DYNAMIC VEHICLE ROUTING WITH PRIORITY CLASSES OF STOCHASTIC DEMANDS[*]

STEPHEN L. SMITH[†], MARCO PAVONE[‡], FRANCESCO BULLO[†] , AND EMILIO FRAZZOLI[‡]

Abstract. In this paper we introduce a dynamic vehicle routing problem in which there are multiple vehicles and multiple priority classes of service demands. Service demands of each priority class arrive in the environment randomly over time and require a random amount of on-site service that is characteristic of the class. To service a demand, one of the vehicles must travel to the demand location and remain there for the required on-site service time. The quality of service provided to each class is given by the expected delay between the arrival of a demand in the class, and that demand's service completion. The goal is to design a routing policy for the service vehicles which minimizes a convex combination of the delays for each class. First, we provide a lower bound on the achievable values of the convex combination of delays. Then, we propose a novel routing policy and analyze its performance under heavy load conditions (i.e., when the fraction of time the service vehicles spend performing on-site service approaches one). The policy performs within a constant factor of the lower bound, where the constant depends only on the number of classes, and is independent of the number of vehicles, the arrival rates of demands, the on-site service times, and the convex combination coefficients.

Key words. vehicle routing, optimization, multi-agent systems, heterogeneous systems

1. Introduction.

A classical problem in queueing theory is that of priority queueing [9]. In the simplest setup, customers arrive at a single server sequentially over time. Each customer is a member of either the high-priority, or the low-priority class. High priority customers and low priority customers form separate queues. The goal is to provide the highest possible quality of service to the high priority queue (Q_1) while maintaining stability of the low priority queue (Q_2). That is, the goal is to minimize the expected delay for high-priority customers while keeping the length of the low-priority queue finite. When both the customer inter-arrival times and the customer service times are distributed exponentially, the preemptive priority policy is known to be optimal [9]:

> When Q_1 is nonempty, serve high priority customers; when Q_1 is empty, serve low-priority customers. If a high priority customer arrives while serving Q_2, then preempt service and immediately begin serving the high-priority customer.

A more general two-class queueing problem is to minimize a convex combination of the service delays for high- and low-priority customers

$$cD_1 + (1-c)D_2, \quad \text{where } c \in (0,1).$$

In this case an optimal policy can be created by using a mixed policy that spends fraction c of the time serving Q_1 as the high-priority queue, and fraction $(1-c)$ serving

[*]This research was partially supported by the National Science Foundation, through grants #0705451 and #0705453, by the Office of Naval Research through grant #N00014-07-1-0721, and by the AirForce Office of Scientific Research through grant #FA9550-07-1-0528. Preliminary versions of this work appeared in [20] and [16].

[†]S. L. Smith and F. Bullo are with the Center for Control, Dynamical Systems and Computation, Department of Mechanical Engineering, University of California, Santa Barbara, CA 93106 (`{stephen,bullo}@engineering.ucsb.edu`).

[‡]M. Pavone and E. Frazzoli are with the Laboratory for Information and Decision Systems, Aeronautics and Astronautics Department, Massachusetts Institute of Technology, Cambridge, MA 02139 (`{pavone,frazzoli}@mit.edu`).

Q_2 as though it is the high-priority queue [7]. The set of achievable delays has also been studied in the more general setting of queueing networks [2].

In this paper we consider an m-class, n-service-vehicle spatial queueing problem, called *dynamic vehicle routing with priority classes*. Demands for service arrive sequentially over time in a compact environment $\mathcal{E}$ in the plane. Each demand is a member of one of m priority classes. Upon arrival, each demand assumes a location in $\mathcal{E}$, and requires a class-dependent amount of on-site service time. To service a demand, one of the n vehicles must travel to the demand location and perform the on-site service. If we specify a policy by which the vehicles serve demands, then the expected delay for demands of class α, denoted D_α, is the expected amount of time between a demands arrival and its service completion. Then, given convex combination coefficients $c_1, \ldots, c_m > 0$, the goal is to find the vehicle routing policy that minimizes $c_1 D_1 + \cdots + c_m D_m$. By increasing the coefficients for certain classes, a higher priority level can be given to their demands. This problem has important applications in areas such as UAV surveillance, where targets are given different priority levels based on their urgency or potential importance [1].

When there is only one class of demands, the problem in this paper is known as the Dynamic Traveling Repairperson Problem (DTRP) [18, 3, 4]. The DTRP was arguably the first member of a class of problems known as dynamic vehicle routing (DVR); for other DVR problems see [10]. In [3, 4], optimal DTRP policies are proposed in heavy load (i.e., when the fraction of time the service vehicles spend performing on-site service approaches one), and in light load (i.e., when the fraction of time the service vehicles spends performing on-site service approaches zero). In [13], a simple unified DTRP policy is presented for both light and heavy load conditions. Recently, there has been an increased interest in DVR among researchers in robotic motion planning, as it provides a powerful method for completing spatially distributed tasks that are generated in real-time. In particular, DVR results have recently been obtained on decentralized policies [8, 15], moving demands [6], impatient demands [14], demands requiring pickup and delivery [23], and demands which require teams of vehicles [19]. Other related vehicle routing problems include the orienteering problem and discounted-reward-TSP problems [5], the dynamic assignment problem [21], and spatial queueing in the context of urban operations research [11].

The main contribution of this paper is to introduce dynamic vehicle routing with priority classes. We derive a lower bound on the achievable values of the convex combination of delays, and propose a novel policy in which each class of demands is served separately from the others. We show that in heavy load, the policy performs within a constant factor $2m^2$ of the lower bound. Thus, the constant factor is independent of the number of vehicles, the arrival rates of demands, the on-site service times, and the convex combination coefficients. To establish the constant factor, we proceed in a similar manner as [13, 14] and develop a system of nonlinear inequality-based recursive equations on the expected number of outstanding demands. We then utilize a novel proof technique to upper bound the limiting number of outstanding demands, which relies on constructing a set of linear equality-based recursive equations to bound trajectories. We present an improvement on the policy in which classes of similar priority are merged together. We also perform extensive simulations and introduce an effective heuristic improvement called the tube heuristic.

The paper is organized as follows. In Section 2 we give some asymptotic properties of optimal Euclidean traveling salesperson tours. In Section 2.2 we formalize the problem and in Section 3 we derive a lower bound on the achievable delay. In Section 4

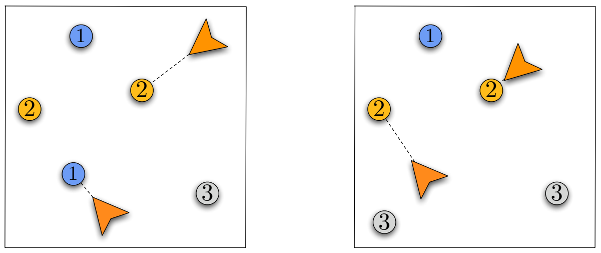

FIG. 2.1. *A depiction of the problem for two vehicles and three priority classes. Left figure: One vehicle is moving to a class 1 demand, and the other to a class 2 demand. Right figure: The bottom vehicle has serviced the class 1 demand and is moving to a class 2 demand. A new class 3 demand has arrived.*

we introduce and analyze the Separate Queues policy, and present the improvements given by *queue merging* and the *tube heuristic*. Finally, in Section 5 we present simulation results.

2. Background and Problem Statement.

In this section we summarize the asymptotic properties of optimal Euclidean traveling salesperson tours, and formalize dynamic vehicle routing problem with priority classes.

2.1. The Euclidean Traveling Salesperson Problem.

Given a set Q of N points in $\mathbb{R}^2$, the Euclidean traveling salesperson problem (TSP) is to find the minimum-length tour of Q (i.e., the shortest closed path through all points). Let $\mathrm{TSP}(Q)$ denote the minimum length of a tour through all the points in Q. Assume that the locations of the N points are random variables independently and identically distributed, uniformly in a compact set $\mathcal{E}$ with area $|\mathcal{E}|$; in [22] it is shown that there exists a constant β_{TSP} such that, almost surely,

$$\lim_{N\to+\infty} \frac{\mathrm{TSP}(Q)}{\sqrt{N}} = \beta_{\mathrm{TSP}}\sqrt{|\mathcal{E}|}. \tag{2.1}$$

The constant β_{TSP} has been estimated numerically as $\beta_{\mathrm{TSP}} \approx 0.7120 \pm 0.0002$, [17]. The bound in equation (2.1) holds for all compact sets $\mathcal{E}$, and the shape of $\mathcal{E}$ only affects the convergence rate to the limit. In [11], the authors note that if $\mathcal{E}$ is “fairly compact [square] and fairly convex”, then equation (2.1) provides an adequate estimate of the optimal TSP tour length for values of N as low as 15.

2.2. Problem Statement.

Consider a compact environment $\mathcal{E}$ in the plane with area $|\mathcal{E}|$. The environment contains n vehicles, each with maximum speed v. Demands of type $\alpha \in \{1, \dots, m\}$ (also 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 $\mathcal{E}$. An α-demand is serviced when the vehicle spends an on-site service time at the demand location, which is generally distributed with finite mean $\bar{s}_\alpha$.

Consider the arrival of the ith α-demand. The service delay for the ith demand, $D_\alpha(i)$, is the time elapsed between its arrival and its service completion. The wait time is defined as $W_\alpha(i) := D_\alpha(i) - s_\alpha(i)$, where $s_\alpha(i)$ is the on-site service time required by demand i. A policy for routing the vehicles is said to be *stable* if the expected number of demands in the system for each class is bounded uniformly at all times. A necessary condition for the existence of a stable policy is

$$\varrho := \frac{1}{n}\sum_{\alpha=1}^{m} \lambda_\alpha \bar{s}_\alpha < 1. \tag{2.2}$$

The *load factor* ϱ is a standard quantity in queueing theory [9], and is used to capture the fraction of time the n servers (vehicles) must be busy in any stable policy. In general, it is difficult to study a queueing system for all values of $\varrho \in [0, 1)$, and a common technique is to focus on the limiting regimes of $\varrho \to 1^-$, referred to as the *heavy load* regime, and $\varrho \to 0^+$, referred to as the *light load* regime.

Given a stable policy P the steady-state service delay is defined as $D_\alpha(P) := \lim_{i\to+\infty} \mathbb{E}[D_\alpha(i)]$, and the steady-state wait time is $W_\alpha(P) := D_\alpha(P) - \bar{s}_\alpha$. Thus, for a stable policy P, the *average delay per demand* is

$$D(P) = \frac{1}{\Lambda}\sum_{\alpha=1}^{m} \lambda_\alpha D_\alpha(P),$$

where $\Lambda := \sum_{\alpha=1}^{m} \lambda_\alpha$. The average delay per demand is the standard cost functional for queueing systems with multiple classes of demands. Notice that we can write $D(P) = \sum_{\alpha=1}^{m} c_\alpha D_\alpha(P)$ with $c_\alpha = \lambda_\alpha/\Lambda$. Thus, we can model priority among classes by allowing any convex combination of $D_1, \dots, D_m$. If $c_\alpha > \lambda_\alpha/\Lambda$, then the delay of α-demands is being weighted more heavily than in the average case. Thus, the quantity $c_\alpha\Lambda/\lambda_\alpha$ gives the priority of α demands compared to that given in the average delay case. Without loss of generality we can assume that priority classes are labeled so that

$$\frac{c_1}{\lambda_1} \geq \frac{c_2}{\lambda_2} \geq \cdots \geq \frac{c_m}{\lambda_m}, \tag{2.3}$$

implying that if $\alpha < \beta$ for some $\alpha, \beta \in \{1, \dots, m\}$, then the priority of α-demands is at least as high as that of β-demands. With these definitions, we are now ready to state our problem.

Problem Statement: Let Π be the set of all causal, stable and stationary policies for dynamic vehicle routing with priority classes. Given the coefficients $c_\alpha > 0$, $\alpha \in \{1, \dots, m\}$, with $\sum_{\alpha=1}^{m} c_\alpha = 1$, and satisfying equation (2.3), let $D(P) := \sum_{\alpha=1}^{m} c_\alpha D_\alpha(P)$ be the cost of policy $P \in \Pi$. Then, the problem is to determine a vehicle routing policy P^*, if one exists, such that

$$D(P^*) = \inf_{P\in\Pi} D(P). \tag{2.4}$$

We let D^* denote the right-hand side of equation (2.4). A policy P for which $D(P)/D^*$ is bounded has a *constant-factor guarantee*. If $\limsup_{\varrho\to1^-} D(P)/D^* = \kappa < +\infty$, then the policy P has a *heavy-load constant-factor guarantee* of κ. In this paper we focus on the heavy-load regime, and look for policies with a heavy-load constant-factor guarantee that is *independent* of the number of vehicles, the arrival rates of demands, the on-site service times, and the convex combination coefficients. In the light-load regime, existing policies for the dynamic traveling repairperson can be used, as is summarized in the following remark.

Remark 2.1 (Light-load regime). *In light load, $\varrho \to 0^+$, optimal policies have been developed for the dynamic traveling repairperson problem (i.e., the single-class* *dynamic vehicle routing problem). These policies rely on the computation of a set of n-median locations for the environment $\mathcal{E}$; that is, a set of n positions $\mathcal{Q}^* \subset \mathcal{E}$, that minimize*

$$\mathbb{E}\left[\min_{\mathbf{q}\in\mathcal{Q}^*} \|\mathbf{q}-\mathbf{q}_0\|\right],$$

where $\mathbf{q}_0$ is a uniformly distributed location in $\mathcal{E}$. In particular, the n Stochastic Queue Median (nSQM) policy, first introduced in [4], can be described as follows:

> *Place one vehicle at each of the n-median locations of $\mathcal{E}$. When a demand arrives, assign it to the closest median location, and to the corresponding vehicle. Have each vehicle service its demands in the first-come-first-served order, returning to its median location after each service is completed.*

In fact, by following the proof in [4], one can show that the nSQM policy is an optimal policy for dynamic vehicle routing with priority classes. The proof of this statement is omitted in the interest of brevity, and we refer interested readers to [4] for details.●

3. Lower Bound in Heavy Load.

3. Lower Bound in Heavy Load. In this section we present two lower bounds on the delay in equation (2.4). The first holds only in heavy load (i.e., as $\varrho \to 1^-$), while the second (less tight) bound holds for all ϱ.

THEOREM 3.1 (Heavy-load lower bound). *For every routing policy P,*

$$D(P) \geq \frac{\beta_{\mathrm{TSP}}^2 |\mathcal{E}|}{2n^2 v^2 (1-\varrho)^2} \sum_{\alpha=1}^{m} \left(c_\alpha + 2 \sum_{j=\alpha+1}^{m} c_j \right) \lambda_\alpha \quad \textit{as } \varrho \to 1^-, \tag{3.1}$$

where $c_1, \ldots, c_m$ satisfy equation (2.3).

Before proving Theorem 3.1 let us quickly comment on the form of equation (3.1). The right-hand side of equation (3.1) approaches $+\infty$ as $\varrho \to 1^-$. Thus, one should more formally write the inequality with $D(P)(1-\varrho)^2$ on the left-hand side, so that the right-hand side is finite. However, this makes the presentation less readable, and thus, henceforth we adhere to the less formal but more transparent style of equation (3.1).

Proof. Consider a tagged demand i of type α, and let us quantify its total service requirement. The demand requires on-site service time $s_\alpha(i)$. Let us denote by $d_\alpha(i)$ the distance from the location of the demand served prior to i, to i's location. In order to compute a lower bound on the wait time, we will allow "remote" servicing of some of the demands. For an α-demand i that can be serviced remotely, the travel distance $d_\alpha(i)$ is zero (i.e., a service vehicle can service the ith α-demand from any location by simply stopping for the on-site service time $s_\alpha(i)$). Thus, the wait time for the modified remote servicing problem provides a lower bound on the wait time for the problem of interest. To formalize this idea, we introduce the variables $r_\alpha \in \{0,1\}$ for each $\alpha \in \{1,\ldots,m\}$. If $r_\alpha = 0$, then α-demands can be serviced remotely. If $r_\alpha = 1$, then α-demands must be serviced on location. We assume that $r_\alpha = 1$ for at least one $\alpha \in \{1,\ldots,m\}$. Thus, the total service requirement of α-demand i is $r_\alpha d_\alpha(i) + s_\alpha(i)$. The steady-state expected service requirement is $r_\alpha \bar{d}_\alpha + s_\alpha$, where $\bar{d}_\alpha := \lim_{i\to+\infty} \mathbb{E}[d_\alpha(i)]$. In order to maintain stability of the system we must require

$$\frac{1}{n}\sum_{\alpha=1}^{m} \lambda_\alpha \left(\frac{r_\alpha \bar{d}_\alpha}{v} + \bar{s}_\alpha \right) < 1. \tag{3.2}$$

Applying the definition of ϱ in equation (2.2), we write equation (3.2) as

$$\sum_{\alpha=1}^{m} r_\alpha \lambda_\alpha \bar{d}_\alpha < (1-\varrho)nv. \tag{3.3}$$

For a stable policy P, let $\bar{N}_\alpha$ represent the steady-state expected number of unserviced α-demands. Then, the expected total number of outstanding demands that require on-site service (i.e., cannot be serviced remotely) is given by $\sum_{j=1}^{m} r_j \bar{N}_j$. We now apply a result from the dynamic traveling repairperson problem (see [25], page 23) which states that in heavy load ($\varrho \to 1^-$), if the steady-state number of outstanding demands is N, then a lower bound on expected travel distance between demands is $(\beta_{\mathrm{TSP}}/\sqrt{2})\sqrt{|\mathcal{E}|/N}$. Applying this result we have that

$$\bar{d}_\alpha \geq \frac{\beta_{\mathrm{TSP}}}{\sqrt{2}} \sqrt{\frac{|\mathcal{E}|}{\sum_j r_j \bar{N}_j}} =: \bar{d}, \tag{3.4}$$

for each $\alpha \in \{1,\dots,m\}$. Combining equations (3.3) and (3.4),

$$\frac{\sum_\alpha r_\alpha \lambda_\alpha}{nv(1-\varrho)} < \frac{1}{\bar{d}}.$$

Applying the definition of $\bar{d}$, squaring both sides, and rearranging we obtain

$$\frac{\beta_{\mathrm{TSP}}^2}{2} \frac{|\mathcal{E}|(\sum_\alpha r_\alpha \lambda_\alpha)^2}{n^2 v^2 (1-\varrho)^2} < \sum_\alpha r_\alpha \bar{N}_\alpha.$$

From Little's law, $\bar{N}_\alpha = \lambda_\alpha W_\alpha$ for each $\alpha \in \{1,\dots,m\}$, and thus

$$\sum_\alpha r_\alpha \lambda_\alpha W_\alpha > \frac{\beta_{\mathrm{TSP}}^2}{2} \frac{|\mathcal{E}|}{n^2 v^2 (1-\varrho)^2} \left(\sum_\alpha r_\alpha \lambda_\alpha\right)^2. \tag{3.5}$$

Recalling that $W_\alpha = D_\alpha - \bar{s}_\alpha$ and $r_\alpha \in \{0,1\}$ for each $\alpha \in \{1,\dots,m\}$, we see that equation (3.5) gives us $2^m - 1$ constraints on the feasible values of $D_1(P),\dots,D_m(P)$. Hence, a lower bound on D^* can be found by minimizing $\sum_{\alpha=1}^{m} W_\alpha$ subject to the constraints in equation (3.5). We can lower bound the solution to the optimization problem by minimizing the cost function subject to only a subset of the $2^m - 1$ constraints. In particular, consider the following linear program

$$\begin{aligned} &\textbf{minimize} && \sum_{\alpha=1}^{m} c_\alpha W_\alpha, \\ &\textbf{subject to} && \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ \lambda_1 & \lambda_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & & 0 \\ \lambda_1 & \lambda_2 & \lambda_3 & \cdots & \lambda_m \end{bmatrix} \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_m \end{bmatrix} \geq \Psi \begin{bmatrix} \lambda_1^2 \\ (\lambda_1+\lambda_2)^2 \\ \vdots \\ (\lambda_1+\cdots+\lambda_m)^2 \end{bmatrix}, \end{aligned}$$

where

$$\Psi := \frac{\beta_{\mathrm{TSP}}^2}{2} \frac{|\mathcal{E}|}{n^2 v^2 (1-\varrho)^2}.$$

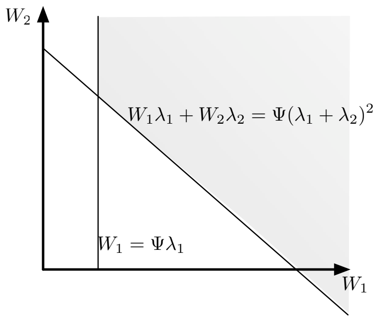

FIG. 3.1. *The feasible region of the linear program for 2 queues. When class* 1 *is of higher priority, the solution is given by the corner. Otherwise, the solution is* $-\infty$.

The above problem is feasible (see Fig. 3.1), it has only one basic feasible solution, and it is of the form: minimize $\mathbf{c}^T\mathbf{W}$ subject to $A\mathbf{W} \geq \mathbf{b}$. Thus, either the problem is unbounded, or the solution $\mathbf{W}^*$ is given by the basic feasible solution. To establish boundedness we consider the dual problem: maximize $\mathbf{b}^T\mathbf{y}$ subject to $A^T\mathbf{y} = \mathbf{c}$ and $\mathbf{y} \geq 0$. By the Duality Theorem of Linear Programming [12], if the dual is feasible, then the minimization problem is bounded. To check feasibility of the dual, we solve for $A^T\mathbf{y} = \mathbf{c}$, with $\mathbf{y} \geq 0$, to obtain

$$\begin{aligned} y_\alpha &= \frac{c_\alpha}{\lambda_\alpha} - \frac{c_{\alpha+1}}{\lambda_{\alpha+1}} \geq 0 \quad \text{for all } \alpha \in \{1,\dots,m-1\},\\ y_m &= \frac{c_m}{\lambda_m} \geq 0. \end{aligned}$$

Thus, the dual is feasible if and only if the priority classes are labeled as in equation (2.3). When equation (2.3) is satisfied, the minimization problem is bounded, and its solution $(W_1^*,\dots,W_m^*)$ is given by

$$W_\alpha^* = \frac{\Psi}{\lambda_\alpha}\left((\lambda_1+\cdots+\lambda_\alpha)^2 - (\lambda_1+\cdots+\lambda_{\alpha-1})^2\right) = \Psi\left(\lambda_\alpha + 2\sum_{j=1}^{\alpha-1}\lambda_j\right).$$

(In fact, this is the solution of the full optimization problem with $2^m - 1$ constraints. This fact can be verified, somewhat tediously, by writing the dual of the full problem and directly computing its solution. To shorten the presentation we omit the direct computation and use the above technique.) The optimal value of the cost function, and thus the lower bound on D^*, is given by

$$\sum_{\alpha=1}^m c_\alpha W_\alpha^* = \Psi\sum_{\alpha=1}^m c_\alpha\left(\lambda_\alpha + 2\sum_{j=1}^{\alpha-1}\lambda_j\right) = \Psi\sum_{\alpha=1}^m\left(c_\alpha + 2\sum_{j=\alpha+1}^m c_j\right)\lambda_\alpha.$$

Applying the definition of Ψ we obtain the desired result. $\square$

REMARK 3.2 (Lower bound for all $\varrho \in [0,1)$). *With slight modifications, it is possible to obtain a less tight lower bound valid for all values of* ϱ. *In the above derivation, the assumption that* $\varrho \to 1^-$ *is used only in equation* (3.4). *It is possible to use, instead, a lower bound valid for all* $\varrho \in [0,1)$ *(see [4]):*

$$\bar{d}_\alpha \geq \gamma\sqrt{\frac{|\mathcal{E}|}{\sum_\alpha r_\alpha N_\alpha + n/2}},$$

where $\gamma = 2/(3\sqrt{2\pi}) \approx 0.266$. Using this bound we obtain the same linear program as in the proof of Theorem 3.1, with the difference that Ψ is now a function given by

$$\Psi(x) := \frac{\gamma^2|\mathcal{E}|}{n^2 v^2 (1-\varrho)^2} x - \frac{n}{2}.$$

Following the procedure in the proof of Theorem 3.1

$$W_1^* = \frac{\gamma^2|\mathcal{E}|}{n^2 v^2 (1-\varrho)^2} \lambda_1 - \frac{n}{2\lambda_1}$$
$$W_\alpha^* = \frac{\gamma^2|\mathcal{E}|}{n^2 v^2 (1-\varrho)^2} \left(\lambda_\alpha + 2 \sum_{j=1}^{\alpha-1} \lambda_j \right),$$

for each $\alpha \in \{2, \dots, m\}$. Finally, for every policy P, $D_\alpha(P) \geq W_\alpha^* + \bar{s}_\alpha$, and thus

$$D(P) \geq \frac{\gamma^2|\mathcal{E}|}{n^2 v^2 (1-\varrho)^2} \sum_{\alpha=1}^{m} \left(\left(c_\alpha + 2 \sum_{j=\alpha+1}^{m} c_j \right) \lambda_\alpha \right) - \frac{nc}{2\lambda_1} + \sum_{\alpha=1}^{m} c_\alpha \bar{s}_\alpha, \tag{3.6}$$

for all $\varrho \in [0,1)$ under the labeling in equation (2.3). •

In the remainder of the paper we design a policy and establish a constant-factor guarantee relative to the heavy load lower bound.

4. Separate Queues Policy.

In this section we introduce and analyze the Separate Queues (SQ) policy. We show that this policy is within a factor $2m^2$ of the lower bound in heavy load.

To present the SQ policy we need some notation. We assume vehicle $k \in \{1, \dots, n\}$ has a service region $R^{[k]} \subseteq \mathcal{E}$, such that $\{R^{[1]}, \dots, R^{[n]}\}$ forms a partition of the environment $\mathcal{E}$. In general the partition could be time varying, but for the description of the SQ policy this will not be required. We assume that information on outstanding demands of type $\alpha \in \{1, \dots, m\}$ in region $R^{[k]}$ at time t is summarized as a finite set of demand positions $Q_\alpha^{[k]}(t)$ with $N_\alpha^{[k]}(t) := \operatorname{card}(Q_\alpha^{[k]}(t))$. Demands of type α with location in $R^{[k]}$ are inserted in the set $Q_\alpha^{[k]}$ as soon as they are generated. Removal from the set $Q_\alpha^{[k]}$ requires that service vehicle k moves to the demand location, and provides the on-site service. The SQ policy is described in Algorithm 1. Fig. 4.1 shows an illustrative example of the SQ policy. In the first two frames the vehicle is servicing only class 1 (circle shaped) demands, whereas in the third frame, the vehicle is servicing class 2 (diamond shaped) demands.

4.1. Stability Analysis of the SQ Policy in Heavy Load.

In this section we analyze the SQ policy in heavy load, i.e., as $\varrho \to 1^-$. In the SQ policy each region $R^{[k]}$ has equal area, and contains a single vehicle. Thus, the n vehicle problem in a region of area $|\mathcal{E}|$ has been turned into n independent single-vehicle problems, each in a region of area $|\mathcal{E}|/n$, with arrival rates λ_α/n. To determine the performance of the policy we need only study the performance in a single region k. For simplicity of notation we omit the label k. We refer to the time instant t_i in which the vehicle computes a new TSP tour as the epoch i of the policy; we refer to the time interval between epoch i and epoch $i+1$ as the ith iteration and we will refer to its length as T_i. Finally, let $N_\alpha(t_i) := N_{\alpha,i}$, $\alpha \in \{1, \dots, m\}$, be the number of outstanding α-demands at beginning of iteration i.

Algorithm 1: Separate Queues (SQ) Policy

```
   Assumes: A probability distribution $\mathbf{p} = [p_1, \dots, p_m]$.
1  Partition $\mathcal{E}$ into $n$ equal area regions and assign one vehicle to each region.
2  foreach vehicle-region pair $k$ do
3      if the set $\cup_\alpha Q_\alpha^{[k]}$ is empty then
4          Move vehicle toward the median of its own region until a demand arrives.
5      else
6          Select $Q \in \{Q_1^{[k]}, \dots, Q_m^{[k]}\}$ according to $\mathbf{p}$.
7          if $Q$ is empty then
8              Reselect until $Q$ is nonempty.
9          Compute TSP tour through all demands in $Q$.
10         Service $Q$ following the TSP tour, starting at the demand closest to the vehicle's current position.
11     Repeat.
12 Optimize over $\mathbf{p}$.
```

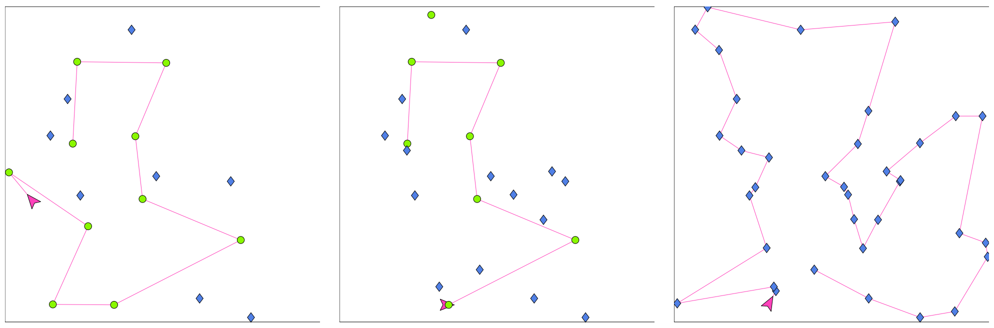

FIG. 4.1. *A representative simulation of the SQ policy for one vehicle and two priority classes. Circle shaped demands are high priority, and diamond shaped are low priority. The vehicle is marked by a chevron shaped object and TSP tour is shown in a solid line.*

The following straightforward lemma, similar to Lemma 1 in [13], will be essential in deriving our main results.

LEMMA 4.1 (Number of outstanding demands). *In heavy load (i.e., $\varrho \to 1^-$), after a transient, the number of demands serviced in a single tour of the vehicle in the SQ policy is very large with high probability (i.e., the number of demands tends to $+\infty$ with probability that tends to 1, as ϱ approaches 1^-).*

Proof. Consider the case where the vehicle moves with infinite velocity (i.e., $v \to +\infty$); then the system is reduced to the usual M/G/1 queue. The infinite-velocity system has fewer demands (for every $\alpha \in \{1, \dots, m\}$) waiting in queue. A known result on M/G/1 queues [24] states that, after transients, the total number of demands, as $\varrho \to 1^-$, is very large with high probability. Thus, in the SQ policy, the number of demands in all m classes is very large with high probability. In particular, this implies that, after a transient, the number of demands is very large with high probability at the instances when the vehicle starts a new tour. ∎

Let TS_j be the event that Q_j is selected for service at iteration i of the SQ policy.

By the total probability law

$$\mathbb{E}\left[N_{\alpha,i+1}\right] = \sum_{j=1}^{m} p_j \mathbb{E}\left(N_{\alpha,i+1} | TS_j\right), \quad \alpha \in \{1, \dots, m\},$$

where the conditioning is with respect to the task being performed during iteration i. During iteration i of the policy, demands arrive according to independent Poisson processes. Call $N_{\alpha,i}^{\text{new}}$ the α-demands ($\alpha \in \{1, \dots, m\}$) newly arrived during iteration i; then, by definition of the SQ policy

$$\mathbb{E}\left(N_{\alpha,i+1} | TS_j\right) = \begin{cases} \mathbb{E}\left(N_{\alpha,i}^{\text{new}} | TS_j\right), & \text{if } \alpha = j \\ \mathbb{E}\left(N_{\alpha,i} | TS_j\right) + \mathbb{E}\left(N_{\alpha,i}^{\text{new}} | TS_j\right), & \text{otherwise.} \end{cases}$$

By the law of iterated expectation, we have $\mathbb{E}\left(N_{\alpha,i}^{\text{new}} | TS_j\right) = (\lambda_\alpha / n) \mathbb{E}\left(T_i | TS_j\right)$. Moreover, since the number of demands outstanding at the beginning of iteration i is independent of the task that will be chosen, we have $\mathbb{E}\left(N_{\alpha,i} | TS_j\right) = \mathbb{E}\left[N_{\alpha,i}\right]$. Thus we obtain

$$\mathbb{E}\left(N_{\alpha,i+1} | TS_j\right) = \begin{cases} \frac{\lambda_\alpha}{n} \mathbb{E}\left(T_i | TS_j\right), & \text{if } \alpha = j \\ \mathbb{E}\left[N_{\alpha,i}\right] + \frac{\lambda_\alpha}{n} \mathbb{E}\left(T_i | TS_j\right), & \text{otherwise.} \end{cases}$$

Therefore, we are left with computing the conditional expected values of T_i. The length of T_i is given by the time needed by the vehicle to travel along the TSP tour plus the time spent to service demands. Assuming i large enough, Lemma (4.1) holds, and we can apply equation (2.1) to estimate from the quantities $N_{\alpha,i}$, $\alpha \in \{1, \dots, m\}$, the length of the TSP tour at iteration i. Conditioning on TS_j (when only demands of type j are serviced), we have

$$\begin{aligned} \mathbb{E}\left(T_i | TS_j\right) &= \frac{\beta_{\text{TSP}} \sqrt{|\mathcal{E}|/n}}{v} \mathbb{E}\left(\sqrt{N_{j,i}} | TS_j\right) + \mathbb{E}\left(\textstyle\sum_{k=1}^{N_{j,i}} s_{j,k} | TS_j\right) \\ &\leq \frac{\beta_{\text{TSP}} \sqrt{|\mathcal{E}|/n}}{v} \sqrt{\mathbb{E}\left[N_{j,i}\right]} + \mathbb{E}\left[N_{j,i}\right] \bar{s}_j, \end{aligned}$$

where we have

- applied equation (2.1);
- applied Jensen's inequality for concave functions, in the form $\mathbb{E}\left[\sqrt{X}\right] \leq \sqrt{\mathbb{E}\left[X\right]}$;
- removed the conditioning on TS_j, since the random variables $N_{\alpha,i}$ are independent from *future* events, and in particular from the choice of the task at iteration i; and
- used the crucial fact that the on-site service times are independent from the number of outstanding demands.

Collecting the above results (and using the shorthand $\bar{X}$ to indicate $\mathbb{E}\left[X\right]$, where X is any random variable), we have

$$\bar{N}_{\alpha,i+1} \leq (1 - p_\alpha) \bar{N}_{\alpha,i} + \sum_{j=1}^{m} p_j \frac{\lambda_\alpha}{n} \left[\frac{\beta_{\text{TSP}} \sqrt{|\mathcal{E}|}}{\sqrt{n} v} \sqrt{\bar{N}_{j,i}} + \bar{N}_{j,i} \bar{s}_j \right], \tag{4.1}$$

for each $\alpha \in \{1, \dots, m\}$. The m inequalities above describe a system of recursive relations that describe an upper bound on $\bar{N}_{\alpha,i}$, $\alpha \in \{1, \dots, m\}$. The following theorem bounds the values to which they converge.

THEOREM 4.2 (Steady-state queue length). *For every set of initial conditions $\{\bar N_{\alpha,0}\}_{\alpha\in\{1,\dots,m\}}$, the trajectories $i \mapsto \bar N_{\alpha,i}$, $\alpha \in \{1,\dots,m\}$, resulting from equations* (4.1)*, satisfy*

$$\limsup_{i\to+\infty} \bar N_{\alpha,i} \le \frac{\beta_{\mathrm{TSP}}^2 |\mathcal{E}|}{n^3 v^2 (1-\varrho)^2} \frac{\lambda_\alpha}{p_\alpha} \left(\sum_{j=1}^m \sqrt{\lambda_j p_j} \right)^2, \quad \text{as } \varrho \to 1^-.$$

Proof. Define $q_j := 1 - p_j$ and let $\hat\lambda_\alpha$ denote the arrival rate in region $R^{[k]}$. Thus $\hat\lambda_\alpha := \lambda_\alpha / n$ for each $\alpha \in \{1,\dots,m\}$. Let $x(i) := (\bar N_{1,i}, \bar N_{2,i}, \dots, \bar N_{m,i}) \in \mathbb{R}^m$ and define two matrices

$$A := \begin{bmatrix} \hat\lambda_1 p_1 \bar s_1 + q_1 & \hat\lambda_1 p_2 \bar s_2 & \dots & \hat\lambda_1 p_m \bar s_m \\ \hat\lambda_2 p_1 \bar s_1 & \hat\lambda_2 p_2 \bar s_2 + q_2 & \dots & \hat\lambda_2 p_m \bar s_m \\ \vdots & & \ddots & \vdots \\ \hat\lambda_m p_1 \bar s_1 & \hat\lambda_m p_2 \bar s_2 & \dots & \hat\lambda_m p_m \bar s_m + q_m \end{bmatrix},$$

and

$$B := \frac{\beta_{\mathrm{TSP}} \sqrt{|\mathcal{E}|}}{\sqrt{n} v} \begin{bmatrix} \hat\lambda_1 p_1 & \hat\lambda_1 p_2 & \dots & \hat\lambda_1 p_m \\ \hat\lambda_2 p_1 & \hat\lambda_2 p_2 & \dots & \hat\lambda_2 p_m \\ \vdots & & \ddots & \vdots \\ \hat\lambda_m p_1 & \hat\lambda_m p_2 & \dots & \hat\lambda_m p_m \end{bmatrix}.$$

Then, letting the relation "$\le$" in $\mathbb{R}^m$ denote the product order of m copies of $\mathbb{R}$ (in other words, for v, $w \in \mathbb{R}^m$, the relation $v \le w$ is interpreted component-wise), equations (4.1) can be written as

$$x(i+1) \le A x(i) + B \begin{bmatrix} \sqrt{x_1(i)} \\ \sqrt{x_2(i)} \\ \vdots \\ \sqrt{x_m(i)} \end{bmatrix} =: f(x(i)), \tag{4.2}$$

where $f : \mathbb{R}_{\ge 0} \to \mathbb{R}_{\ge 0}$, and $x_j(i)$, $j \in \{1,\dots,m\}$, are the components of vector $x(i)$. We refer to the discrete system in equation (4.2) as System-X. Next we define two auxiliary systems, System-Y and System-Z. We define System-Y as

$$y(i+1) = f(y(i)). \tag{4.3}$$

System-Y is, therefore, equal to System-X, with the exception that we replaced the inequality with an equality.

Pick, now, any $\varepsilon > 0$. From Young's inequality

$$\sqrt{a} \le \frac{1}{4\varepsilon} + \varepsilon a, \quad \text{for all } a \in \mathbb{R}_{\ge 0}. \tag{4.4}$$

Hence, for $i \mapsto y(i) \in \mathbb{R}^m_{\ge 0}$, the equation (4.3) becomes

$$\begin{aligned} y(i+1) &\le A y(i) + B\Big(\frac{1}{4\varepsilon} \mathbf{1}_m + \varepsilon\, y(i)\Big) \\ &= \Big(A + \varepsilon B\Big) y(i) + \frac{1}{4\varepsilon} B \mathbf{1}_m. \end{aligned}$$

where $\mathbf{1}_m$ is the vector $(1, 1, \dots, 1)^{\mathrm{T}} \in \mathbb{R}^m$. Next, define System-Z as

$$z(i+1) = \Big(A + \varepsilon B\Big) z(i) + \frac{1}{4\varepsilon} B \mathbf{1}_m =: g(z(i)). \tag{4.5}$$

The proof now proceeds as follows. First, we show that if $x(0) = y(0) = z(0)$, then

$$x(i) \leq y(i) \leq z(i), \quad \text{for all } i \geq 0. \tag{4.6}$$

Second, we show that the trajectories of System-Z are bounded; this fact, together with equation (4.6), implies that also trajectories of System-Y and System-X are bounded. Third, and last, we will compute $\limsup_{i \to +\infty} y(i)$; this quantity, together with equation (4.6), will yield the desired result.

Let us consider the first issue. We have $y(1) = f(y(0))$ and $z(1) = g(z(0))$. Since, by assumption $z(0) = y(0)$, we have that $g(z(0)) = g(y(0)) \geq f(y(0))$, where the last inequality follows from equation (4.4) and by definition of f and g . Therefore, we get $y(1) \leq z(1)$. Then, we have $y(2) = f(y(1))$ and $z(2) = g(z(1))$. Since $z(1), y(1) \in \mathbb{R}^m_{\geq 0}$, and the elements in matrices A and B are all non-negative, then $y(1) \leq z(1)$ implies $g(y(1)) \leq g(z(1))$. Using similar arguments, we can write $z(2) \geq g(y(1)) \geq f(y(1)) = x(2)$; therefore, we get $y(2) \leq z(2)$. Then, it is immediate by induction that $y(i) \leq z(i)$ for all $i \geq 0$.

Similarly, we have $x(1) \leq f(x(0)) = f(y(0)) = y(1)$, where we have used the assumption $x(0) = y(0)$. Then, we get $x(1) \leq y(1)$. Since $x(1), y(1) \in \mathbb{R}^m_{\geq 0}$, the elements in matrices A and B are nonnegative, and by the monotonicity of $\sqrt{\cdot}$, then $x(1) \leq y(1)$ implies $f(x(1)) \leq f(y(1))$. Therefore, we can write $x(2) \leq f(x(1)) \leq f(y(1)) = y(2)$; thus, we get $x(2) \leq y(2)$. Then, it is immediate to show by induction that $x(i) \leq y(i)$ for all $i \geq 0$, and equation (4.6) holds.

We now turn our attention to the second issue, namely boundedness of trajectories for System-Z (in equation (4.5)). Notice that System-Z is a discrete-time linear system. The eigenvalues of A are characterized in the following lemma.

Lemma 4.3. *The eigenvalues of A are real and have magnitude strictly less than* 1 *(i.e., A is a stable matrix).*

Proof. Let $w \in \mathbb{C}^m$ be an eigenvector of A, and $\mu \in \mathbb{C}$ be the corresponding eigenvalue. Then we have $Aw = \mu w$. Define $r := (p_1 \bar{s}_1, p_2 \bar{s}_2, \dots, p_m \bar{s}_m)$. Then the m eigenvalue equations are

$$\hat{\lambda}_j \, w \cdot r + q_j w_j = \mu \, w_j, \quad j \in \{1, \dots, m\}, \tag{4.7}$$

where $w \cdot r$ is the scalar product of vectors w and r, and w_j is the jth component of w.

There are two possible cases. If $w \cdot r = 0$, then equation (4.7) becomes $q_j \, w_j = \mu \, w_j$, for all j. Since $w \neq 0$, there exists j^* such that $w_j^* \neq 0$; thus, we have $\mu = q_{j^*}$. Since $q_{j^*} \in \mathbb{R}$ and $0 < q_{j^*} < 1$, we have that μ is real and $|\mu| < 1$.

Assume, now, that $w \cdot r \neq 0$. This implies that $\mu \neq q_j$ and $w_j \neq 0$ for all j, thus we can write, for all j,

$$w_j = \frac{\hat{\lambda}_j}{\mu - q_j} \, w \cdot r, \tag{4.8}$$

and hence

$$w_j = \frac{\hat{\lambda}_j}{\hat{\lambda}_1} \frac{\mu - q_1}{\mu - q_j} w_1.$$

Therefore, (4.8) can be rewritten as

$$\sum_{j=1}^{m} \frac{r_j \hat{\lambda}_j}{\mu - q_j} = 1. \tag{4.9}$$

Equation (4.9) implies that the eigenvalues are real. To see this, write $\mu = a + ib$, where i is the imaginary unit: then

$$\sum_{j=1}^{m} \frac{r_j \hat{\lambda}_j}{a + ib - q_j} = \sum_{j=1}^{m} \frac{r_j \hat{\lambda}_j [(a - q_j) - ib]}{(a - q_j)^2 + b^2}.$$

Thus equation (4.9) implies

$$b \underbrace{\sum_{j=1}^{m} \frac{r_j \hat{\lambda}_j}{(a - q_j)^2 + b^2}}_{>0} = 0,$$

that is, $b = 0$. Equation (4.9) also implies that the eigenvalues (that are real) have magnitude strictly less than 1. Indeed, assume, by contradiction, that $\mu \geq 1$. Then we have $\mu - q_j \geq 1 - q_j > 0$ (recall that the eigenvalues are real and $0 < q_j < 1$) and we can write

$$\sum_{j=1}^{m} \frac{r_j \hat{\lambda}_j}{\mu - q_j} \leq \sum_{j=1}^{m} \frac{r_j \hat{\lambda}_j}{1 - q_j} = \sum_{j=1}^{m} \bar{s}_j \hat{\lambda}_j = \varrho < 1,$$

which is a contradiction. Assume, again by contradiction, that $\mu \leq -1$. In this case we trivially get another contradiction $\sum_{j=1}^{m} r_j \hat{\lambda}_j / (\mu - q_j) < 0$, since $\mu - q_j < 0$. ◻

Hence, $A \in \mathbb{R}^{m \times m}$ has eigenvalues strictly inside the unit disk, and since the eigenvalues of a matrix depend continuously on the matrix entries, there exists a sufficiently small $\varepsilon > 0$ such that the matrix $A + \varepsilon B$ has eigenvalues strictly inside the unit disk. Accordingly, each solution $i \mapsto z(i) \in \mathbb{R}^m_{\geq 0}$ of System-Z converges exponentially fast to the unique equilibrium point

$$z^* = \Big(I_m - A - \varepsilon B\Big)^{-1} \frac{1}{4\varepsilon} B \mathbf{1}_m. \tag{4.10}$$

Combining equation (4.6) with the previous statement, we see that the solutions $i \mapsto x(i)$ and $i \mapsto y(i)$ are bounded. Thus

$$\limsup_{i \to +\infty} x(i) \leq \limsup_{i \to +\infty} y(i) < +\infty. \tag{4.11}$$

Finally, we turn our attention to the third issue, namely the computation of $y := \limsup_{i \to +\infty} y(i)$. Taking the lim sup of the left- and right-hand sides of equation (4.3), and noting that

$$\limsup_{i \to +\infty} \sqrt{y_\alpha(i)} = \sqrt{\limsup_{i \to +\infty} y_\alpha(i)} \quad \text{for } \alpha \in \{1, 2, \ldots, m\},$$

since $x \mapsto \sqrt{x}$ is continuous and strictly monotone increasing on $\mathbb{R}_{>0}$, we obtain that

$$y_\alpha = (1 - p_\alpha) y_\alpha + \hat{\lambda}_\alpha \sum_{j=1}^{m} p_j \bigg(\frac{\beta_{\text{TSP}} \sqrt{|\mathcal{E}|}}{\sqrt{n} v} \sqrt{y_j} + \bar{s}_j y_j \bigg).$$

Rearranging we obtain

$$p_\alpha y_\alpha = \hat{\lambda}_\alpha \sum_{j=1}^{m} p_j \left(\frac{\beta_{\mathrm{TSP}} \sqrt{|\mathcal{E}|}}{\sqrt{n} v} \sqrt{y_j} + \bar{s}_j y_j \right). \tag{4.12}$$

Dividing $p_\alpha y_\alpha$ by $p_1 y_1$ we obtain

$$y_\alpha = \frac{\hat{\lambda}_\alpha p_1}{\hat{\lambda}_1 p_\alpha} y_1. \tag{4.13}$$

Combining equations (4.12) and (4.13), we obtain

$$p_1 y_1 = \varrho\, p_1 y_1 + \frac{\beta_{\mathrm{TSP}} \sqrt{|\mathcal{E}|}}{\sqrt{n} v} \sqrt{p_1 \hat{\lambda}_1 y_1} \sum_{j=1}^{m} \sqrt{\hat{\lambda}_j p_j}$$

Thus, recalling that $\hat{\lambda}_\alpha = \lambda_\alpha / n$, we obtain

$$y_\alpha = \frac{\beta_{\mathrm{TSP}}^2 |\mathcal{E}|}{n^3 v^2 (1-\varrho^2)} \frac{\lambda_\alpha}{p_\alpha} \left(\sum_{j=1}^{m} \sqrt{\lambda_j p_j} \right)^2 .$$

Noting that from equation (4.11), $\limsup_{i\to+\infty} N_{\alpha,i} \leq y_\alpha$, we obtain the desired result. ☐

4.2. Delay of the SQ Policy in Heavy Load. From Theorem 4.2, and using Little's law, the delay of α-demands satisfies

$$\begin{aligned} D_\alpha(\mathrm{SQ}) &\leq \frac{n}{\lambda_\alpha} \limsup_{i\to+\infty} \bar{N}_{\alpha,i} + \bar{s}_\alpha \\ &= \frac{\beta_{\mathrm{TSP}}^2 |\mathcal{E}|}{n^2 v^2 (1-\varrho)^2} \frac{1}{p_\alpha} \left(\sum_{j=1}^{m} \sqrt{\lambda_j p_j} \right)^2 . \end{aligned}$$

Thus, the delay (as defined in equation (2.4)) of the SQ policy, satisfies

$$D(\mathrm{SQ}) \leq \frac{\beta_{\mathrm{TSP}}^2 |\mathcal{E}|}{n^2 v^2 (1-\varrho)^2} \sum_{\alpha=1}^{m} \frac{c_\alpha}{p_\alpha} \left(\sum_{i=1}^{m} \sqrt{\lambda_i p_i} \right)^2 , \quad \text{as } \varrho \to 1^-. \tag{4.14}$$

With this expression we prove our main result on the performance of the SQ policy.

THEOREM 4.4 (SQ policy performance). *As $\varrho \to 1^-$, the delay of the SQ policy is within a factor $2m^2$ of the optimal delay. This factor is independent of the arrival rates $\lambda_1, \dots, \lambda_m$, coefficients $c_1, \dots, c_m$, service times $\bar{s}_1, \dots, \bar{s}_m$, and the number of vehicles n.*

Proof. We would like to compare the performance of this policy with the lower bound. To do this, consider setting

$$p_\alpha := c_\alpha \quad \text{for each } \alpha \in \{1, \dots, m\}.$$

Defining $\Psi := \beta_{\mathrm{TSP}}^2 |\mathcal{E}| / (n^2 v^2 (1-\varrho)^2)$, equation (4.14) can be written as

$$D(\mathrm{SQ}) \leq \Psi m \left(\sum_{i=1}^{m} \sqrt{c_i \lambda_i} \right)^2 .$$

Next, the lower bound in equation (3.1) is

$$D^* \geq \frac{\Psi}{2} \sum_{i=1}^{m} \left(c_i + 2 \sum_{j=i+1}^{m} c_j \right) \lambda_i \geq \frac{\Psi}{2} \sum_{i=1}^{m} (c_i \lambda_i) .$$

Thus, comparing the upper and lower bounds

$$\frac{D(\text{SQ})}{D^*} \leq 2m \frac{\left(\sum_{i=1}^m \sqrt{c_i \lambda_i}\right)^2}{\sum_{i=1}^m (c_i \lambda_i)}. \tag{4.15}$$

Letting $x_i := \sqrt{c_i \lambda_i}$, and $\mathbf{x} := [x_1, \ldots, x_m]$, the numerator of the fraction in equation (4.15) is $\|\mathbf{x}\|_1^2$, and the denominator is $\|\mathbf{x}\|_2^2$. But the one- and two-norms of a vector $\mathbf{x} \in \mathbb{R}^m$ satisfy $\|\mathbf{x}\|_1 \leq \sqrt{m}\|\mathbf{x}\|_2$. Thus,

$$\frac{D(\text{SQ})}{D^*} \leq 2m \left(\frac{\|\mathbf{x}\|_1}{\|\mathbf{x}\|_2} \right)^2 \leq 2m^2, \quad \text{as } \varrho \to 1^-,$$

and the policy is a $2m^2$-factor approximation. $\square$

REMARK 4.5 (Relation to RP policy in [20]). *For $m = 2$ the SQ policy is within a factor of 8 of the optimal. This improves on the factor of 12 obtained for the Randomized Priority (RP) policy in [20]. However, it appears that the RP policy bound is not tight, since for two classes, simulations indicate it performs no worse than the SQ policy.* •

4.3. Separate Queues Policy with Queue Merging. In this section we propose a modification the SQ policy based on *queue merging*. Queue merging is guaranteed to never increase the upper bound on the expected delay, and in certain instances it significantly decreases the upper bound. The modification can be used when we have a modest number of classes (fewer than, say, 20), which encompasses most applications of interest.

To motivate the modification, consider the case when all classes have equal priority (i.e., $c_1/\lambda_1 = \cdots = c_m/\lambda_m$), and we use the probability assignment $p_\alpha = c_\alpha$ for each class α. Then, the upper bound for the Separate Queues policy in equation (4.14) becomes

$$\Psi m(\lambda_1 + \cdots + \lambda_m),$$

where $\Psi := \beta_{\text{TSP}}^2 |\mathcal{E}| / (n^2 v^2 (1 - \varrho)^2)$.

On the other hand, if we ignore priorities, merge the m classes into a single class, and run the SQ policy on the merged class (i.e., at each iteration, service all outstanding demands in $\mathcal{E}$ via the TSP tour), then the upper bound becomes

$$\Psi(\lambda_1 + \cdots + \lambda_m).$$

Thus, there is a factor of m separating the two upper bounds. This is due to the fact that the basic SQ policy services each of the m classes separately, even when they have the same priority.

The above discussion motivates the addition of queue merging to the SQ policy. We define a *merge configuration* to be a partition of m classes $\{1, \ldots, m\}$ into ℓ

sets $C_1, \dots, C_\ell$, where $\ell \in \{1, \dots, m\}$. The upper bound for a merge configuration $\{C_1, \dots, C_\ell\}$ is

$$\Psi \ell \left(\sum_i^{\ell} \sqrt{\sum_{\alpha \in C_i} c_\alpha \sum_{\beta \in C_i} \lambda_\beta} \right)^2. \tag{4.16}$$

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

Separate Queues with Merging Policy

```
1 Find the merge configuration $\{C_1, \dots, C_\ell\}$ which minimizes equation (4.16).
2 Run the Separate Queues policy on $\ell$ 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 (4.16) in step 1 of the SQ with Merging policy, one must search over all possible partitions of a set of m elements. The number of partitions is given by the Bell Number B_m which is defined recursively as $B_m = \sum_{k=0}^{m-1} B_k \binom{m-1}{k}$. Thus, the search becomes infeasible for more than 10 classes.

If the search space is too large, then one can limit the search to all partitions such that if $i < j$, then each class in C_i has higher priority than all classes in C_j. This is the set of partitions in which only adjacent classes are merged. For m classes, there are 2^{m-1} such merge configurations, which is significantly less than the Bell number B_m, but is still infeasible more than, say, 20 classes.

4.4. The "Tube" Heuristic for Improving Performance. We now introduce a simple heuristic improvement for the SQ policy that can be used for implementation. The heuristic improvement is as follows:

Tube Heuristic: When following the tour in step 10 of the SQ policy, 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. A similar difficulty arises when attempting to analyze the nearest neighbor policy [3]. However, we can demonstrate the effectiveness of this heuristic through simulations.

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 the area of the "tube" of width 2ϵ centered around a tour that passes through the card(Q) demands in Q, has area upper bounded by $2\epsilon\beta_{\text{TSP}}\sqrt{\text{card}(Q)|\mathcal{E}|}$. While following the tour, a vehicle will deviate to service no more than $2\epsilon\beta_{\text{TSP}}\sqrt{\text{card}(Q)/|\mathcal{E}|}(\bar{N}_1 + \dots + \bar{N}_m)$ demands. Finally, since the vehicle will have to travel no more than 2ϵ to service each demand in the "tube," we see that ϵ should scale as

$$\epsilon \sim \sqrt{\frac{f|\mathcal{E}|}{\bar{N}_1 + \dots + \bar{N}_m}},$$

where $\bar{N}_\alpha$ is expected number of α-demands in the environment, and f is the fractional increase in tour length (e.g., 10%).

Fig. 4.2 shows numerical results for the Tube Heuristic for a single unit speed vehicle in a square environment with side length 50. The simulation is performed

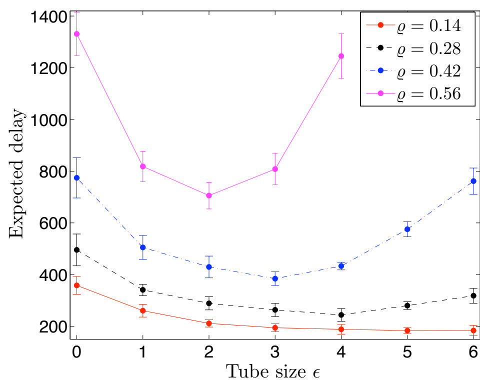

FIG. 4.2. *The tube heuristic for two classes of demands with $c = 0.8$, $\lambda_2 = 6\lambda_1$, and several different load factors ϱ. The delay at $\epsilon = 0$ corresponds to the basic SQ policy.*

for two classes of demands with $c = 0.8$, $\lambda_2 = 6\lambda_1$, and several different load factors ϱ. Each experimental data point represents the average of the steady state delay of ten runs, where each run consists of 200 iterations of the SQ policy. To ensure convergence to steady state and avoid effects due to the transient response, only the last 50 iterations in each run are used to calculate the delay. The basic policy is shown in left-most data points (i.e., $\epsilon = 0$). Fig. 4.2 demonstrates that as the load factor increases, the value of ϵ should be chosen smaller in order to achieve the best performance. Table 4.1 shows the improvement in expected delay when using the tube heuristic. The heuristic appears to decrease the delay by a factor of 2. One should note that the heuristic is difficult to accurately simulate for high load factors. This is due to the additional computations required to determine if a newly arrived demand lies within a distance ϵ of the current tour. A more sophisticated implementation of

Load factor ϱ	Delay	Best ϵ	Delay with best ϵ	Heuristic improvement
0.14	358 (34)	5	183 (11)	0.51 (0.16)
0.28	496 (61)	4	244 (25)	0.49 (0.23)
0.42	774 (78)	3	384 (26)	0.50 (0.17)
0.56	1330 (84)	2	706 (52)	0.53 (0.14)
0.70	3380 (357)	1	1770 (121)	0.52 (0.17)

TABLE 4.1

A comparison between the expected delay of the basic SQ policy, and the SQ policy with the tube heuristic. The values in brackets give the standard deviation of the corresponding table entry.

tube heuristic is to define an ϵ_α for each $\alpha \in \{1, \dots, m\}$, where the magnitude of ϵ_α is proportional to its priority, and thus proportional to the probability p_α.

5. Simulations and Discussion.

In this section we discuss, through the use of simulations, the performance of the SQ policy with the probability assignment $p_\alpha := c_\alpha$, for each $\alpha \in \{1, \dots, m\}$. In particular, we study (i) the tightness of the upper bound in equation (4.14), (ii) conditions for which the gap between the lower bound in equation (3.1) and the upper bound in equation (4.14) is maximized, (iii) the suboptimality of the probability assignment $p_\alpha = c_\alpha$, and (iv) the difference in performance between the SQ policy and a policy that merges all classes together irrespective of priorities. Simulations of the SQ policy were performed using `linkern`[1] as a solver to generate approximations to the optimal TSP tour.

5.1. Tightness of the Upper Bound. We consider one vehicle, four classes of demands, and several values of the load factor ϱ. For each value of ϱ we perform 100 runs. In each run we uniformly randomly generate arrival rates $\lambda_1, \dots, \lambda_m$, convex combination coefficients $c_1, \dots, c_m$, and on-site service times $\bar{s}_1, \dots, \bar{s}_m$, and normalize the values such that the constraints $\sum_{\alpha=1}^m \lambda_\alpha \bar{s}_\alpha = \varrho$ and $\sum_{\alpha=1}^m c_\alpha = 1$ are satisfied. In each run we iterate the SQ policy 4000 times, and compute the steady-state expected delay by considering the number of demands in the last 1000 iterations. For each value of ϱ we compute the ratio χ between the expected delay and the theoretical upper bound in equation (4.14). Table 5.1 reports the ratio, its standard deviation, and its minimum and maximum values for each ϱ value. One can see that the upper bound provides a reasonable approximation for load factors as low as $\varrho = 0.75$.

Load factor (ϱ)	$\mathbb{E}[\chi]$	σ_χ	$\max \chi$	$\min \chi$
0.75	0.803	0.092	1.093	0.354
0.8	0.778	0.108	0.943	0.256
0.85	0.773	0.111	1.150	0.417
0.9	0.733	0.159	1.162	0.203
0.95	0.716	0.131	0.890	0.257

TABLE 5.1
Ratio χ between experimental results and upper bound for various values of ϱ.

5.2. Maximum deviation from lower bound. In Theorem 4.4 we showed that the SQ policy performs within a factor of $2m^2$ of the lower bound for all initial conditions. The difference between the upper bound equation (4.14) and the lower bound in equation (3.1) can be made arbitrarily close to $2m^2$ by choosing $\lambda_1 \ll \lambda_2 \ll \dots \ll \lambda_m$ and $c_1 \gg c_2 \gg \dots \gg c_m$, with $\lambda_\alpha c_\alpha = a$, for each $\alpha \in \{1, \dots, m\}$ and for some positive constant a. In this case, the upper bound is equal to $Bm^3 a$ and the lower bound is approximately $Bma/2$. To test the deviation experimentally we simulated the SQ policy for several values of ϱ with initial conditions of $\lambda_m = b\lambda_{m-1} = b^2\lambda_{m-1} = \dots = b^{m-1}\lambda_1$ and $c_1 = bc_2 = \dots = b^{m-1}c_m$, where $b = 2$. Fig. 5.1 shows that the experimentally determined ratio of delays (averaged over 10 simulation runs) indeed increases proportionally to m^2.

5.3. Suboptimality of the Approximate Probability Assignment. To prove Theorem 4.4 we used the probability assignment

$$p_\alpha := c_\alpha \quad \text{for each } \alpha \in \{1, \dots, m\}. \tag{5.1}$$

However, one would like to select $[p_1, \dots, p_m] =: \mathbf{p}$ that minimizes the right-hand side of equation (4.14). The minimization of the right-hand side of equation (4.14) is a constrained multi-variable nonlinear optimization problem over $\mathbf{p}$, that is, in m dimensions. Thus, for a general m class problem, solving the optimization problem is difficult. However, for two classes of demands the optimization is over a single variable p_1 (with the constraint that $p_2 = 1 - p_1$), and it can be readily solved. A comparison of optimized upper bound, denoted upbd_{opt}, with the upper bound obtained using the probability assignment in equation (5.1), denoted upbd_c, is shown in Fig. 5.2. In

[1]The TSP solver `linkern` is freely available for academic research use at `http://www.tsp.gatech.edu/concorde.html`.

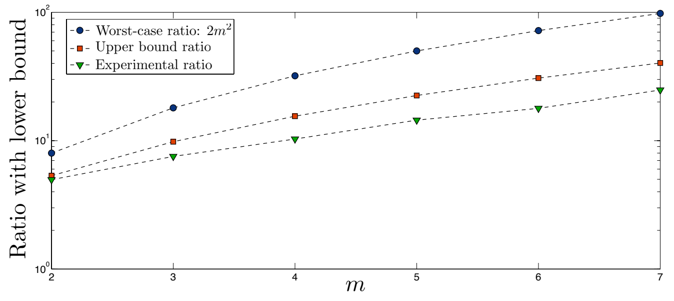

FIG. 5.1. *Experimental results for the SQ policy in worst-case conditions;* $\varrho = 0.85$ *and* $\lambda_1 = 1$.

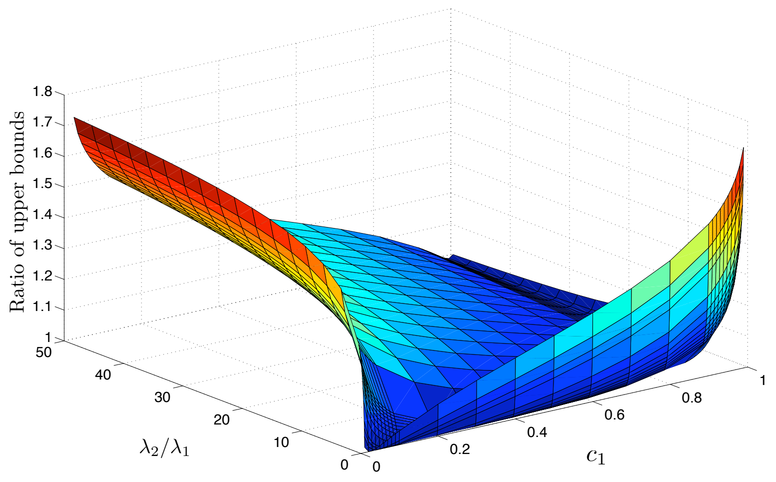

FIG. 5.2. *The ratios* $\text{upbd}_c/\text{upbd}_{opt}$ *for 2 classes of demands.*

this figure the ratio of upper bounds is bounded by two.

For $m > 2$ we approximate the solution of the optimization problem as follows. For each value of m we perform 1000 runs. In each run we randomly generate $\lambda_1, \ldots, \lambda_m$, $c_1, \ldots, c_m$, and five sets of initial probability assignments $\mathbf{p}_1, \ldots, \mathbf{p}_5$. From each initial probability assignment we use a line search to locally optimize the probability assignment. We take the ratio between upbd_c and the least upper bound $\text{upbd}_{\text{local opt}}$ obtained from the five locally optimized probability assignments. We also record the maximum variation in the five locally optimized upper bounds. This is summarized in Table 5.2. The second column shows the largest ratio obtained over the 1000 runs. The third column shows the largest % variation in the 1000 runs. The assignment in equation (5.1) seems to perform within a factor of two of the optimized assignment, and the optimization appears to converge to values close to a global optimum since all five random conditions converge to values that are within $\sim 0.1\%$ of each other on every run.

5.4. The Complete Merge Policy. As described in Section 4.3, a naive policy for our problem is to ignore priorities, merge all classes into a single class, and repeatedly form TSP tours through all outstanding demands. We call this policy the

Number of classes (m)	$\text{upbd}_c/\text{upbd}_{\text{local opt}}$	Max. % variation in ratio
3	1.60	0.12
4	1.51	0.04
5	1.51	0.08
6	1.74	0.02
7	1.88	0.08
8	1.63	0.15

TABLE 5.2
Ratio of upper bound with $p_\alpha = c_\alpha$ for each $\alpha \in \{1, \dots, m\}$ and the upper bound with a locally optimized probability assignment.

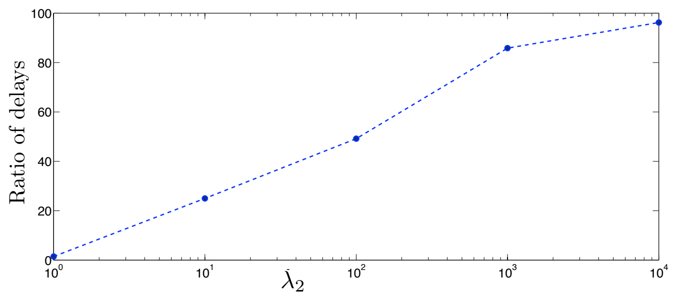

FIG. 5.3. *Ratio of experimental delays between Complete Merge policy and SQ policy as a function of λ_2, with $m = 2$, $\lambda_1 = 1$, c = 0.995 and $\varrho = 0.9$.*

Complete Merge (CM) policy. In this section we briefly verify that the performance of the Complete Merge policy can be arbitrarily bad when compared to the SQ policy. In addition, the conditions for which the Complete Merge policy performs poorly are precisely the conditions of interest for most applications; when low priority demands arrive much more frequently than high priority demands. To see this, define the total arrival rate $\Lambda := \sum_{\alpha=1}^m \lambda_\alpha$ and total mean on-site service $\bar{S} := \sum_{\alpha=1}^m \lambda_\alpha$. Using the upper bounds in [3], we immediately obtain that $D(\text{CM}) \leq \frac{\beta_{\text{TSP}}^2 |\mathcal{E}| \Lambda}{n^2 v^2 (1-\varrho)^2}$. Thus, the ratio $D(\text{CM})/D(\text{SQ})$ can be made arbitrarily large by choosing $\lambda_1 \ll \lambda_2 \ll \cdots \ll \lambda_m$ and $c_1 \gg c_2 \gg \cdots \gg c_m$. Fig. 5.3 shows the experimentally obtained ratio between the delay of the Complete Merge policy and that of the SQ policy (averaged over 10 simulation runs), and verifies the poor performance of the CM policy.

6. Conclusions.

In this paper we introduced a dynamic vehicle routing problem with priority classes. We captured the priority levels of classes by writing the system delay as a convex combination of the delay of each class. We determined a lower bound on the achievable values of the convex combination of the class delays. We then presented the Separate Queues (SQ) policy and showed that it performs within a constant factor of the lower bound, which depends only on the number of the classes. We believe that it may be possible to improve the lower bound and remove, or reduce, the constant factor's dependence on the number of classes. For future work we are interested in combining the aspects of multi-class vehicle routing with problems in which demands require teams of vehicles for their service. We are also interested in extending our results to the case of non-uniform demand densities (possibly class dependent), and to impatient demands that disappear if they are not serviced within a certain time window.

REFERENCES

[1] 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, 18 (2002), pp. 911–922.

[2] D. Bertsimas, I. C. Paschalidis, and J. N. Tsitsiklis, *Optimization of multiclass queueing networks: Polyhedral and nonlinear characterizations of achievable performance*, The Annals of Applied Probability, 4 (1994), pp. 43–75.

[3] D. J. Bertsimas and G. J. van Ryzin, *A stochastic and dynamic vehicle routing problem in the Euclidean plane*, Operations Research, 39 (1991), pp. 601–615.

[4] ———, *Stochastic and dynamic vehicle routing in the Euclidean plane with multiple capacitated vehicles*, Operations Research, 41 (1993), pp. 60–76.

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

[6] 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, (2009). Submitted.

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

[8] E. Frazzoli and F. Bullo, *Decentralized algorithms for vehicle routing in a stochastic time-varying environment*, in IEEE Conf. on Decision and Control, Paradise Island, Bahamas, Dec. 2004, pp. 3357–3363.

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

[10] A. Larsen, *The Dynamic Vehicle Routing Problem*, PhD thesis, Technical University of Denmark, Lyngby, Denmark, 2000.

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

[12] D. G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley, 2 ed., 1984.

[13] J. D. Papastavrou, *A stochastic and dynamic routing policy using branching processes with state depended immigration*, European Journal of Operational Research, 95 (1996), pp. 167–177.

[14] 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, (2008). To appear.

[15] M. Pavone, E. Frazzoli, and F. Bullo, *Distributed and adaptive algorithms for vehicle routing in a stochastic and dynamic environment*, IEEE Transactions on Automatic Control, (2009). Submitted.

[16] M. Pavone, S. L. Smith, F. Bullo, and E. Frazzoli, *Dynamic multi-vehicle routing with multiple classes of demands*, in American Control Conference, St. Louis, MO, June 2009. To appear.

[17] G. Percus and O. C. Martin, *Finite size and dimensional dependence of the Euclidean traveling salesman problem*, Physical Review Letters, 76 (1996), pp. 1188–1191.

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

[19] S. L. Smith and F. Bullo, *The dynamic team forming problem: Throughput and delay for unbiased policies*, Systems & Control Letters, (2008). Submitted.

[20] S. L. Smith, M. Pavone, F. Bullo, and E. Frazzoli, *Dynamic vehicle routing with heterogeneous demands*, in IEEE Conf. on Decision and Control, Cancún, México, Dec. 2008, pp. 1206–1211.

[21] M. Z. Spivey and W. B. Powell, *The dynamic assignment problem*, Transportation Science, 38 (2004), pp. 399–419.

[22] J. M. Steele, *Probabilistic and worst case analyses of classical problems of combinatorial optimization in Euclidean space*, Mathematics of Operations Research, 15 (1990), p. 749.

[23] H. A. Waisanen, D. Shah, and M. A. Dahleh, *A dynamic pickup and delivery problem in mobile networks under information constraints*, IEEE Transactions on Automatic Control, 53 (2008).

[24] R. W. Wolff, *Stochastic modeling and the theory of queues*, Prentice Hall, 1989.

[25] H. Xu, *Optimal policies for stochastic and dynamic vehicle routing problems*, Dept. of Civil and Environmental Engineering, Massachusetts Institute of Technology, Cambridge, MA, 1995.