand locations.

The parameter ϵ should be chosen such that the total tour length is not increased by more than, say, 10%. A rough calculation shows that ϵ should scale as

$$\epsilon \sim \sqrt{\frac{\mu |\mathcal{Q}|}{\text{total expected number of demands}}},$$

where μ is the fractional increase in tour length (e.g., 10%). Numerical simulations presented in [13] show that this heuristic, with an appropriately chosen value of ϵ, improves the SQ performance by a factor of approximately 2. In a more sophisticated implementation we define an ϵ_α for each $\alpha \in$

$\{1, \ldots, n\}$, where the magnitude of ϵ_α is proportional to the probability p_α.

VI. Routing for Robotic Networks: Constraints on Vehicle Motion

In this section, we consider the m-DTRP described in the earlier sections with the addition of differential constraints on the vehicle's motion [18]. In particular, we concentrate on vehicles that are constrained to move on the plane at constant speed $v > 0$ along paths with a minimum radius of curvature $\rho > 0$. Such vehicles, often referred to as Dubins vehicles, have been extensively studied in the robotics and control literature [35], [36], [37]. Moreover, the Dubins vehicle model is widely accepted as a reasonably accurate model to represent aircraft kinematics, e.g., for air traffic control [38], [39], and UAV mission planning purposes [40], [4], [41]. Accordingly, the DVR problem studied in this section will be referred to as the *m-Dubins DTRP*. In this section we focus only on the case of a uniform spatial density φ.

A feasible path for the Dubins vehicle (called *Dubins path*) is defined as a path that is twice differentiable almost everywhere, and such that its radius of curvature is bounded below by ρ. Since a Dubins vehicle cannot stop, we only consider zero on-site service time. Hence, the generic load factor $\varrho = \lambda \bar{s}/m$, as defined in subsection III-B, becomes inappropriate for this setup and, in accordance with Remark III.2, the heavy-load regime is defined as $\lambda/m \to +\infty$. We correspondingly define the light-load regime as $\lambda/m \to 0^+$.

A. Lower Bounds

In this section we provide lower bounds on the system time for the m-Dubins DTRP.

Theorem VI.1 (System time lower bounds)**.** *The optimal system time for the m-Dubins DTRP satisfies the following lower bounds:*

$$\overline{T}^* \geq \frac{H_m^*(\mathcal{Q})}{v}, \tag{12}$$

$$\liminf_{d_\rho \to +\infty} \overline{T}^* \left(\frac{m}{\rho|\mathcal{Q}|}\right)^{1/3} \geq \frac{3\sqrt[3]{3}}{4v}, \tag{13}$$

$$\liminf_{\frac{\lambda}{m} \to +\infty} \overline{T}^* \frac{m^3}{\lambda^2} \geq \frac{81}{64} \frac{\rho|\mathcal{Q}|}{v^3}, \tag{14}$$

where $|\mathcal{Q}|$ is the area of $\mathcal{Q}$ and $d_\rho := \frac{\rho^2 m}{|\mathcal{Q}|}$ is the nonholonomic vehicle density.

The lower bound (12) follows from equation (1); however this bound is obtained by approximating the Dubins distance (i.e., the length of the shortest feasible path for a Dubins vehicle) with the Euclidean distance. The lower bound (13) is obtained by explicitly taking into account the Dubins turning cost. Although the first two lower bounds of Theorem VI.1 are valid for any λ, they are particularly useful in the light-load regime. The lower bound (14) is valid and useful in the heavy-load regime.

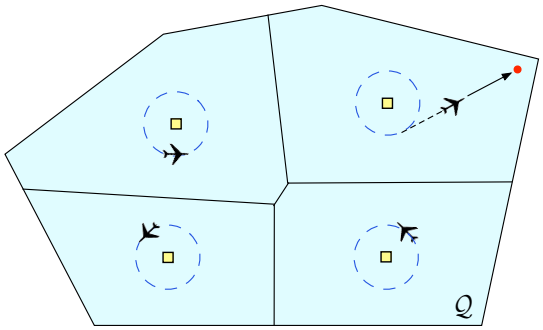

Fig. 3. Illustration of the Median Circling policy. The squares represent $P_m^*(\mathcal{Q})$, the m-median of $\mathcal{Q}$. Each vehicle loiters about its respective generator at a radius ρ. The regions of dominance are the Voronoi partition generated by $P_m^*(\mathcal{Q})$. In this figure, a demand has appeared in the subregion roughly in the upper-right quarter of the domain. The vehicle responsible for this subregion has left its loitering orbit and is en route to service the demand.

B. Routing Policies for the m-Dubins DTRP

We start by considering two policies that are particularly efficient in light load. The first light-load policy, called the Median Circling policy, imitates the optimal policy for Euclidean vehicles, assigning static regions of responsibility. As usual, let $P_m^*(\mathcal{Q})$ be the m-median of $\mathcal{Q}$. The policy is formally described as follows.

> **The Median Circling (MC) Policy** — Let the *loitering orbits* for the vehicles be circular trajectories of radius ρ centered at entries of $P_m^*(\mathcal{Q})$, with each vehicle allotted one trajectory. Each vehicle visits the demands in the Voronoi region $V_i(P_m^*(\mathcal{Q}))$ in the order in which they arrive. When no demands are available, the vehicle returns to its loitering orbit; the direction in which the orbit is followed is not important, and can be chosen in such a way that the orbit is reached in minimum time.

An illustration of the MC policy is shown in Figure 3.

We next introduce a second light-load policy, namely the Strip Loitering policy, which is more efficient than the MC policy when the nonholonomic vehicle density is large and relies on dynamic regions of responsibility for the vehicles. An illustration of the Strip Loitering policy is shown in Figure 4.

> **The Strip Loitering (SL) Policy** — Bound the environment $\mathcal{Q}$ with a rectangle of minimum *height*, where height denotes the smaller of the two side lengths of a rectangle. Let R and S be the *width* and *height* of this bounding rectangle, respectively. Divide $\mathcal{Q}$ into strips of width r, where
>
> $$r = \min\left\{\left(\frac{4}{3\sqrt{\rho}} \frac{RS + 10.38\rho S}{m}\right)^{2/3}, \; 2\rho\right\}.$$
>
> Orient the strips along the side of length R. Construct a closed Dubins path, henceforth referred to as the *loitering path*, which runs along the longitudinal bisector of each strip, visiting all strips in top-to-bottom sequence, making U-turns between strips at the edges of $\mathcal{Q}$, and finally returning to the initial configuration. The m vehicles are allotted loitering

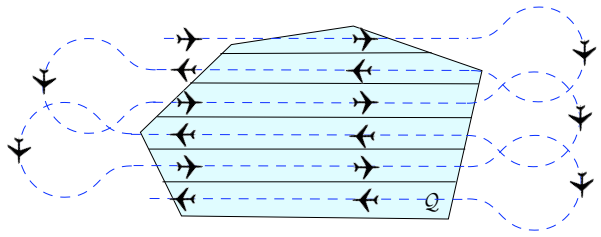

Fig. 4. Illustration of the Strip Loitering policy. The trajectory providing closure of the loitering path (along which the vehicles travel from the end of the last strip to the beginning of the first strip) is not shown here for clarity of the drawing.

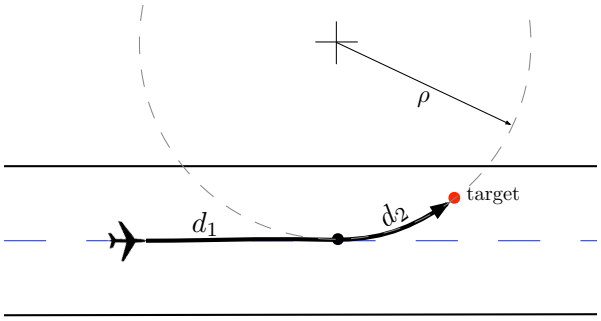

Fig. 5. Close-up of the Strip Loitering policy with construction of the point of departure and the distances d_1, and d_2 for a given demand, at the instant of appearance.

positions on this path, equally spaced, in terms of path length.

When a demand arrives, it is allocated to the closest vehicle among those that lie within the same strip as the demand and that have the demand in front of them. When a vehicle has no outstanding demands, the vehicle returns to its loitering position as follows. (We restrict the exposition to the case when a vehicle has only one outstanding demand when it leaves its loitering position and no more demands are allotted to it before it returns to its loitering position; other cases can be handled similarly.) After making a left turn of length d_2 (as shown in Figure 5) to service the demand, the vehicle makes a right turn of length $2d_2$ followed by another left turn of length d_2, and then returns to the loitering path. However, the vehicle has fallen behind in the loitering path by a distance $4(d_2 - \rho \sin \frac{d_2}{\rho})$. To rectify this, as it nears the end of the current strip, it takes its U-turn a distance $2(d_2 - \rho \sin \frac{d_2}{\rho})$ early.

Note that the loitering path must cover $\mathcal{Q}$, but it need not cover the entire bounding box of $\mathcal{Q}$. The bounding box is merely a construction used to place an upper bound on the total path length.

The MC and SL policies will be proven to be efficient in light-load. We now propose the Bead Tiling policy which will be proven to be efficient in heavy-load. A key component of the algorithm is the construction of a novel geometric set, tuned to the kinetic constraints of the Dubins vehicle, called the *bead* [17]. The construction of a bead $\mathcal{B}_\rho(\ell)$ for a given ρ and an additional parameter $\ell > 0$ is illustrated in Figure 6.

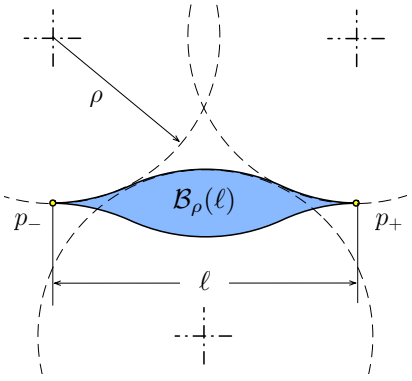

Fig. 6. An illustration for the construction of the bead for a given ρ and ℓ.

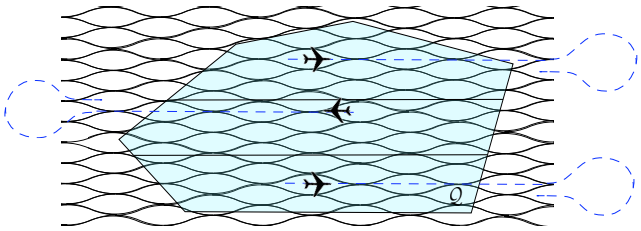

Fig. 7. An illustration of the mBT policy.

We start with the policy for a single vehicle.

The Bead Tiling (BT) Policy — Bound the environment $\mathcal{Q}$ with a rectangle of minimum *height*, where height denotes the smaller of the two side lengths of a rectangle. Let R and S be the *width* and *height* of this bounding rectangle, respectively. Tile the plane with identical beads $\mathcal{B}_\rho(\ell)$ with $\ell = \min\{C_{\mathrm{BTA}} v/\lambda, 4\rho\}$, where

$$C_{\mathrm{BTA}} = \frac{7-\sqrt{17}}{4}\left(1 + \frac{7\pi\rho S}{3|\mathcal{Q}|}\right)^{-1}.$$

The beads are oriented to be along the width of the bounding rectangle. The Dubins vehicle visits all beads intersecting $\mathcal{Q}$ in a row-by-row fashion in top-to-bottom sequence, servicing at least one demand in every nonempty bead. This process is repeated indefinitely.

The BT policy is extended to the m-vehicle Bead Tiling (mBT) policy in the following way (see Figure 7).

The m-vehicle Bead Tiling (mBT) Policy — Divide the environment into regions of dominance with lines parallel to the bead rows. Let the area and height of the i-th vehicle's region be denoted with $|\mathcal{Q}|_i$ and S_i. Place the subregion dividers in such a way that

$$|\mathcal{Q}|_i + \frac{7}{3}\pi\rho S_i = \frac{1}{m}\left(|\mathcal{Q}| + \frac{7}{3}\pi\rho S\right)$$

for all $i \in \{1, \ldots, m\}$. Allocate one subregion to every vehicle and let each vehicle execute the BT policy in its own region.

C. Analysis of Routing Policies

We now present the performance analysis of the routing policies we introduced in the previous section.

Theorem VI.2 (MC policy performance in light-load)**.** *The MC policy is a stabilizing policy in light-load, i.e., as $\lambda/m \to 0^+$. The system time of the Median Circling policy in light-load satisfies, as $\lambda/m \to 0^+$,*

$$\limsup_{\frac{\lambda}{m}\to 0^+} \frac{\overline{T}_{\mathrm{MC}}}{\overline{T}^*} \leq 1 + 25\sqrt{d_\rho},$$

and, in particular,

$$\lim_{d_\rho \to 0^+} \limsup_{\frac{\lambda}{m}\to 0^+} \frac{\overline{T}_{\mathrm{MC}}}{\overline{T}^*} = 1.$$

Theorem VI.2 implies that the MC policy is optimal in the asymptotic regime where $\lambda/m \to 0^+$ *and* $d_\rho \to 0^+$. Hence, the MC policy is particularly efficient in light-load for low values of the nonholonomic vehicle density. Moreover, Theorem VI.2 together with Theorem VI.1 and Equation (21) (provided in the Appendix) implies that the optimal system time in the aforementioned asymptotic regime belongs to $\Theta\left(1/(v\sqrt{m})\right)$.

We now characterize the performance of the SL policy.

Theorem VI.3 (SL policy performance in light-load)**.** *The SL policy is a stabilizing policy in light-load, i.e., when $\lambda/m \to 0^+$. Moreover, the system time of the SL policy satisfies, as $\lambda/m \to 0^+$,*

$$\overline{T}_{\mathrm{SL}} \leq \begin{cases} \frac{1.238}{v}\left(\frac{\rho RS + 10.38\rho^2 S}{m}\right)^{1/3} + \frac{R+S+6.19\rho}{mv} \\ \qquad\qquad \textit{for } m \geq 0.471\left(\frac{RS}{\rho^2} + \frac{10.38S}{\rho}\right), \\ \frac{RS+10.38\rho S}{4\rho m v} + \frac{R+S+6.19\rho}{mv} + \frac{1.06\rho}{v} \quad \textit{otherwise}. \end{cases}$$

Theorem VI.3 together with Theorem VI.1 implies that the SL policy is within a constant factor of the optimal in the asymptotic regime where $\lambda/m \to 0^+$ *and* $d_\rho \to +\infty$. Moreover, in such asymptotic regime the optimal system time belongs to $\Theta\left(1/(v\sqrt[3]{m})\right)$.

Finally, we characterize the performance of the mBT policy.

Theorem VI.4 (mBT policy performance in heavy-load)**.** *The mBT policy is a stabilizing policy. Moreover, the system time for the mBT policy satisfies the following*

$$\lim_{\frac{\lambda}{m}\to +\infty} \overline{T}_{\mathrm{mBT}} \frac{m^3}{\lambda^2} \leq 71 \frac{\rho|\mathcal{Q}|}{v^3}\left(1 + \frac{7\pi\rho S}{3|\mathcal{Q}|}\right)^3.$$

Theorem VI.3 together with Theorem VI.1 implies that the mBT policy is within a constant factor of the optimal in heavy-load, and that the optimal system time in this case belongs to $\Theta\left(\lambda^2/(mv)^3\right)$.

It is instructive to compare the scaling of the optimal system time with respect to λ, m and v for the m-DTRP and for the m-Dubins DTRP. Such comparison is shown in Table III. One can observe that in heavy-load the optimal system time for the m-Dubins DTRP is of the order $\lambda^2/(mv)^3$, whereas for the m-DTRP it is of the order $\lambda/(mv)^2$. Therefore, our analysis rigorously establishes the following intuitive fact: bounded-curvature constraints make the optimal system much more sensitive to increases in the demand generation rate. Perhaps less intuitive is the fact that the optimal system time is also more sensitive with respect to the number of vehicles and the vehicle speed in the m-Dubins DTRP as compared to the m-DTRP. Extension of the results for the Dubins DTRP in the light-load case for dimensions higher than 2 is still an open problem. We have extended the results for the Dubins DTRP in the heavy-load case to the three-dimensional case [43]. However, the extension to dimensions higher than 3 is still an open problem.

TABLE III
A COMPARISON BETWEEN THE SCALING OF THE OPTIMAL SYSTEM TIME FOR THE m-DTRP AND FOR THE m-DUBINS DTRP.

	m-DTRP	m-Dubins DTRP
$\overline{T}^*$ $(\lambda/m \to +\infty)$	$\Theta\left(\lambda/(mv)^2\right)$ [8]	$\Theta\left(\lambda^2/(mv)^3\right)$ [18]
$\overline{T}^*$ $(\lambda/m \to 0^+)$	$\Theta\left(1/(v\sqrt{m})\right)$ [42]	$\Theta\left(1/(v\sqrt{m})\right)$ $(d_\rho \to 0^+)$ $\Theta\left(1/(v\sqrt[3]{m})\right)$ $(d_\rho \to +\infty)$ [18]

In [18], we report results from illustrative numerical experiments on the performance of MC, SL and mBT policies. A close observation of the system time in the light-load case shows that the territorial MC policy is optimal as $d_\rho \to 0^+$ and the gregarious SL policy is constant-factor optimal as $d_\rho \to +\infty$. This suggests the existence of a phase transition in the optimal policy for the light-load scenario as one increases the number of vehicles for a fixed ρ and $\mathcal{Q}$ (recall that $d_\rho = \rho^2 m/|\mathcal{Q}|$). It is desirable to study the fundamental factors driving this transition, ignoring its dependence on the shape of $\mathcal{Q}$. Towards this end, envision an infinite number of vehicles operating on the unbounded plane. In this case, the configuration $P_m^*(\mathcal{Q})$ yielding the minimum of the function H_m is that in which the Voronoi partition induced by $P_m^*(\mathcal{Q})$ is a network of regular hexagons [42]. Moreover, in this scenario, the SL policy reduces to vehicles moving straight on infinite strips. In this setup, it is observed that the phase transition can be characterized by a critical value of the dimensionless parameter of the nonholonomic density [18], estimated to be $d_\rho^{\mathrm{unbd}} \approx 0.0587$. An alternate interpretation is that the transition occurs when each vehicle is responsible for a region of area 5.42 times that of a minimum turning-radius disk. This critical value of d_ρ obtained for unbounded domain has been found to be very close to the values obtained, via numerical experiments, for bounded domains [18]. This result provides a system architect with valuable information to decide upon the optimal strategy in the light-load scenario for given problem parameters. Similar phase transition phenomena for other vehicles have been studied in [44].

VII. Routing for Robotic Networks: Team Forming for Cooperative Tasks

Here we study demands (or tasks) that require the simultaneous services of several vehicles [20]. In particular,

consider m vehicles, each capable of providing one of k services. We assume that there are $m_j > 0$ vehicles capable of providing service j (called vehicles of service-type j), for each $j \in \{1, \ldots, k\}$, and thus $m := \sum_{j=1}^{k} m_j$.

In addition, we assume there are $\mathcal{K}$ different types of tasks. Tasks of type $\alpha \in \{1, \ldots, \mathcal{K}\}$ arrive according to a Poisson process with rate λ_α, and assume a location i.i.d. uniformly in $\mathcal{Q}$.[2] The total arrival rate is $\lambda := \sum_{\alpha=1}^{\mathcal{K}} \lambda_\alpha$. Each task-type α requires a subset of the k services. We record the required services in a zero-one (column) vector $R_\alpha \in \{0,1\}^k$. The jth entry of R_α is 1 if service j is required for task-type α, and 0 otherwise. The on-site service time for each task-type α has mean $\bar{s}_\alpha$. To complete a task of type α, a team of vehicles capable of providing the required services must travel to the task location and remain there simultaneously for the on-site service time. We refer to this problem as the *dynamic team forming problem* [20].

As a motivating example, consider the scenario given in Section I where each demand (or task) corresponds to an event that requires close-range observation. The sensors required to properly assess each event will depend on that event's properties. In particular, a event may require several different sensing modalities, such as electro-optical, infra-red, synthetic aperture radar, foliage penetrating radar, and moving target indication radar [45]. One solution would be to equip each UAV with all sensing modalities (or services). However, in many cases, most events will require only a few sensing modalities. Thus, we might increase our efficiency by having a larger number of UAVs, each equipped with a single modality, and then forming the appropriate sensing team to observe each event.

We restrict our attention to *task-type unbiased* policies; policies π for which the system time of each task (denoted by $\overline{T}_{\pi,\alpha}$) is equal, and thus $\overline{T}_{\pi,1} = \overline{T}_{\pi,2} = \cdots = \overline{T}_{\pi,\mathcal{K}} =: \overline{T}_\pi$. We seek policies π which minimize the expected system time of tasks $\overline{T}_\pi$. Policies of this type are amenable to analysis because the task-type unbiased constraint collapses the feasible set of system times from a subset of $\mathbb{R}^{\mathcal{K}}$ to a subset of $\mathbb{R}$. Defining the matrix

$$R := [R_1 \ \cdots \ R_{\mathcal{K}}] \in \{0,1\}^{k \times \mathcal{K}}, \tag{15}$$

a necessary condition for stability is

$$R[\lambda_1 \bar{s}_1 \ \cdots \ \lambda_{\mathcal{K}} \bar{s}_{\mathcal{K}}]^T < [m_1 \ \cdots \ m_k]^T \tag{16}$$

component-wise. Note that this condition is akin to the "load factor" in Subsection III-B. However, the space of load factors is much richer, and thus light and heavy-load are no longer simply defined. To simplify the problem we take an alternative approach. We study the performance as the number of vehicles becomes very large, i.e., $m \to +\infty$. In addition, we simply look at the order of the expected delay, without concern for the constant factors. It turns out that this analysis provides substantial insight into the performance of different team forming policies. This type of asymptotic analysis is frequently performed in computational complexity [46] and ad-hoc networking [47].

[2] As in Section V, the algorithms in this section extend directly to a non-uniform spatial density by utilizing simultaneously equitably partitions.

A. Three Team Forming Policies

We now present three team forming policies.

The Complete Team (CT) Policy — Form m/k teams of k vehicles, where each team contains one vehicle of each service-type. Each team meets and moves as a single entity. As tasks arrive, service them by one of the m/k teams according to the UTSP policy.

For the second policy, recall that the vector $R\mathbf{1}_{\mathcal{K}}$ records in its jth entry the number of task-types that require service j, where $\mathbf{1}_{\mathcal{K}}$ is a $\mathcal{K} \times 1$ vector of ones. Thus, if

$$R\mathbf{1}_{\mathcal{K}} \leq [m_1, \ldots, m_k]^T \tag{17}$$

component-wise, then there are enough vehicles of each service-type to create $\lfloor m_{\text{TST}} \rfloor$ teams, where

$$m_{\text{TST}} := \min \left\{ \frac{m_j}{e_j^T R \mathbf{1}_{\mathcal{K}}} \,\middle|\, j \in \{1, \ldots, k\} \right\}$$

dedicated teams for each task-type, where e_j is the jth vector of the standard basis of $\mathbb{R}^k$. Thus, when equation (17) is satisfied, we have the following policy.

The Task-Specific Team (TT) Policy — For each of the $\mathcal{K}$ task-types, create $\lfloor m_{\text{TST}} \rfloor$ teams of vehicles, where there is one vehicle in the team for each service required by the task-type. Service each task by one of its $\lfloor m_{\text{TST}} \rfloor$ corresponding teams, according to the UTSP policy.

The task-specific team policy can be applied only when there is a sufficient number of vehicles of each service-type. The following policy requires only a single vehicle of each service type. The policy partitions the task-types into groups, where each group is chosen such that there is a sufficient number of vehicles to create a dedicated team for each task-type in the group. The task-specific team policy is then run on each group sequentially. The groups are defined via a *service schedule* which is a partition of the $\mathcal{K}$ task-types into $L \leq \mathcal{K}$ time slots, such that each task-type appears in precisely one time slot, and the task-types in each time slot are pairwise disjoint (i.e., in a given time slot, each service appears in at most one task-type).[3] We now formally present the third policy.

The Scheduled Task-Specific Team (STT) Policy — Partition $\mathcal{Q}$ into $m_{\text{CT}} := \min\{m_1, \ldots, m_k\}$ equal area regions and assign one vehicle of each service-type to each region. In each region form a queue for each of the $\mathcal{K}$ task-types. For each time slot in the schedule and each task-type in the time slot, create a team containing one vehicle for each required service. For each team, service the first n tasks (n is a design parameter) in the corresponding queue by following an optimal TSP tour. When the end of the service schedule is reached, repeat. Optimize over n (see [20] for details).

[3] Computing an optimal schedule is equivalent to solving a vertex coloring problem, which is NP-hard. However, an approximate schedule can be computed via known vertex coloring heuristics; see [20] for more details.

B. Performance of Policies

To analyze the performance of the policies we make the following simplifying assumptions: (A1) There are n/k vehicles of each service-type. (A2) The arrival rate is $\lambda/\mathcal{K}$ for each task-type. (A3) The on-site service time has mean $\bar{s}$ and is upper bounded by $s_{\max}$ for all task-types. (A4) There exists $p \in [1/k, 1]$ such that for each $j \in \{1, \ldots, k\}$ the service j appears in $p\mathcal{K}$ of the $\mathcal{K}$ task-types. Thus, each task will require service j with probability p.

With these assumptions, the stability condition in equation (16) simplifies to

$$\frac{\lambda}{m} < \frac{1}{pk\bar{s}}. \tag{18}$$

We say that λ is the *total throughput* of the system (i.e., the total number of tasks served per unit time), and $B_m := \lambda/m$ is the *per-vehicle throughput*.

Finally, we study the system time as the number of vehicles m becomes large. As m increases, if the density of vehicles is to remain constant, then the environment must grow. In fact, the ratio $\sqrt{|\mathcal{Q}|}/v$ must scale as $\sqrt{m}$, [48]. In [2] this scaling is referred to as a *critical environment*. Thus we will study the performance in the asymptotic regime where (i) the number of vehicles $m \to +\infty$; (ii) on-site service times are independent of m; (iii) $|\mathcal{Q}(m)|/(mv^2(m)) \to \text{constant} > 0$.

To characterize the system time as a function of the per-vehicle throughput B_m we introduce the canonical throughput vs. system time profile $f_{\overline{T}_{\min}, \overline{T}_{\text{ord}}, B_{\text{crit}}} : \mathbb{R}_{>0} \to \mathbb{R}_{>0} \cup \{+\infty\}$ which has the form

$$B_m \mapsto \begin{cases} \max\left\{\overline{T}_{\min}, \dfrac{\overline{T}_{\text{ord}}(B_m/B_{\text{crit}})}{(1 - B_m/B_{\text{crit}})^2}\right\}, & \text{if } B_m < B_{\text{crit}}, \\ +\infty, & \text{if } B_m \geq B_{\text{crit}}. \end{cases} \tag{19}$$

This profile (see Figure 8) is described by the three positive parameters $\overline{T}_{\min}$, $\overline{T}_{\text{ord}}$ and B_{crit}, where $\overline{T}_{\text{ord}} \geq \overline{T}_{\min}$. These parameters have the following interpretation:

- $\overline{T}_{\min}$ is the minimum achievable system time for any positive throughput.
- B_{crit} is the maximum achievable throughput (or capacity).
- $\overline{T}_{\text{ord}}$ is the system time when operating at $(3 - \sqrt{5})/2 \approx 38\%$ of capacity B_{crit}. Additionally, $\overline{T}_{\text{ord}}$ captures the order of the system time when operating at a constant fraction of capacity.

For each of the three policies π, we can write the system time as $\overline{T}_\pi \in O\big(f_{\overline{T}_{\min}, \overline{T}_{\text{ord}}, B_{\text{crit}}}(B_m)\big)$ for some values of $\overline{T}_{\min}$, $\overline{T}_{\text{ord}}$, and B_{crit}. In addition, we can write the lower bound in the form $\overline{T}^* \in \Omega\big(f_{\overline{T}_{\min}, \overline{T}_{\text{ord}}, B_{\text{crit}}}(B_m)\big)$. We summarize the corresponding parameter values for the policies and the lower bound in Table IV. We refer the reader to [20] for the proof of each upper and lower bound. In this table, k is the number of services, $\mathcal{K}$ is number of task-types, L is the length of the service schedule, $C := m_{\text{TST}}/\lfloor m_{\text{TST}} \rfloor$, and p is the probability that a task-type requires service j for each $j \in \{1, \ldots, k\}$.

From these results we draw several conclusions. First, if the throughput is very low, then the CT Policy has an expected system time of $\Theta(\sqrt{k})$, which is within a constant factor of the optimal. In addition, if p is close to one and each task requires nearly every service, then CT is within a constant factor of the optimal in terms of capacity and system time. Second, if $p \sim 1/k$ and each task requires few services, then the capacity of CT is sub-optimal, and the capacity of both TT and STT are within a constant factor of optimal. However, the system time of the TT and STT policies may be much higher than the lower bound when the number of task-types $\mathcal{K}$ is very large. Third, the TT policy performs at least as well as the STT policy, both in terms of capacity and system time. Thus, one should use the TT policy if there is a sufficient number of vehicles of each service-type. However, if $p \sim 1/k$ and if resources are limited such that the TT policy cannot be used, then the STT Policy should be used to maximize capacity.

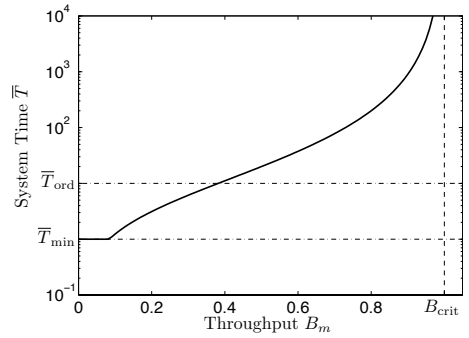

Fig. 8. The canonical throughput vs. system time profile for the dynamic team forming problem. The semi-log plot is for parameter values of $\overline{T}_{\min} = 1$, $\overline{T}_{\text{ord}} = 10$, and $B_{\text{crit}} = 1$. If $B_m \geq B_{\text{crit}}$, then the system time is $+\infty$.

TABLE IV
A COMPARISON OF THE CANONICAL THROUGHPUT VS. SYSTEM TIME PARAMETERS FOR THE THREE POLICIES. HERE $p\mathcal{K} \leq L \leq \mathcal{K}$ IS THE SCHEDULE LENGTH.

	$\overline{T}_{\min}$	$\overline{T}_{\text{ord}}$	B_{crit}
Lower bound $(\overline{T}^*)$	$\sqrt{k}$	k	$\frac{1}{pk\bar{s}}$
CT Policy	$\sqrt{k}$	k	$\frac{1}{k\bar{s}}$
TT Policy	$\sqrt{pk\mathcal{K}}$	$pk\mathcal{K}$	$\frac{1}{C\bar{s}pk}$
STT Policy	$L\sqrt{k}$	Lk	$\frac{\mathcal{K}}{s_{\max}Lk}$

VIII. Summary and Future Directions

In this paper we presented a joint algorithmic and queueing approach to the design of cooperative control, task allocation and dynamic routing strategies for networks of uninhabited vehicles required to operate in dynamic and uncertain environments. The approach integrates ideas from dynamics, combinatorial optimization, teaming, and distributed algorithms. We have presented dynamic vehicle routing algorithms with provable performance guarantees for several important problems. These include adaptive and decentralized implementations, demands with time constraints and priority levels, vehicles with motion constraints, and team forming. These results complement those from the online algorithms literature, in that they characterize average case performance (rather than worst-case), and exploit probabilistic knowledge about future demands.

Dynamic vehicle routing is an active area of research and, in recent years, several directions have been pursued which were not covered in this paper. In [14], [49], we consider moving demands. The work focuses on demands that arrive on a line segment, and move in a perpendicular direction at fixed speed. The problem has applications in perimeter defense as well as robotic pick-and-place operations. In [19], a setup is considered where the information on outstanding demands is provided to the vehicles through limited-range on-board sensors, thus adding a search component to the DVR problem with full information. The work in [50] and [51] considers the dynamic pickup and delivery problem, where each demand consists of a source-destination pair, and the vehicles are responsible for picking up a message at the source, and delivering it to the destination. In [8], the authors consider the case in which each vehicle can serve at most a finite number of demands before returning to a depot for refilling. In [52], a DVR problem is considered involving vehicles whose dynamics can be modeled by state space models that are affine in control and have an output in $\mathbb{R}^2$. Finally, in [21] we consider a setup where the servicing of a demand needs to be done by a vehicle under the supervision of a remotely located human operator.

The dynamic vehicle routing approach presented in this paper provides a new way of studying robotic systems in dynamically changing environments. We have presented results for a wide variety of problems. However, this is by no means a closed book. There is great potential for obtaining more general performance guarantees by developing methods to deal with correlation between demand positions. In addition, there are many other key problems in robotic systems that could benefit from being studied from the perspective presented in this paper. Some examples include search and rescue missions, force protection, map maintenance, and pursuit-evasion.

ACKNOWLEDGMENTS

The authors wish to thank Alessandro Arsie, Shaunak D. Bopardikar, and John J. Enright for numerous helpful discussions about topics related to this paper.

REFERENCES

[1] B. J. Moore and K. M. Passino, "Distributed task assignment for mobile agents," *IEEE Transactions on Automatic Control*, vol. 52, no. 4, pp. 749–753, 2007.

[2] S. L. Smith and F. Bullo, "Monotonic target assignment for robotic networks," *IEEE Transactions on Automatic Control*, vol. 54, no. 9, pp. 2042–2057, 2009.

[3] M. Alighanbari and J. P. How, "A robust approach to the UAV task assignment problem," *International Journal on Robust and Nonlinear Control*, vol. 18, no. 2, pp. 118–134, 2008.

[4] R. W. Beard, T. W. McLain, M. A. Goodrich, and E. P. Anderson, "Coordinated target assignment and intercept for unmanned air vehicles," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 6, pp. 911–922, 2002.

[5] G. Arslan, J. R. Marden, and J. S. Shamma, "Autonomous vehicle-target assignment: A game theoretic formulation," *ASME Journal on Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 584–596, 2007.

[6] P. Toth and D. Vigo, eds., *The Vehicle Routing Problem*. Monographs on Discrete Mathematics and Applications, SIAM, 2001.

[7] D. J. Bertsimas and G. J. van Ryzin, "A stochastic and dynamic vehicle routing problem in the Euclidean plane," *Operations Research*, vol. 39, no. 4, pp. 601–615, 1991.

[8] D. J. Bertsimas and G. J. van Ryzin, "Stochastic and dynamic vehicle routing in the Euclidean plane with multiple capacitated vehicles," *Operations Research*, vol. 41, no. 1, pp. 60–76, 1993.

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

[10] H. N. Psaraftis, "Dynamic vehicle routing problems," in *Vehicle Routing: Methods and Studies* (B. Golden and A. Assad, eds.), pp. 223–248, Elsevier (North-Holland), 1988.

[11] 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*, vol. 14, no. 3, pp. 350–364, 2009.

[12] M. Pavone and E. Frazzoli, "Dynamic vehicle routing with stochastic time constraints," in *IEEE Int. Conf. on Robotics and Automation*, (Anchorage, AK), pp. 1460–1467, May 2010.

[13] 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*, vol. 48, no. 5, pp. 3224–3245, 2010.

[14] 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*, vol. 55, no. 11, pp. 2554–2569, 2010.

[15] M. Pavone, E. Frazzoli, and F. Bullo, "Adaptive and distributed algorithms for vehicle routing in a stochastic and dynamic environment," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1259–1274, 2011.

[16] A. Arsie, K. Savla, and E. Frazzoli, "Efficient routing algorithms for multiple vehicles with no explicit communications," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2302–2317, 2009.

[17] K. Savla, E. Frazzoli, and F. Bullo, "Traveling Salesperson Problems for the Dubins vehicle," *IEEE Transactions on Automatic Control*, vol. 53, no. 6, pp. 1378–1391, 2008.

[18] 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*, vol. 34, no. 4, pp. 1152–1166, 2009.

[19] J. J. Enright and E. Frazzoli, "Cooperative UAV routing with limited sensor range," in *AIAA Conf. on Guidance, Navigation and Control*, (Keystone, CO), pp. AIAA–2006–6208, Aug. 2006.

[20] S. L. Smith and F. Bullo, "The dynamic team forming problem: Throughput and delay for unbiased policies," *Systems & Control Letters*, vol. 58, no. 10-11, pp. 709–715, 2009.

[21] K. Savla, T. Temple, and E. Frazzoli, "Human-in-the-loop vehicle routing policies for dynamic environments," in *IEEE Conf. on Decision and Control*, (Cancún, México), pp. 1145–1150, Dec. 2008.

[22] M. Pavone, *Dynamic Vehicle Routing for Robotic Networks*. PhD thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, June 2010.

[23] D. D. Sleator and R. E. Tarjan, "Amortized efficiency of list update and paging rules," *Communications of the ACM*, vol. 28, no. 2, pp. 202–208, 1985.

[24] S. O. Krumke, W. E. de Paepe, D. Poensgen, and L. Stougie, "News from the online traveling repairman," *Theoretical Computer Science*, vol. 295, no. 1-3, pp. 279–294, 2003.

[25] S. Irani, X. Lu, and A. Regan, "On-line algorithms for the dynamic traveling repair problem," *Journal of Scheduling*, vol. 7, no. 3, pp. 243–258, 2004.

[26] P. Jaillet and M. R. Wagner, "Online routing problems: Value of advanced information and improved competitive ratios," *Transportation Science*, vol. 40, no. 2, pp. 200–210, 2006.

[27] B. Golden, S. Raghavan, and E. Wasil, *The Vehicle Routing Problem: Latest Advances and New Challenges*, vol. 43 of *Operations Research/Computer Science Interfaces*. Springer, 2008.

[28] P. Van Hentenryck, R. Bent, and E. Upfal, "Online stochastic optimization under time constraints," *Annals of Operations Research*, vol. 177, no. 1, pp. 151–183, 2009.

[29] P. K. Agarwal and M. Sharir, "Efficient algorithms for geometric optimization," *ACM Computing Surveys*, vol. 30, no. 4, pp. 412–458, 1998.

[30] Z. Drezner, ed., *Facility Location: A Survey of Applications and Methods*. Series in Operations Research, Springer, 1995.

[31] H. Xu, *Optimal Policies for Stochastic and Dynamic Vehicle Routing Problems*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1995.

[32] S. Bespamyatnikh, D. Kirkpatrick, and J. Snoeyink, "Generalizing ham sandwich cuts to equitable subdivisions," *Discrete & Computational Geometry*, vol. 24, no. 4, pp. 605–622, 2000.

[33] M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo, "Distributed algorithms for environment partitioning in mobile robotic networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 8, pp. 1834–1848, 2011.

[34] J. D. Papastavrou, "A stochastic and dynamic routing policy using branching processes with state dependent immigration," *European Journal of Operational Research*, vol. 95, pp. 167–177, 1996.

[35] L. E. Dubins, "On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, pp. 497–516, 1957.

[36] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006. Available at `http://planning.cs.uiuc.edu`.

[37] U. Boscain and B. Piccoli, *Optimal Syntheses for Control Systems on 2-D Manifolds*. Mathématiques et Applications, Springer, 2004.

[38] L. Pallottino and A. Bicchi, "On optimal cooperative conflict resolution for air traffic management systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 221–231, 2000.

[39] C. Tomlin, I. Mitchell, and R. Ghosh, "Safety verification of conflict resolution manoeuvres," *IEEE Transactions on Intelligent Transportation Systems*, vol. 2, no. 2, pp. 110–120, 2001.

[40] P. Chandler, S. Rasmussen, and M. Pachter, "UAV cooperative path planning," in *AIAA Conf. on Guidance, Navigation and Control*, (Denver, CO), pp. AIAA–2000–4370, Aug. 2000.

[41] C. Schumacher, P. R. Chandler, S. J. Rasmussen, and D. Walker, "Task allocation for wide area search munitions with variable path length," in *American Control Conference*, (Denver, CO), pp. 3472–3477, 2003.

[42] E. Zemel, "Probabilistic analysis of geometric location problems," *Annals of Operations Research*, vol. 1, no. 3, pp. 215–238, 1984.

[43] K. Savla, F. Bullo, and E. Frazzoli, "Traveling Salesperson Problems for a double integrator," *IEEE Transactions on Automatic Control*, vol. 54, no. 4, pp. 788–793, 2009.

[44] K. Savla and E. Frazzoli, "On endogenous reconfiguration for mobile robotic networks," in *Workshop on Algorithmic Foundations of Robotics*, (Guanajuato, México), pp. 53–67, Dec. 2008.

[45] E. K. P. Chong, C. M. Kreucher, and A. O. Hero III, "Monte-Carlo-based partially observable Markov decision process approximations for adaptive sensing," in *Int. Workshop on Discrete Event Systems*, (Göteborg, Sweden), pp. 173–180, May 2008.

[46] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, vol. 21 of *Algorithmics and Combinatorics*. Springer, 4 ed., 2007.

[47] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, 2000.

[48] 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*, vol. 26, no. 3, pp. 255–272, 2007.

[49] S. L. Smith, S. D. Bopardikar, and F. Bullo, "A dynamic boundary guarding problem with translating demands," in *IEEE Conf. on Decision and Control and Chinese Control Conference*, (Shanghai, China), pp. 8543–8548, Dec. 2009.

[50] 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*, vol. 53, no. 6, pp. 1419–1433, 2008.

[51] M. R. Swihart and J. D. Papastavrou, "A stochastic and dynamic model for the single-vehicle pick-up delivery problem," *European Journal of Operational Research*, vol. 114, no. 3, pp. 447–464, 1999.

[52] S. Itani, E. Frazzoli, and M. A. Dahleh, "Dynamic travelling repairperson problem for dynamic systems," in *IEEE Conf. on Decision and Control*, (Cancún, México), pp. 465–470, Dec. 2008.

[53] J. Beardwood, J. Halton, and J. Hammersly, "The shortest path through many points," in *Proceedings of the Cambridge Philosophy Society*, vol. 55, pp. 299–327, 1959.

[54] J. M. Steele, "Probabilistic and worst case analyses of classical problems of combinatorial optimization in Euclidean space," *Mathematics of Operations Research*, vol. 15, no. 4, pp. 749–770, 1990.

[55] A. G. Percus and O. C. Martin, "Finite size and dimensional dependence of the Euclidean traveling salesman problem," *Physical Review Letters*, vol. 76, no. 8, pp. 1188–1191, 1996.

[56] R. C. Larson and A. R. Odoni, *Urban Operations Research*. Prentice Hall, 1981.

[57] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, "On the solution of traveling salesman problems," in *Documenta Mathematica, Journal der Deutschen Mathematiker-Vereinigung*, (Berlin, Germany), pp. 645–656, Aug. 1998. Proceedings of the International Congress of Mathematicians, Extra Volume ICM III.

[58] N. Christofides, "Worst-case analysis of a new heuristic for the traveling salesman problem," Tech. Rep. 388, Carnegie Mellon University, Pittsburgh, PA, Apr. 1976.

[59] S. Arora, "Nearly linear time approximation scheme for Euclidean TSP and other geometric problems," in *IEEE Symposium on Foundations of Computer Science*, (Miami Beach, FL), pp. 554–563, Oct. 1997.

[60] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations Research*, vol. 21, pp. 498–516, 1973.

[61] N. Megiddo and K. J. Supowit, "On the complexity of some common geometric location problems," *SIAM Journal on Computing*, vol. 13, no. 1, pp. 182–196, 1984.

[62] C. H. Papadimitriou, "Worst-case and probabilistic analysis of a geometric location problem," *SIAM Journal on Computing*, vol. 10, no. 3, pp. 542–557, 1981.

APPENDIX

A. Asymptotic Properties of the Traveling Salesman Problem in the Euclidean Plane

Let D be a set of n points in $\mathbb{R}^d$ and let $\mathrm{TSP}(D)$ denote the minimum length of a tour through all the points in D; by convention, $\mathrm{TSP}(\emptyset) = 0$. Assume that the locations of the n points are random variables independently and identically distributed in a compact set $\mathcal{Q}$; in [53], [54] it is shown that there exists a constant $\beta_{\mathrm{TSP},d}$ such that, almost surely,

$$\lim_{n\to+\infty} \frac{\mathrm{TSP}(D)}{n^{1-1/d}} = \beta_{\mathrm{TSP},d} \int_{\mathcal{Q}} \bar{\varphi}(q)^{1-1/d}\, dq, \qquad (20)$$

where $\bar{\varphi}$ is the density of the absolutely continuous part of the point distribution. For the case $d = 2$, the constant $\beta_{\mathrm{TSP},2}$ has been estimated numerically as $\beta_{\mathrm{TSP},2} \simeq 0.7120 \pm 0.0002$ [55]. Notice that the bound (20) holds for all compact sets: the shape of the set only affects the convergence rate to the limit. According to [56], if $\mathcal{Q}$ is a "fairly compact and fairly convex" set in the plane, then equation (20) provides a "good" estimate of the optimal TSP tour length for values of n as low as 15.

B. Tools for Solving TSPs

The TSP is known to be NP-complete, which suggests that there is no general algorithm capable of finding the optimal tour in an amount of time polynomial in the size of the input. Even though the exact optimal solution of a large TSP can be very hard to compute, several exact and heuristic algorithms and software tools are available for the numerical solution of TSPs.

The most advanced TSP solver to date is arguably `concorde` [57]. Polynomial-time algorithms are available for constant-factor approximations of TSP solutions, among which we mention Christofides' algorithm [58]. On a more theoretical side, Arora proved the existence of polynomial-time approximation schemes for the TSP, providing a $(1+\varepsilon)$ constant-factor approximation for any $\varepsilon > 0$ [59].

A modified version of the Lin-Kernighan heuristic [60] is implemented in `linkern`; this powerful solver yields approximations in the order of 5% of the optimal tour cost very quickly for many instances. For example, in our numerical experiments on a machine with a 2GHz Intel Core Duo processor, approximations of random TSPs with 10,000 points typically required about twenty seconds of CPU time.[4]

[4]The `concorde` and `linkern` solvers are freely available for academic research use at `www.tsp.gatech.edu/concorde/index.html`.

We presented algorithms that require online solutions of possibly large TSPs. Practical implementations of the algorithms would rely on heuristics or on polynomial-time approximation schemes, such as Lin-Kernighan's or Christofides'. If a constant-factor approximation algorithm is used, the effect on the asymptotic performance guarantees of our algorithms can be simply modeled as a scaling of the constant $\beta_{\mathrm{TSP},d}$.

C. Properties of the Multi-Median Function

In this subsection, we state certain useful properties of the multi-median function, introduced in Section III-A. The multi-median function can be written as

$$\begin{aligned} H_m(P,\mathcal{Q}) &:= \mathbb{E}\left[\min_{k\in\{1,\dots,m\}} \|p_k - q\|\right] \\ &= \sum_{k=1}^{m} \int_{V_k(P)} \|p_k - q\|\varphi(q)\,dq, \end{aligned}$$

where $\mathcal{V}(P) = (V_1(P),\dots,V_m(P))$ is the Voronoi partition of the set $\mathcal{Q}$ generated by the points P. It is straightforward to show that the map $P \mapsto H_1(P,\mathcal{Q})$ is differentiable and strictly convex on $\mathcal{Q}$. Therefore, it is a simple computational task to compute $P_1^*(\mathcal{Q})$. On the other hand, when $m > 1$, the map $P \mapsto H_m(P,\mathcal{Q})$ is differentiable (whenever $(p_1,\dots,p_m)$ are distinct) but not convex, thus making the solution of the continuous m-median problem hard in the general case. It is known [29], [61] that the discrete version of the m-median problem is NP-hard for $d \geq 2$. However, one can provide tight bounds on the map $m \mapsto H_m^*(\mathcal{Q})$. This problem is studied thoroughly in [62] for square regions and in [42] for more general compact regions. It is shown that, in the limit $m \to +\infty$, $H_m^*(\mathcal{Q})\sqrt{m} \to c_{\mathrm{hex}}\sqrt{|\mathcal{Q}|}$ almost surely [42], where $c_{\mathrm{hex}} \approx 0.377$ is the first moment of a hexagon of unit area about its center. This optimal asymptotic value is achieved by placing the m points at the centers of the hexagons in a regular hexagonal lattice within $\mathcal{Q}$ (the honeycomb heuristic). Working towards the above result, it is also shown [42], [18] that for any $m \in \mathbb{N}$:

$$0.3761\sqrt{\frac{|\mathcal{Q}|}{m}} \leq H_m^*(\mathcal{Q}) \leq c(\mathcal{Q})\sqrt{\frac{|\mathcal{Q}|}{m}}, \tag{21}$$

where $c(\mathcal{Q})$ is a constant depending on the shape of $\mathcal{Q}$.