Optimizing Task Waiting Times in Dynamic Vehicle Routing

Alexander Botros ⓘ, *Member, IEEE,* Barry Gilhuly ⓘ, *Member, IEEE,* Nils Wilde ⓘ, *Member, IEEE,*
Armin Sadeghi ⓘ, *Member, IEEE,* Javier Alonso-Mora ⓘ, *Senior Member, IEEE,*
and Stephen L. Smith ⓘ, *Senior Member, IEEE*

Abstract—We study the problem of deploying a fleet of mobile robots to service tasks that arrive stochastically over time and at random locations in an environment. This is known as the Dynamic Vehicle Routing Problem (DVRP) and requires robots to allocate incoming tasks among themselves and find an optimal sequence for each robot. State-of-the-art approaches only consider average wait times and focus on high-load scenarios where the arrival rate of tasks approaches the limit of what can be handled by the robots while keeping the queue of unserviced tasks bounded, i.e., stable. To ensure stability, these approaches repeatedly compute minimum distance tours over a set of newly arrived tasks. This letter is aimed at addressing the missing policies for moderate-load scenarios, where quality of service can be improved by prioritizing long-waiting tasks. We introduce a novel DVRP policy based on a cost function that takes the p-norm over accumulated wait times and show it guarantees stability even in high-load scenarios. We demonstrate that the proposed policy outperforms the state-of-the-art in both mean and 95th percentile wait times in moderate-load scenarios through simulation experiments in the Euclidean plane as well as using real-world data for city scale service requests.

Index Terms—Path planning for multiple mobile robots or agents, planning, scheduling and coordination, task planning.

I. INTRODUCTION

THE Dynamic Vehicle Routing Problem is an important and long-studied challenge in assigning autonomous vehicles (or robots) to tasks as they appear in the environment. Applications include pickup-and-delivery [1], [2], mobility-on-demand transportation systems [3], [4], [5], sensor networks [6], [7], surveillance [8], [9], personal care [10], [11], and environmental monitoring [12], [13], [14].

In a typical application, one or more robots assign and schedule incoming tasks to optimize their quality of service. When the set of tasks is finite and known a priori, the resulting problem is known as *Vehicle Routing:* a fleet of m robots services n tasks while maximizing some service measure. When new tasks arrive over time, the problem becomes the *dynamic vehicle routing problem* (DVRP). The conventional approach for multiple-robot vehicle routing with provable properties [6], [7] is to divide the environment into equitable partitions. A robot is assigned to each partition thus simplifying the multiple-robot problem into several single-robot instances. In this letter we seek to improve multi-robot performance by making advances in the underlying single-robot policies.

A characteristic of DVRPs is the load factor $\rho \in (0, 1)$ [15], [16], [17]. Letting λ denote the arrival rate of incoming tasks and letting $\bar{s}$ be the expected service time, the load factor is given by $\rho = \lambda\bar{s}/m$ for m robots. It captures the fraction of time the robot fleet *must* be working to service tasks. In light load $\rho \to 0^+$, the queue length of unserviced tasks approaches zero, and robots have enough time between task arrivals to move to optimal waiting locations [6]. In heavy load $\rho \to 1^-$, all robots are continuously servicing tasks with no idle time, and the queue length of unserviced tasks approaches infinity. Neither of these states is operationally desirable: under light load, robots are underutilized, and under heavy load, task wait times become undesirably long [15]. Previous work focused on guarantees under the extremes of light or heavy load, neglecting the space where these systems operate ideally [18].

In this letter, we focus on systems operating under moderate load, where for example $\rho \in [.5, .9]$. Moderate load is arguably of the most practical significance, since it corresponds to the case where the queue of unserviced tasks is neither empty nor approaching infinite length. In fact, as a rule of thumb, the load factor of a robot should be below 0.9 [18] since the length of the outstanding task queue is proportional to $1/(1 - \rho)^2$ [19]. We propose a policy for moderate loads that seeks to solve the multi-objective problem of minimizing both average and maximum wait times. To that end, we study a cost function to compute new tours where we minimize the sum of the wait times, each raised to an exponent $p > 1$. Under this cost function, tasks gain priority by waiting sufficiently long. While there is some advantage to this cost function under heavy load ($\rho \to 1^-$), the advantage disappears as the queue grows and travel distances between tasks approach zero. We illustrate the advantages of our method in Fig. 1 for a monitoring application such as forest fire detection [20]. A network of low-cost sensors (e.g., distributed in the environment, onboard high-altitude drones, or satellites) can detect points of interest. Upon detection, a drone with high-resolution sensors is tasked to visit these locations to collect more accurate data. State-of-the-art methods [7] find tours of minimal length to minimize total mission time, shown in Fig. 1(a)). In contrast,

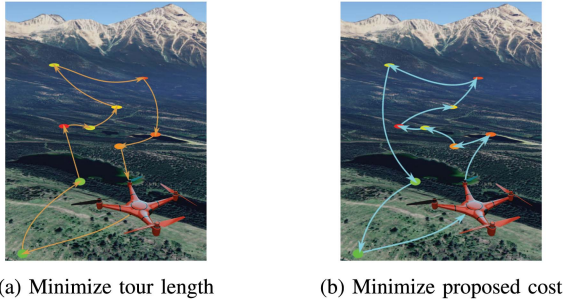(a) Minimize tour length   (b) Minimize proposed cost

Fig. 1. Effect of different cost functions on the optimal tour of a monitoring UAV. Triggered ground sensors add observation tasks (colored dots) for the UAV. Color (increasing values from green to red) represents how long a task has already been waiting when the tour is planned (latent wait time).

our method, shown in Fig. 1(b)), gives a higher priority to tasks that have waited longer and thus visits these locations earlier. Tasks with longer waiting times (red) are serviced earlier in the tour using our method, resulting in a lower mean service time for all tasks. This results in a lower average and maximum wait time.

Related Work: Vehicle Routing problems have been studied in many forms over the past several decades [15], [21], [22], with active research continuing today [23], [24], [25], [26], [27], [28], [29], [30]. Following the taxonomy in [27], we consider the *dynamic and stochastic* variant of DVRP where new tasks arrive online (dynamic) according to a stochastic process (stochastic) and are randomly distributed in the environment.

The categorization of light and heavy load states was considered in [15]. The bounds on the performance of the optimal policy in heavy load given in [15], [19] are a building block for a number of studies on DVRP variants [6], [7], [16]. In [7], the authors propose a BATCH policy where the TSP is calculated on the task locations of a batch of tasks, and at each iteration, a randomly chosen fraction of the batch is serviced. The authors show that this approach is within a factor of two of optimal in heavy load. The aforementioned studies focused on heavy or light loads and proposed policies that compute tours which minimize the total travel time (i.e., TSP minimization). An alternative policy is to minimize the maximum wait time and/or the average wait time [31], [32], [33] in the context of fairness among tasks. Closely related to our approach, [34] uses a vehicle routing cost based on the square of wait times; while the cost function is a special case of our cost function where the wait time is raised to some exponent $p > 1$, the policy is different. In [34] routes are recomputed whenever new tasks arrive, while our work uses the BATCH approach from [7]. This allows us to derive theoretical guarantees on system stability which are not provided in [31], [32], [33], [34].

The DVRP finds widespread applications in various robotics and transportation domains. Fleets of autonomous vehicles are deployed for pickup-and-delivery throughout cities [1], [2] or for providing on-site service in indoor environments such as hospitals [10], [11]. In many cases, vehicles need to react to new requests appearing over time. For instance, in autonomous mobility on demand [3], [4], [5] it is desired to minimize the average response time to the incoming requests, while also limiting maximum waiting times. Other applications include the deployment of autonomous drones as mobile sensor networks [6], [7], for surveillance [8], [9], search and rescue [35], and in environmental monitoring [12], [13], [14].

Contributions: Our contributions are as follows. First, we propose a new BATCH policy using the sum of wait times, each raised to an exponent p, as the cost. This actively considers the time a task has already waited and thus allows for prioritizing long-waiting tasks. Second, we establish a relation between the length of the optimal tour for the proposed cost and the number of tasks in the tour, and prove that for $p > 1$ the proposed policy is stable under any load factor $\rho < 1$, providing the first stability result for a policy with this type of cost function. Finally, we demonstrate that under moderate load our policy achieves a lower average wait-time and reduces the number of outlying tasks with high wait-time.

II. PROBLEM FORMULATION

In this section, we revisit the formulation of the well-known DVRP. Consider a convex and closed environment $\mathcal{E} \subset \mathbb{R}^d$ where d is the dimension. A set of m vehicles traveling with constant speed v are servicing the tasks arriving in $\mathcal{E}$. The tasks arrive according to a Poisson process with time-intensity $\lambda \in \mathbb{R}_{>0}$. Task locations are distributed according to a spatial density $\varphi : \mathcal{E} \to \mathbb{R}_{>0}$. Task arrivals are i.i.d (independent and identically distributed), and servicing a task requires a vehicle to visit the location of the task. The time required to service task j at its location, denoted by s_j, is i.i.d with finite first and second moments, i.e., $\overline{s}$ and $\overline{s^2}$. The *load factor* for this system of m vehicles is defined as $\rho = \lambda \overline{s}/m$. Given the arrived tasks and the current vehicle locations, a policy repeatedly computes tours for each vehicle. The wait time of task j, denoted by W_j, is the time between the arrival time of the task and the time that a vehicle arrives at the task location and starts servicing it. The system time of task j, denoted by T_j, is the time between the arrival of the task and the time that a vehicle completes servicing the task, i.e., $T_j = W_j + s_j$. The steady-state system time is denoted by $\overline{T} = \limsup_{j \to \infty} \mathbb{E}[T_j]$. Let π denote a tour of tasks found by following a routing policy, and let $\overline{T}_\pi$ be the corresponding system time. A policy is *stable* if the expected number of outstanding tasks is uniformly bounded at all times. Finally, we assume that vehicles have sufficient capacity for all tasks and do not need to return to the depot.

Problem 1 (DVRP): Given an environment $\mathcal{E}$, m vehicles, and tasks arriving according to a stochastic process with density λ, spatial distribution φ, and random service times following a Gaussian distribution $\mathcal{N}(\overline{s}, \sigma)$, find a stable policy that computes a tour π minimizing the system time, i.e., $\inf_\pi \overline{T}_\pi$.

III. APPROACH

We adapt a state-of-the-art partitioning approach [7] to cast the multi-robot setting of the DVRP into a set of single-robot instances. After reviewing previously proposed DVRP policies with provable stability guarantees, we present a novel single vehicle policy based on a cost function where we take the p-norm of wait times and show that this policy is also stable.

A. Vehicle Routing Cost Function

We study how a single vehicle computes routes to service tasks in the environment $\mathcal{E} \subset \mathbb{R}^d$. At a start time t_0, let $\mathcal{T}$ be set of n tasks that have arrived but not been serviced. Further, let x_s be the vehicle's starting pose, let $x_i, i = 1, \ldots, n$ denote the pose of task $\tau_i \in \mathcal{T}$, and let $L_{i,j} = ||x_i - x_j||/v$ denote the travel

---

Algorithm 1: Generalized Policy.

1: **procedure** Tour $(\mathcal{T}, p \geq 1, \eta \in (0,1], p_t \in \{0,1\},$
       $R \in \{0,1\})$
2:    Replace t_i with $p_t t_i$ in c^p (1)
3:    **while** $\mathcal{T} \neq \emptyset$ **do**
4:       Compute a c^p-minimizing tour π^* of $\mathcal{T}$
5:       $(x_s, x_1, x_2, \ldots, x_n) \leftarrow \pi^*$
6:       index $\leftarrow (\text{RandomInteger}(1,n))^R$
7:       Service tasks $\tau_{\text{index}}, \ldots, \tau_{\text{index}+\eta n}$ in order
8:       Remove tasks $\tau_{\text{index}}, \ldots, \tau_{\text{index}+\eta n}$ from $\mathcal{T}$
9:       **if** a new task τ_{n+1} arrives at any time **then**
10:         add τ_{n+1} to $\mathcal{T}$
11:      return to x_{waiting}

---

time between x_i, x_j for all $i, j \in \{1, \ldots, n\} \cup \{0\}$. We assume constant speed v, which, without loss of generality, we set to 1 implying that length and travel time for any segment are equivalent. For each task τ_i, let $t_i \geq 0$ denote the *latent wait time* of task $\tau_i \in \mathcal{T}$: the difference between the time of planning the current tour and when τ_i arrived. Finally, let s_i denote the time required to service task τ_i. Thus, we decompose the system time, T_i, into three components: $T_i = t_i + \{tour\ travel\ time\ to\ \tau_i\}_i + s_i$.

Using these definitions, a tour π is an ordering of task service visits that minimizes some undesirable properties. Given an arbitrary tour $\pi = (x_s, \tau_1, \tau_2, \ldots, \tau_n)$ as well as a value $p \in \mathbb{N}_{\geq 1}$, we consider a cost function:

$$c^p(\pi) = \left(\sum_{i=1}^{n} \left(\overbrace{t_i}^{\text{latent wait time of task } \tau_i} + \underbrace{\sum_{j=1}^{i} \left(\frac{L_{j,j-1}}{v} + \bar{s} \right)}_{\text{time to reach and service task } \tau_i} \right)^p \right)^{1/p} . \quad (1)$$

We notice that c^p represents the L^p-norm of the waiting time of each task in $\mathcal{T}$ assuming service times s_i equal the expected value $\bar{s}$. We assume that the service time of any task is not revealed before servicing [6]. Thus, treating service time as $\bar{s}$ when evaluating a candidate tour is not unreasonable. Since $\bar{s}$ is constant, in the remainder of this letter, we denote $L_{j,j-1}/v + \bar{s}$ as simply $l_{j,j-1}$ for brevity. We offer an observation:

Observation 1 (Variants of c^p cost): The cost $c^p(\pi)$ generalizes previously proposed cost functions:
1) Minimizing $c^1(\pi)$ is equivalent to minimizing the mean wait time of all tasks in $\mathcal{T}$. (1) reduces to the sum of wait times and long waits are no longer penalized.
2) Minimizing $c^\infty(\pi)$ is equivalent to minimizing the maximum wait time of all tasks in $\mathcal{T}$ since the longest wait time becomes the dominant term and is heavily penalized. If $t_i = 0$ holds (i.e., all tasks arrive at the beginning of the tour), then the problem is equivalent to minimizing the total travel distance i.e., solving the *classic* TSP.

B. Solution Archetypes and Proposed Approach

In this section, we present three state-of-the-art approaches to Problem 1 as well as our proposed method. We begin with a generalized algorithm (Algorithm 1) that, with appropriately selected inputs, captures all four approaches. The algorithm Tour takes as input a set of n tasks $\mathcal{T}$, an exponent p for

the c^p cost defined in (1), a value $\eta \in (0,1]$, and two boolean variables p_t and R. The high-level idea behind the algorithm is this: we start by computing a tour π^* that minimizes c^p over all tours of $\mathcal{T}$ (Line 4) and then select a fragment of length ηn — which we refer to as an η-*fragment* — of this tour to service (Lines 6–8). The boolean input R controls which fragment to service. If $R = 0$ then index$= 1$ (Line 6) and we service the first η-fragment, whereas if $R = 1$ then we service a random η-fragment (Lines 7–8). If a new task τ_{n+1} arrives during this time, it is added to $\mathcal{T}$ (Lines 9 - 10). Once the η-fragment has been serviced, the process repeats. Finally, the boolean variable p_t controls whether the latent wait time is considered in the cost c^p. If there are no outstanding tasks (Line 11), the robot returns to a given waiting pose x_{waiting} usually defined as the centroid of the workspace [6], [7].

BATCH policy: The BATCH policy proposed in [6] has the structure TOUR($\mathcal{T}, p = \infty, \eta = 1, p_t = 0, R = 0$). Since $p = \infty$, and $p_t = 0$, the cost c^p is the total length of the tour, and c^p is minimized by a tour π^{TSP} that solves the TSP. Further, since $\eta = 1, R = 0$, the full set of tasks $\mathcal{T}$ is serviced before re-planning.

η- BATCH policy: In contrast, the authors of [7] proposed η-BATCH, which we capture with TOUR($\mathcal{T}, p = \infty, \eta \in (0,1), p_t = 0, R = 1$). Again, the tour π^* in Line 4 of Algorithm 1 is π^{TSP} that solves the TSP for tasks $\mathcal{T}$. However, since $\eta \in (0,1)$, a η-fragment is randomly selected from π^* and serviced before re-planning.

The notion of a η-fragment is proposed in [7] to improve the average wait time of tasks in $\mathcal{T}$. If $\eta = 1$, then new tasks wait for all existing tasks to be serviced before they are considered. However, if $\eta \in (0,1)$ and the first η-fragment is serviced before re-planning, then it is possible for some tasks to be continually overlooked. Thus, the η-BATCH policy arbitrarily selects an η-fragment to service before re-planning ($R = 1$) which ensures that, in expectation, no task must wait longer than $1/\eta$ iterations of Algorithm 1 before servicing.

DC - BATCH policy: The single vehicle Divide and Conquer strategy [7] partitions the environment into r equitable sectors. The algorithm plans a TSP tour for the tasks that arrive in a sector, services them, and then moves to the next sector with tasks in a round-robin fashion.

Proposed c^p- BATCH policy: We propose a new policy labeled c^p-BATCH which has the form TOUR($\mathcal{T}, p \in (1,\infty), \eta \in (0,1), p_t = 1, R = 0$). That is, we include t_i in the cost (1) and thus do not find a tour π^* of minimal length, but minimal c^p-cost. Further, we do not select the η-fragment randomly.

The motivation for our approach is that values $p \in (1,\infty)$ and $p_t = 1$ incorporate the latent wait time of tasks into the cost. If tasks have been waiting too long to be serviced, it will become too costly not to service them. Thus we adopt the η-fragment strategy of [7] to improve performance without having to also impose randomness to mitigate long maximum wait times. The value $p = 1$ has been excluded since it places no additional penalty on long waits, leading to the possibility of distant tasks being ignored [21]. Examples of the BATCH, $\eta - \text{BATCH}$, DC-BATCH, and proposed approaches are illustrated in Fig. 2. Observe that $\eta - \text{BATCH}$ services τ_5 first: the randomness of the η-fragment causes this approach to miss τ_1 which was closer to the start location. This does not occur in the proposed approach which services the first η-fragment before re-planning. The stability of the proposed method is established next.
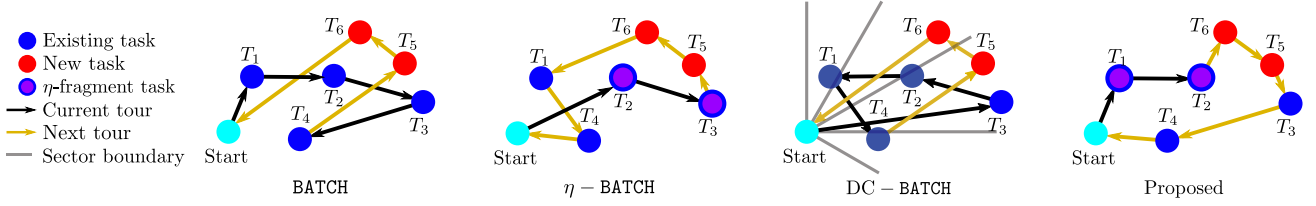
Fig. 2. Example solutions for the different archetypes on the same tasks. Here, $x_s = \texttt{Start} = x_{\text{waiting}}$. Tasks $\tau_1 - \tau_4$ arrived before planning the current tour, tasks τ_5, τ_6 arrive immediately after the tour commences execution.

IV. STABILITY ANALYSIS

In this section, we prove the stability of the proposed approach. We use the following short-hand notation. Given a tour $\pi = (\tau_0, \ldots, \tau_n)$ of a set of tasks $\mathcal{T}$, we use l_j to represent the euclidean distance between tasks $\tau_{j-1}, \tau_j \in \mathcal{T}$. That is, $l_j = l_{j,j-1}$. Further, $l(\pi)$ denotes the total length of π. We start with assumptions:

Assumption 1: Both $\bar{s}$ and λ are known and the expected load factor is $\bar{\rho} = \lambda \bar{s} < 1$.

Assumption 2: All tasks arrive in a connected planar region with area A, finite perimeter P, and maximum Euclidean distance between tasks Q'.

Assumption 3: At iteration 0 of Algorithm 1, the number of unserviced tasks is finite and latent wait times t_i are bounded by an arbitrarily large constant R (i.e., $R = \max_{t_i \in \mathcal{T}} t_i$).

Assumptions 1 and 2 are common in the literature (e.g. [6], [7]). Further, Assumption 1 does not include the spatial distribution of tasks φ and the variance of service times, $\overline{s^2}$ since the stability result does not depend on them. Assumption 3 is reasonable since it states only existing tasks at the start have not been waiting for unbounded time. Letting $Q = Q' + \bar{s}$, we note that Q represents the maximum time to service a task and move to the next under expected service times.

The proof of stability of the c^p-BATCH policy follows closely the proofs for the policies proposed in [6], [7] in that it appeals to a well-established recursion. Letting N_k, T_k be the random variables representing the number of outstanding tasks at the beginning of iteration k of Algorithm 1, and the total service time of those tasks, respectively, and letting λ, η be constants representing the arrival rate and fragment length (used as input to Algorithm 1), respectively, then from [7]:

$$\mathbb{E}[N_{k+1}] = \underbrace{\lambda \eta \mathbb{E}[T_k]}_{\text{Newly arrived tasks}} + \underbrace{(1 - \eta)\mathbb{E}[N_k]}_{\text{left-over tasks from iteration } k} , \quad (2)$$

The authors of [7] observe that at unit speed, $\mathbb{E}[T_k] \leq Q + \eta \mathbb{E}[l(\pi_k^*)] + \eta \mathbb{E}[N_k]\bar{s}$ and (2) reduces to

$$\mathbb{E}[N_{k+1}] \leq \lambda Q + \lambda \eta \mathbb{E}[l(\pi_k^*)] + \rho \eta \mathbb{E}[N_k]$$
$$+ (1 - \eta)\mathbb{E}[N_k]. \quad (3)$$

At a high and informal level, (3) implies that if $\mathbb{E}[l(\pi_k^*)]$ grows with *less than linear* (LL) power in $\mathbb{E}[N_k]$, and $\rho < 1$, then

$$\lim_{\mathbb{E}[N_k] \to \infty} \left(\frac{\mathbb{E}[N_{k+1}]}{\mathbb{E}[N_k]} \right) \leq 1 - \eta(1 - \rho) < 1. \quad (4)$$

implying (at a high level) the stability of the policy. In [6], [7], the authors make use of the well established Beardwood-Halton-Hammersley (BHH) result [36] to prove that if $\pi_k^* = \pi^{\text{TSP}}$, then $\mathbb{E}[l(\pi_k^*)] \leq K \sqrt{\mathbb{E}[N_k]}$ for a constant K from which stability

follows. Though the stability result in [7] is far more rigorous than the argument above, the crux of that argument, and ours, is the LL power of $\mathbb{E}[l(\pi_k^*)]$ in $\mathbb{E}[N_k]$. Unfortunately, the BHH result cannot be applied directly to a c^p-minimizing tour π_k^* for general values of p, since the latent wait time $t_i, \tau_i \in \mathcal{T}$ cannot be separated from the cost. Therefore, unlike the tours π^{TSP} used in [6], [7], the tours π_k^* will depend on these latent wait times. As such, to follow the proofs of stability in [7], we must establish the LL power of $\mathbb{E}[l(\pi_k^*)]$ in $\mathbb{E}[N_k]$ at each iteration k. We begin with supporting results.

Lemma 1 (Bounding Tour Length By Cost): Given any n tasks $\mathcal{T}$ and any tour π on those tasks, if $p \geq 1$, then the length of π obeys the following inequality:

$$l(\pi) \leq (Q(p+1)c^p(\pi)^p)^{\frac{1}{p+1}} \quad (5)$$

Proof: Given tour π on tasks $\mathcal{T}$, we observe trivially that $c^p(\pi)^p \geq \sum_{i=1}^n (\sum_{j=1}^i l_i)^p$. Therefore, the length of any tour $l(\pi)$ is no more than the length of the maximum length tour $l_{\max}(\mathcal{T})$ subject to this constraint:

$$l(\pi) \leq l_{\max}(\mathcal{T}) \doteq \max_{l_1, \ldots, l_n} l_1 + l_2 + \cdots + l_n$$

$$s.t. \sum_{i=1}^n \left(\sum_{j=1}^i l_i \right)^p \leq c^p(\pi)^p, \ l_i \leq Q.$$

It is not difficult to show that there exists a value k such that $l_{\max}(\mathcal{T})$ has the form: $l_1 = \cdots = l_{k-1} = 0$, and $l_k = \cdots = l_n = Q$. That is, that the first k lengths are 0 while the remainder take their maximum value. This is because increasing any value $l_i, i < k$ by δ would require a decrease of a value $l_j, j \geq k$ by more than δ in order to satisfy the constraint. To determine k, we replace the above general form of $l_{\max}(\mathcal{T})$ in the first constraint and simplify, resulting in $Q^p \sum_{i=1}^{n-k+1} i^p \leq c^p(\pi)^p$. Therefore, letting $L = n - k$, and observing that $\sum_{i=1}^{L+1} i^p \geq (L+1)^{p+1}(p+1)^{-1}$ for all $L \geq 0$ and $p \geq 1$, it must hold that

$$Q^p(L+1)^{p+1}(p+1)^{-1} \leq c^p(\pi)^p$$

$$\Rightarrow L \leq \left(\frac{c^p(\pi)^p(p+1)}{Q^p} \right)^{1/p+1}.$$

Finally, given the general form of $l_{\max}(\mathcal{T})$ described above, we observe that $l_{\max}(\mathcal{T}) = LQ$ and the result follows. $\square$

We now establish a bound on the optimal tour length on n tasks $\mathcal{T}$. We let $t_{\max} \geq \max_{\tau_i \in \mathcal{T}} t_i$ be the maximum latent wait time of tasks in $\mathcal{T}$. We assume that $t_{\max}$ is bounded by a linear function of the number of tasks.

Lemma 2 (Deterministic bound: $l(\pi^*)$): For any n tasks $\mathcal{T}$ and $p \geq 1$, if there exists constants $c_t \geq 0, \nu \in [0,1)$ such that

$t_{\max} \leq c_t n^\nu$ and $n \geq P^2/2\,A$, then

$$l(\pi^*) \leq \underbrace{\left(2^{p-1} Q(p+1)(c_t^p + 2^p \sqrt{2\,A}^p)\right)^{\frac{1}{p+1}}}_{\text{a constant}} n^{\frac{p\gamma+1}{p+1}}, \quad (6)$$

where $\gamma = \max\{0.5, \nu\}$.

Proof: We begin by establishing an upper bound on $c^p(\pi^{\text{TSP}})^p$. Let $\tau_1, \tau_2, \ldots, \tau_n$ denote the ordering of the tasks $\mathcal{T}$ as they appear in π^{TSP}. Then, by the definition of the cost $c^p(\pi^{\text{TSP}})$, it must hold that

$$c^p(\pi^{\text{TSP}})^p = \sum_{i=1}^n \left(t_i + \sum_{j=1}^i l_j\right)^p \leq n\left(t_{\max} + \sum_{j=1}^n l_j\right)^p.$$

Noting $\sum_{j=1}^n l_j = l(\pi^{\text{TSP}})$ and $l(\pi^{\text{TSP}})$ is bounded [37] by $\sqrt{2\,A}\sqrt{n} + P$ for all n. Therefore,

$$c^p(\pi^{\text{TSP}})^p \leq n\left(t_{\max} + \sqrt{2An} + P\right)^p$$

$$\leq n\left(c_t n^\nu + 2\sqrt{2An}\right)^p, \text{ since } t_{\max} \leq c_t n^\nu, n \geq P^2/2\,A$$

$$\leq 2^{p-1}(c_t^p + 2^p \sqrt{2\,A}^p) n^{p\gamma+1}. \quad (7)$$

The final inequality holds since $(A+B)^p \leq 2^{p-1}(A^p + B^p)$ for all $A, B \geq 0, p \geq 1$ by the convexity of $x^p, x \geq 0$. Since π^* minimizes c^p over all tours of $\mathcal{T}$, we conclude that $c^p(\pi^*)^p \leq c^p(\pi^{\text{TSP}})^p$ and the result follows from (7) and Lemma 1. □

Let n_k denote the number of unserviced tasks at the beginning of iteration k of Algorithm 1. Thus, n_k is the realization of the random variable N_k. For the remainder of our analysis we consider $n_k \geq P^2/2\,A$ for constants P, A. If there is no iteration k such that $n_r \geq P^2/2\,A$ for every $r \geq k$, then stability holds trivially. Lemma 2 implies that if the latent wait times of tasks at iteration k of Algorithm 1 are all bounded by a function of n_k with LL power then the same holds for the length of tour (since $p\gamma+1/p+1 < 1$). Using this result, we offer a deterministic bound on the length of the optimal tour at any iteration when $\eta = 1$, and extend this result to $\eta \in (0, 1]$.

Theorem 1: On any iteration k of Algorithm 1, if π_k^* is the c^p-minimizing tour computed in Line 4, $p \geq 1$, $n_k \geq P^2/2\,A$, and $\eta = 1$ then there exists constants $\beta \geq 0, \kappa \in [0, 1)$ with

$$l(\pi_k^*) \leq \beta n_k^\kappa. \quad (8)$$

Proof: We offer a proof by way of induction on the number of iterations k of Algorithm 1. Let $\mathcal{T}_k$ denote the outstanding tasks at the start of iteration k, and π_k^* the tour of $\mathcal{T}_k$ from Line 4. In the base case, $k = 1$ and by Assumption 3, $t_{\max} \leq R$. Therefore, $t_{\max} \leq c_t n^\nu$ with $c_t = R, \nu = 0$, and the result holds in the base case by Lemma 2, since $\gamma = 0.5$ and $p+2/2(p+1) < 1$ for $p \geq 1$.

The induction assumption is that at iteration k, $l(\pi_k^*) \leq \beta n_k^\kappa, \beta \geq 0, \kappa \in [0, 1)$. Since $\eta = 1$, the maximum latent wait time of any task in $\mathcal{T}_{k+1}$ is $l(\pi_k^*)$ which occurs if a task arrives just after the beginning of iteration k. If $n_{k+1} < n_k$, then the result holds by the induction assumption. Otherwise, $n_{k+1} \geq n_k$ and by the induction assumption $t_{\max} \leq l(\pi_k^*) \leq \beta n_k^\kappa \leq \beta n_{k+1}^\kappa$. Thus the conditions of Lemma 2 hold with $c_t = \beta, \nu = \kappa$, and by (6), $l(\pi_{k+1}^*) \leq \bar{C} n_{k+1}^{p\gamma+1/p+1}$ where $\bar{C}$ is the constant coefficient in (6) with $\gamma = \max\{1/2, \kappa\} < 1$. Since $\gamma < 1$, it must hold that

$p\gamma+1/p+1 < 1$ and the result holds at iteration $k + 1$ concluding the proof. □

Theorem 2: The result of Theorem 1 still holds if $\eta \in (0, 1], p \geq 1$ for sufficiently large n_k.

Proof: We begin by proving a sub-claim: given a set of n tasks $\mathcal{T}_n$, and a new task $\tau_{n+1} \notin \mathcal{T}_n$, let $\mathcal{T}_{n+1} = \mathcal{T}_n \cup \{\tau_{n+1}\}$ and π_n^*, π_{n+1}^* denote c^p-minimizing tours of $\mathcal{T}_n, \mathcal{T}_{n+1}$, respectively. If n is sufficiently large and there exists constants $c_l, c_u \geq 0, \nu \in [0, 1)$ such that $c_l \sqrt{n} \leq t_i \leq c_u n^\nu$ for all $\tau_i \in \mathcal{T}_n$, then for any $\eta \in (0, 1], \tau_{n+1}$ will not appear in the first η-fragment of π_{n+1}^*. To prove this sub-claim, we show that any tour π_{n+1}' of $\mathcal{T}_{n+1}$ with τ_{n+1} in the first η-fragment cannot be optimal by proposing a lower cost tour π_{n+1}. Let

$$\pi_n^* = (\tau_1, \tau_2, \ldots, \tau_n)$$

$$\pi_{n+1} = (\tau_1, \tau_2, \ldots, \tau_n, \tau_{n+1})$$

$$\pi_{n+1}' = (\tau_{i_1}, \ldots, \tau_{i_r}, \tau_{n+1}, \tau_{i_{r+1}}, \ldots, \tau_{i_n})$$

$$\pi_n' = (\tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_n}).$$

Here, π_n^* denotes the tour that minimizes c^p for tasks $\mathcal{T}_n$, π_{n+1} is the tour of tasks $\mathcal{T}_{n+1}$ that services tasks according to π_n^* and then services τ_{n+1}, π_{n+1}' is any tour of tasks $\mathcal{T}_{n+1}$ with τ_{n+1} inserted after $r \leq \eta n$ tasks, and π_n' is the tour that is identical to π_{n+1}' but with task τ_{n+1} removed. To prove the sub-claim, we will show that for large n, $c^p(\pi_{n+1}) < c^p(\pi_{n+1}')$. To this end, let $w_i(\pi)$ denote the wait time of task τ_i in the tour π, and let Δ the additional wait time of all tasks $\tau_j, j \geq i_{r+1}$ incurred from servicing τ_{n+1} first in π_{n+1}'. If $\Delta = 0$ the claim is redundant to the proof as new tasks can be serviced instantaneously without increasing the wait time of any other task. Otherwise, $\Delta > 0$, and

$$c^p(\pi_{n+1}')^p \geq \sum_{j=i_1}^{i_r} w_j^p + \sum_{j=i_{r+1}}^{i_n} (w_i + \Delta)^p$$

$$\geq \sum_{j=i_1}^{i_n} w_j^p + p\Delta \sum_{j=i_{r+1}}^{i_n} w_i^{p-1} \geq c^p(\pi_n^*)^p + p\Delta(n-r)n^{\frac{p-1}{2}}.$$

The above uses the fact that $(A+B)^p \geq A^p + pBA^{p-1}$ for all $A, B \geq 0, p \geq 1$ and $w_i \geq t_i \geq \sqrt{n}$. Further, observe that $c^p(\pi_{n+1})^p \leq c^p(\pi_n^*)^p + (l(\pi_n^*) + Q)^p$. Therefore, since $r \leq \eta n$, and $t_i \leq c_u n^\nu$, it follows by Lemma 2 that

$$c^p(\pi_{n+1}')^p - c^p(\pi_{n+1}')^p \geq p\Delta(1-\eta)n^{\frac{p+1}{2}} - (l(\pi_n^*) + Q)^p$$

$$\geq p\Delta(1-\eta)n^{\frac{p+1}{2}} - (\bar{C}n^{\frac{p\nu+1}{2(p+1)}} + Q)^p.$$

Finally, noting that $p+1/2 > p(p\nu+1)/2(p+1)$, we have that $c^p(\pi_{n+1}')^p - c^p(\pi_{n+1}')^p \geq 0$ for large n establishing the sub-claim. To prove the result of the Theorem, it suffices to show that at every iteration of Algorithm 1, $t_{\max} \leq c_t n_k^\nu$ for some $c_t \geq 0, \nu \in [0, 1)$ since the result of the Theorem would follow in an identical manner to the proof of Theorem 1. The proof that $t_{\max} \leq c_t n_k^\nu$ at every iteration k is largely omitted for brevity, but we describe the structure. By strong induction on the number of iterations, suppose that n tasks $\mathcal{T}^{\text{old}}$ at iteration k arrived before the start of iteration $k - 1$ but were not serviced in that iteration. Then their wait time is at least $\eta l(\pi_{k-1}^*) \geq \eta l(\pi_k^{\text{TSP}}) \geq \eta\sqrt{n}$. Further, by the strong induction assumption, $t_{\max} \leq c_t n^\nu$. Therefore, the conditions of the sub-claim hold implying that the first η-fragment is comprised entirely of these

tasks. This continues until at most $1/\eta$ iterations when all the tasks in $\mathcal{T}^{\text{old}}$ are serviced. Indeed, in the worst case, if no tasks in $\mathcal{T}^{\text{old}}$ are serviced in $1/\eta$ iterations, we can show using the above analysis that $\mathcal{T}^{\text{old}}$ occupies the first $n = \eta(n/\eta)$-fragment. On each iteration i before the tasks in $\mathcal{T}^{\text{old}}$ have all been serviced, they remain the oldest tasks in $\mathcal{T}_i$, and can be shown to have a maximum latent waiting time $t_{\max} \leq \beta n^\nu / \eta^\nu$ using a similar argument as Theorem 1. $\qquad\square$

Finally, we prove the stability of the proposed policy.

Theorem 3 (Stability of Proposed Algorithm): For all load factors $\rho < 1$, the proposed policy $c^p - \texttt{BATCH}$ with $p \geq 1, \eta > 0$ is stable.

Proof: By Theorem 2, for sufficiently large n_k, it holds that $l(\pi_k^*) \leq \beta n_k^\kappa$ for constants $\beta \geq 0, \kappa \in [0, 1)$. Therefore, $\mathbb{E}[l(\pi_k^*)] \leq \beta \mathbb{E}[N_k]^\kappa$ by Jensen's inequality. The remainder of the proof is nearly identical to [7, Theorem 5.1] with $\mathbb{E}[l(\pi_k^*)] \leq K\sqrt{\mathbb{E}[N_k]}$ replaced with $\mathbb{E}[l(\pi_k^*)] \leq \beta \mathbb{E}[N_k]^\kappa$. From (3) and Theorem 1,

$$\mathbb{E}[N_{k+1}] \leq \lambda Q + \lambda \eta \beta \mathbb{E}[N_k]^\kappa + \rho \eta \mathbb{E}[N_k] + (1 - \eta)\mathbb{E}[N_k], \qquad (9)$$

In an identical manner to [7, Theorem 5.1], it can be shown that $\mathbb{E}[N_{k+1}] \to_k \mathbb{E}[N_k]$ which may be substituted in (3) to produce a closed form constant bound on $\mathbb{E}[N_k]$. $\qquad\square$

In essence, Theorem 3 ensures that the number of unserviced tasks remains bounded at all times, i.e., the policy is stable, and thus the system does not become overburdened.

V. SIMULATION RESULTS

We demonstrate the performance of our proposed $c^p - \texttt{BATCH}$ method in a series of numerical experiments, comparing it against several state-of-the-art baselines.

Experiment setup: As our primary experiment, we consider the single vehicle DVRP in the Euclidean plane, consistent with the problem formulation and previous studies [6], [7]. In a second experiment, we examine the multi-robot case and use real-world data on a roadmap, i.e., a non-euclidean, non-convex, non-symmetric environment. The stability results established herein for the proposed method, and in [6], [7] for the BATCH, η-BATCH methods do not apply under this setting since Assumption 2 is violated. All tours were computed using the LKH-3 solver [38].

Baseline methods: We compare our approach against several DVRP baselines: BATCH [6], η-BATCH [7], and DC-BATCH [7] discussed in Section III-B. For η-BATCH we use $\eta = 0.2$, and for DC-BATCH we use $r = 10$ sectors, consistent with [7]. Finally, we also consider the event-triggered re-planning policy where the cost function is the quadratic wait times (i.e., $p = 2$ in (1)) from [34], labelled as c^2-EVENT.

A. Experiment 1 – Single Robot in the Euclidean Plane

The first simulation environment is a unit square with 3000 tasks and a uniform arrival distribution. Service times are modeled as a Gaussian with a mean of 1 and a standard deviation of 0.1. We begin by considering a single robot moving at speed $v = 1$. We notice that since the multi-robot case is solved by partitioning the environment and then solving a single-robot DVRP in each partition, the results generalize trivially to multiple robots [7].
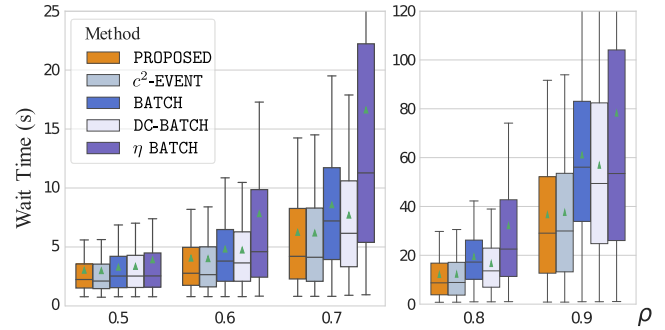

Fig. 3. Experiment 1: Comparison against baselines in the Euclidean environment, showing wait times as a function of the load factor ρ. Boxes and whiskers show the four quartiles over all trials, mean wait times are indicated by the green triangle.

a) Algorithm parameters: The proposed method has two hyper-parameters: the cost exponent p and the batch fragment η. In general, smaller values of η allow for more frequent re-planning and thus better performance; the stability result still applies provided $\eta > 0$. We choose $\eta = 0.05$, which is significantly lower than the proposed value of 0.2 from [7]. Empirically, we found that a value of $p = 1.5$ provides the best performance in terms of mean, median, and as well as number of outliers. Thus, we use $\eta = 0.05$ and $p = 1.5$ for the proposed approach, simply denoted as PROPOSED.

b) Comparison against baselines: Fig. 3 illustrates the resulting wait times for the proposed approach and the baselines. As the load factor increases towards $\rho = 0.9$, all baseline BATCH methods show an increase in mean and variance compared to the PROPOSED. The numerical results, found in Table I, additionally include a method denoted PROPOSED $\eta = 0.2$, identical to PROPOSED (i.e., $p = 1.5$), but with $\eta = 0.2$ as in η-BATCH from [7]. We include this method to explicitly illustrate that any improved performance of PROPOSED over $\eta - \texttt{BATCH}$ is not simply due to the fact that PROPOSED features a smaller value of η. We observe that under loads from $\rho = .5$ and $\rho = .6$, the different methods perform relatively similarly, only $\eta - \texttt{BATCH}$ shows a higher mean and upper end of the distributions. The method PROPOSED and PROPOSED $\eta = 0.2$ provide a consistently better performance than all baselines with exception of c^2-EVENT. While the smaller $\eta = 0.05$ used in PROPOSED gives an additional performance boost, the principal improvement comes from the proposed cost function. The mean wait times averaged over all workloads are increased by factors 1.39 for BATCH, 2.14 for η-BATCH, 1.28 for DC-BATCH, and 1.002 for c^2-EVENT, compared to PROPOSED. Further, all approaches exhibit large deviations from the mean. However, PROPOSED is always among the best with respect to variances, medians and 75th%.

In summary, the proposed method shows substantial improvements in mean and maximum wait times compared to all other BATCH methods. Moreover, c^2-EVENT – which possesses no stability guarantees – only achieves the same performance as the proposed approach, despite re-planning more frequently.

B. Experiment 2 – Multiple Robots With Real-World Data Set

As a further illustration of the applicability of our method, we simulate task arrivals for an on-site service technician based on historical 3–1–1 calls in Montreal-Nord, a borough of the city of Montreal, Canada [39] (see Fig. 4). For this experiment, we

TABLE I
RESULTS FOR EXPERIMENT 1

Method	$\rho = 0.5$		$\rho = 0.6$		$\rho = 0.7$		$\rho = 0.8$		$\rho = 0.9$	
	Mean$\pm\sigma$	95%	Mean$\pm\sigma$	95%	Mean$\pm\sigma$	95%	Mean$\pm\sigma$	95%	Mean$\pm\sigma$	95%
PROPOSED	**3.0** ±2.4	**7.7**	4.1 ±3.5	**11.2**	**6.2** ±5.7	**17.8**	**12.1** ±11.0	**33.8**	**36.5** ±31.0	**96.2**
PROPOSED η=0.2	**3.0**±2.4	**7.7**	4.1 ±3.5	**11.2**	6.3 ±5.7	17.9	12.4 ±11.2	34.4	39.2 ±31.8	99.8
c^2-EVENT	**3.0** ±2.4	7.9	**4.0** ±3.6	11.3	**6.2** ±5.8	17.9	12.2 ±11.2	34.8	37.5 ±31.4	98.1
BATCH	3.3 ±2.3	8.0	4.8 ±3.5	11.9	8.6 ±6.0	20.3	19.4 ±12.1	42.7	61.0 ±35.5	127.4
DC-BATCH	3.3 ±2.4	8.3	4.7 ±3.5	12.0	7.7 ±5.7	19.1	16.6 ±12.2	41.2	56.8 ±39.0	130.5
η-BATCH	3.9 ±3.9	11.5	7.8 ±8.6	24.9	16.6 ±16.4	49.4	32.1 ±30.5	92.1	78.2 ±76.7	229.3

We show mean standard deviation (σ) and 95-percentile of wait times in *seconds*.

TABLE II
RESULTS FOR EXPERIMENT 2, SEPARATED BY REGION

Method	Robot 1		Robot 2		Robot 3		Robot 4		Robot 5		Robot 6		All	
	Mean$\pm\sigma$	95%	Mean$\pm\sigma$	95%	Mean$\pm\sigma$	95%	Mean$\pm\sigma$	95%	Mean$\pm\sigma$	95%	Mean$\pm\sigma$	95%	Mean$\pm\sigma$	95%
PROPOSED	**24**±18	60	**27**±22	**69**	**48**±49	**144**	**66**±70	**201**	**103**±127	**342**	**367**±450	1190	**122**±250	**503**
c^2-EVENT	25±19	62	30±24	78	65±76	209	106±123	361	170±237	663	467±680	1868	161±361	715
BATCH	25±17	**59**	29±20	70	63±52	171	95±71	237	147±123	403	520±351	1214	152±250	736
DC-BATCH	25±18	62	28±21	72	57±50	161	87±71	236	142±132	429	456±359	**1187**	152±237	650
η-BATCH	26±22	69	30±28	83	75±89	246	137±150	449	192±224	628	528±531	1553	187±323	792

We show mean, standard deviation (σ) and 95-percentile of wait times in *minutes*.

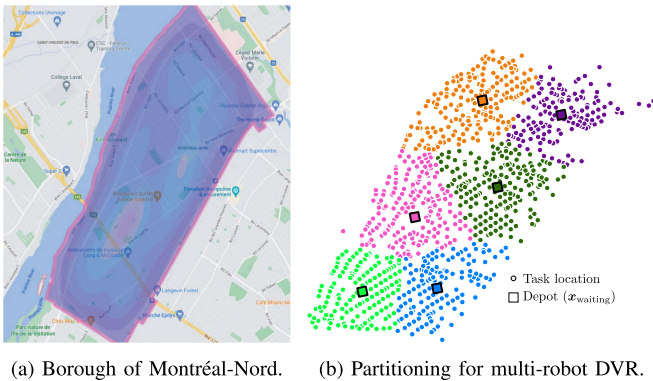(a) Borough of Montréal-Nord. (b) Partitioning for multi-robot DVR.

Fig. 4. Non-emergency assistance requests in the borough of Montréal-Nord, Montréal, Canada, between 2017-2019 [39]. (a): Map with the request density expressed as a heat map. (b): Partitions and depot locations.
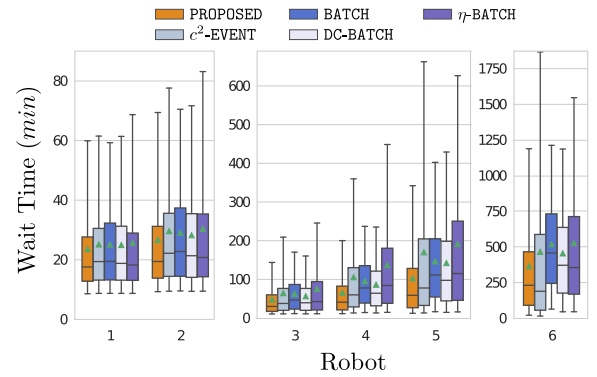

Fig. 5. Experiment 2: Comparison against baselines. Wait times for a load factor $\rho = 0.74$, separated by partition. Boxes and whiskers show the four quartiles over all trials, mean wait times are indicated by the green triangle.

deploy a fleet of $m = 6$ robots to service 5000 tasks. Thus, we divide the environment into 6 partitions using a k-nearest clustering of the request locations, shown in Fig. 4(b). Despite not guaranteeing equitable partitions, k-nearest methods are widely used in practise. All map data taken from OpenStreetMaps[1]. The average travel time over the entire map is 296 s, service times are drawn from a normal distribution with a mean of 10 min and variance of 3 min. We used an overall load factor of $\rho = .74$, selected to guarantee $\rho < 1$ within each partition.

Results are shown in Fig. 5 and Table II. We observe that since the used partitioning technique is not equitable the wait times vary significantly between partitions. In fact, the load in partitions 1 and 2 is at the lower end of moderate load ($\approx .6$), while partition 6 reaches a single-robot load factor of. 99. Nonetheless, for all partitions the proposed method achieves the lowest mean, and with exception of region 1 and 6 the lowest 95th

percentile. Mean wait times are reduced by 20% compared to the best baseline (DC-BATCH), i.e., 30 min. Moreover, we notice that the advantage of our method increases for the partitions under higher loads. In summary, we have shown the proposed method also effectively improves task wait times in a real-world multi-robot setting.

VI. CONCLUSION

We revisited the classic Dynamic Vehicle Routing Problem for stochastic arrivals. We proposed a new BATCH policy that seeks to minimize both average and maximum wait times and showed that this policy is stable even under heavy loads. In simulations we showed that the proposed method outperforms existing baseline policies under various moderate load settings ($\rho \in [0.5, 0.9]$). The baseline of event-triggered re-planning, c^2-EVENT, with a quadratic cost shows a performance comparable to ours in the Euclidean case, yet this method is computationally more burdensome and does not come with theoretical guarantees on stability. Future work should investigate the applicability

---

[1]Map data copyrighted OpenStreetMap contributors and available from https://www.openstreetmap.org

of the proposed p-norm cost for other variants of DVR, such as pick-up-and-delivery where the problem cannot be cast into several single-robot instances.

REFERENCES

[1] P. Grippa, D. A. Behrens, F. Wall, and C. Bettstetter, "Drone delivery systems: Job assignment and dimensioning," *Auton. Robots*, vol. 43, no. 2, pp. 261–274, 2019.

[2] K. Gao and J. Yu, "Capacitated vehicle routing with target geometric constraints," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 7925–7930.

[3] G. Zardini, N. Lanzetti, M. Pavone, and E. Frazzoli, "Analysis and control of autonomous mobility-on-demand systems," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 5, pp. 633–658, 2022.

[4] R. Zhang and M. Pavone, "Control of robotic mobility-on-demand systems: A queueing-theoretical perspective," *Int. J. Robot. Res.*, vol. 35, no. 1/3, pp. 186–203, 2016.

[5] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment," *Proc. Nat. Acad. Sci.*, vol. 114, no. 3, pp. 462–467, 2017.

[6] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. L. Smith, "Dynamic vehicle routing for robotic systems," *Proc. IEEE*, vol. 99, no. 9, pp. 1482–1504, Sep. 2011.

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

[8] O. Ozkan and M. Kaya, "UAV routing with genetic algorithm based matheuristic for border security missions," *Int. J. Optim. Control: Theories Appl.*, vol. 11, no. 2, pp. 128–138, 2021.

[9] Y. Liu, "An optimization-driven dynamic vehicle routing algorithm for on-demand meal delivery using drones," *Comput. Operations Res.*, vol. 111, pp. 1–20, 2019.

[10] A. Sadeghi and S. L. Smith, "Re-deployment algorithms for multiple service robots to optimize task response," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 2356–2363.

[11] N. Wilde and J. Alonso-Mora, "Online multi-robot task assignment with stochastic blockages," in *Proc. IEEE 61st Conf. Decis. Control*, 2022, pp. 5259–5266.

[12] M. Chandarana, D. Hughes, M. Lewis, K. Sycara, and S. Scherer, "Planning and monitoring multi-job type swarm search and service missions," *J. Intell. Robot. Syst.*, vol. 101, pp. 1–14, 2021.

[13] M. J. Sousa, A. Moutinho, and M. Almeida, "Decentralized distribution of UAV fleets based on fuzzy clustering for demand-driven aerial services," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2020, pp. 1–8.

[14] S. L. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," *IEEE Trans. Robot.*, vol. 28, no. 2, pp. 410–426, Apr. 2012.

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

[16] S. L. Smith, M. Pavone, F. Bullo, and E. Frazzoli, "Dynamic vehicle routing with priority classes of stochastic demands," *SIAM J. Control Optim.*, vol. 48, no. 5, pp. 3224–3245, 2010.

[17] S. Bajaj and S. D. Bopardikar, "Dynamic boundary guarding against radially incoming targets," in *Proc. IEEE Conf. Decis. Control*, 2019, pp. 4804–4809.

[18] W. Whitt, "Understanding the efficiency of multi-server service systems," *Manage. Sci.*, vol. 38, no. 5, pp. 708–723, 1992.

[19] D. J. Bertsimas and G. Van Ryzin, "Stochastic and dynamic vehicle routing with general demand and interarrival time distributions," *Adv. Appl. Probability*, vol. 25, no. 4, pp. 947–978, 1993.

[20] S. Sudhakar, V. Vijayakumar, C. S. Kumar, V. Priya, L. Ravi, and V. Subramaniyaswamy, "Unmanned aerial vehicle (UAV) based forest fire detection and monitoring for reducing false alarms in forest-fires," *Comput. Commun.*, vol. 149, pp. 1–16, 2020.

[21] H. N. Psaraftis, "Dynamic vehicle routing problems," *Veh. Routing: Methods Stud.*, vol. 16, pp. 223–248, 1988.

[22] H. N. Psaraftis, "A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem," *Transp. Sci.*, vol. 14, no. 2, pp. 130–154, 1980.

[23] B. H. O. Rios, E. C. Xavier, F. K. Miyazawa, P. Amorim, E. Curcio, and M. J. Santos, "Recent dynamic vehicle routing problems: A survey," *Comput. Ind. Eng.*, vol. 160, 2021, Art. no. 107604.

[24] D. Aksaray, C.-I. Vasile, and C. Belta, "Dynamic routing of energy-aware vehicles with temporal logic constraints," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 3141–3146.

[25] C.-I. Vasile, J. Tumova, S. Karaman, C. Belta, and D. Rus, "Minimum-violation scLTL motion planning for mobility-on-demand," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 1481–1488.

[26] C. Sarkar, H. S. Paul, and A. Pal, "A scalable multi-robot task allocation algorithm," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 5022–5027.

[27] H. N. Psaraftis, M. Wen, and C. A. Kontovas, "Dynamic vehicle routing problems: Three decades and counting," *Networks*, vol. 67, no. 1, pp. 3–31, 2016.

[28] P. Toth and D. Vigo, *Vehicle Routing: Problems, Methods, and Applications*. Philadelphia, PA, USA: SIAM, 2014.

[29] N. Soeffker, M. W. Ulmer, and D. C. Mattfeld, "Stochastic dynamic vehicle routing in the light of prescriptive analytics: A review," *Eur. J. Oper. Res.*, vol. 298, pp. 801–820, 2021.

[30] S. D. Bopardikar and V. Srivastava, "Dynamic vehicle routing in presence of random recalls," *IEEE Control Syst. Lett.*, vol. 4, no. 1, pp. 37–42, Jan. 2020.

[31] A. M. Campbell, D. Vandenbussche, and W. Hermann, "Routing for relief efforts," *Transp. Sci.*, vol. 42, no. 2, pp. 127–145, 2008.

[32] M. Huang, K. Smilowitz, and B. Balcik, "Models for relief routing: Equity, efficiency and efficacy," *Transp. Res. Part E: Logistics Transp. Rev.*, vol. 48, no. 1, pp. 2–18, 2012.

[33] M. Kulich, L. Přeučil, and J. J. M. Bront, "On multi-robot search for a stationary object," in *Proc. Eur. Conf. Mobile Robots*, 2017, pp. 1–6.

[34] F. Ferrucci and S. Bock, "A general approach for controlling vehicle en-route diversions in dynamic vehicle routing problems," *Transp. Res. Part B: Methodological*, vol. 77, pp. 76–87, 2015.

[35] M. Chandarana, M. Lewis, K. Sycara, and S. Scherer, "Determining effective swarm sizes for multi-job type missions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4848–4853.

[36] J. Beardwood, J. H. Halton, and J. M. Hammersley, "The shortest path through many points," in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 55. Cambridge, U.K.: Cambridge Univ. Press, 1959, pp. 299–327.

[37] M. Haimovich and A. H. Rinnooy Kan, "Bounds and heuristics for capacitated routing problems," *Math. Operations Res.*, vol. 10, no. 4, pp. 527–542, 1985.

[38] K. Helsgaun, *An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems*. Roskilde, Denmark: Roskilde Univ., 2017.

[39] Government and Municipalities of Québec, "3-1-1 Requests (archives 2016-2019)," *Citizen Serv. Requests (311 Requests)*, 2023. [Online]. Available: https://open.canada.ca/data/en/dataset/5866f832-676d-4b07-be6a-e99c21eb17e4