DMVP: Foremost Waypoint Coverage of Time-Varying Graphs

Eric Aaron[1], Danny Krizanc[2](✉), and Elliot Meyerson[2]

[1] Computer Science Department, Vassar College, Poughkeepsie, NY, USA
eaaron@cs.vassar.edu

[2] Department of Mathematics and Computer Science,
Wesleyan University, Middletown, CT, USA
{dkrizanc,ekmeyerson}@wesleyan.edu

Abstract. We consider the Dynamic Map Visitation Problem (DMVP), in which a team of agents must visit a collection of critical locations as quickly as possible, in an environment that may change rapidly and unpredictably during the agents' navigation. We apply recent formulations of time-varying graphs (TVGs) to DMVP, shedding new light on the computational hierarchy $\mathcal{R} \supset \mathcal{B} \supset \mathcal{P}$ of TVG classes by analyzing them in the context of graph navigation. We provide hardness results for all three classes, and for several restricted topologies, we show a separation between the classes by showing severe inapproximability in $\mathcal{R}$, limited approximability in $\mathcal{B}$, and tractability in $\mathcal{P}$. We also give topologies in which DMVP in $\mathcal{R}$ is fixed parameter tractable, which may serve as a first step toward fully characterizing the features that make DMVP difficult.

1 Introduction

In navigation-oriented application domains such as autonomous mobile robots, wireless sensor networks, security, surveillance, mechanical inspection, and more, graph representations are commonly employed for formulating and analyzing the central navigation or area inspection problems. Many approaches to coverage problems [11–13] are based on static graph representations, as are visitation problems [1] or related combinatorial optimization problems such as the k-Chinese Postman Problem [3,7] and k-Traveling Repairman Problem [14,15]. But static graph structures do not represent the dynamic environments that can occur in applications of autonomous robots or non-player characters in video games and virtual worlds. In this paper, we present the *Dynamic Map Visitation Problem* (DMVP), applying recent formulations of *highly dynamic graphs* (or *time-varying graphs* (TVGs)) [9,23] to an essential graph navigation problem: In DMVP, a team of agents must inspect a collection of critical locations on a map (represented as a graph) as quickly as possible, but the agents' environment may change during navigation.

The application of TVG models is essential to DMVP. In applications such as planetary exploration [26], search and rescue in hazardous environments

D. Kratsch and I. Todinca (Eds.): WG 2014, LNCS 8747, pp. 29–41, 2014.
DOI: 10.1007/978-3-319-12340-0_3

(e.g., natural disasters, areas of armed conflict), or even ad-hoc network inspection, many aspects of the structure of graph waypoints and edges governing navigation can change during agent navigation, and TVG models can capture variation in graph structure in ways that static graphs cannot. Our paper presents new results about DMVP complexity and demonstrates distinctions among classes of TVGs; details of our main results are summarized in Sect. 1.2.

When incorporating dynamics into a problem such as DMVP, there are many options for how to constrain/model the dynamics of the graph. Dynamics can be deterministic (e.g., [8,19,20,24,27]) or stochastic (e.g., [5,10]). In this paper, to provide a foundation for future work, and exemplify the aspects of topologies and dynamics that make our problem easy or hard, we focus on the deterministic case. The deterministic approach is also particularly relevant for situations in which some prediction of changes is feasible. Quite a bit of this previous work has required that the graph be connected *at all times* [10,20,22]. Indeed, for complete map visitation to be possible, every critical location must be eventually reachable. However, in application environments such as those outlined above, at any given time the waypoint graph may be disconnected. Our model must be general enough to allow for this phenomenon.

We adopt three classes of TVGs, each of which places constraints on edge dynamics. In $\mathcal{R}$, edges must reappear eventually; in $\mathcal{B}$, edges must appear within some time bound; in $\mathcal{P}$ edge appearances are periodic. These classes have proven to be critical to the TVG taxonomy [9]. They have been studied with respect to problems such as broadcast [8] and exploration [16,19], with results relating to feasibility of computation and bounds on broadcast and exploration time. $\mathcal{R}$, $\mathcal{B}$, and $\mathcal{P}$ place intuitive constraints on the nature of dynamic navigation domains. Even the assumption of periodicity of edges has applications to navigation of transportation networks [16,19], as well as environments periodically patrolled by other agents, who can prohibit or guarantee safe traversal of an edge.

In this paper, we shed further light on the computational hierarchy of $\mathcal{R}$, $\mathcal{B}$, and $\mathcal{P}$ [8], by analyzing them in the context of DMVP, a natural but difficult problem in global navigation. We provide hardness results for all three classes. For several restricted topologies, we demonstrate separation between the classes by showing severe inapproximability in $\mathcal{R}$, limited approximability in $\mathcal{B}$, and tractability in $\mathcal{P}$. We also give topologies in which DMVP in $\mathcal{R}$ is tractable and fixed parameter tractable, which may serve as a first step towards fully characterizing the topological features that make DMVP difficult. Because our goal in this paper is to cleanly differentiate the classes of dynamics we are exploring, rather than explore the interactions between multiple agents, our results here focus on the case of a single agent.

1.1 Definitions and TVG Concepts

As a foundation for our work, we adopt the definitions below from Santoro et al. [9].

Definition 1. *A TVG (time-varying graph, dynamic graph, or dynamic network) is a five-tuple $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$, where $\mathcal{T} \subseteq \mathbb{T}$ is the* lifetime *of the system,* presence function $\rho(e,t) = 1 \iff$ *edge $e \in E$ is available at time $t \in \mathcal{T}$, and* latency function $\zeta(e,t)$ *gives the time it takes to cross e if starting at time t. The graph $G = (V, E)$ is called the* underlying graph *of $\mathcal{G}$, with $|V| = n$.*

In the most general case, $\mathbb{T}$ can be $\mathbb{R}$, and edges can be directed. However, in our work we consider the discrete case in which $\mathbb{T} = \mathbb{N}$, edges are undirected, and all edges have uniform travel cost $\zeta(e,t) = 1$ at all times. If agent a is at u, and edge (u, v) is available at time τ, then a can take (u, v) during this time step, visiting v at time $\tau + 1$. As a traverses $\mathcal{G}$ we say a both *visits* and *covers* the vertices in its traversal, and we will henceforth use these terms interchangeably. A *temporal subgraph* of a TVG $\mathcal{G}$ results from restricting the lifetime $\mathcal{T}$ of $\mathcal{G}$ to some $\mathcal{T}' \subseteq \mathcal{T}$.

Definition 2. *$\mathcal{J} = \{(e_1, t_1), ..., (e_k, t_k)\}$ is a* journey $\iff$ *$\{e_1, ..., e_k\}$ is a walk in G (called the* underlying walk *of $\mathcal{J}$), $\rho(e_i, t_i) = 1$ and $t_{i+1} \geq t_i + \zeta(e_i, t_i)$ for all $i < k$. The* topological length *of $\mathcal{J}$ is k, the number of edges traversed. The* temporal length *is the duration of the journey: $(arrival\ date) - (departure\ date)$.*

Given a date t, a journey from u to v departing on or after t whose arrival date t' is soonest is called *foremost*; whose topological length is minimal is called *shortest*; and whose temporal length is minimal is called *fastest*.

In [9], a hierarchy of thirteen classes of TVG's is presented. In related work on exploration [16] and broadcast [8], focus is primarily on the chain $\mathcal{R} \supset \mathcal{B} \supset \mathcal{P}$ defined below. We adopt these classes into our domain, which we believe enforce natural constraints in our application environments.

Definition 3 *(Recurrent edges). $\mathcal{R}$ is the class of all TVG's $\mathcal{G}$ such that G is connected, and $\forall e \in E, \forall t \in \mathcal{T}, \exists t' > t$ s.t. $\rho(e, t') = 1$.*

Definition 4 *(Time-bounded recurrent edges). $\mathcal{B}$ is the class of all TVG's $\mathcal{G}$ such that G is connected, and $\forall e \in E, \forall t \in \mathcal{T}, \exists t' \in [t, t + \Delta)$ s.t. $\rho(e, t') = 1$, for some integer Δ.*

Definition 5 *(Periodic edges). $\mathcal{P}$ is the class of all TVG's $\mathcal{G}$ such that G is connected, and $\forall e \in E, \forall t \in \mathcal{T}, \forall k \in \mathbb{N}, \rho(e, t) = \rho(e, t + kp)$ for some integer p. p is called the* period *of $\mathcal{G}$.*

As much as possible, we also take standard notation and terms from the graph theory literature. We rely on several underlying graph topologies. A *star* is a tree in which at most one vertex has degree greater than one. The leaves of a star are called *points*. A *spider* is a tree in which at most one vertex has degree greater than two. In other words, a spider consists of a set of vertex-disjoint paths, called *arms*, each of which has exactly one endpoint connected to the common central vertex c. A *comb* is a max-degree 3 tree, in which there exists a simple path containing every vertex of degree 3. Such a path is called a *backbone* of the comb. Paths edge-disjoint to the backbone are called *arms*. A leaf distance

1 from the backbone is called a *tooth*. An *r-almost-tree* is a connected graph with $|V| + r - 1$ edges, that is, r edges can be removed to produce a tree.

Problem. *Given a TVG $\mathcal{G}$ and a set of starting locations S for k agents in G, the TVG foremost coverage or dynamic map visitation problem (*DMVP*) is the task of finding journeys starting at time 0 for each of these k agents such that every node in V is in some journey, and the maximum temporal length among all k journeys is minimized. The decision variant asks whether these journeys can be found such that no journey ends after time t.*

We think of the input $\mathcal{G}$ as a temporal subgraph of some TVG $\mathcal{G}_\infty$ with lifetime $\mathbb{N}$ and the same edge constraints as $\mathcal{G}$. Thus, the limited information provided in $\mathcal{G}$ is used to compute complete solutions for agents covering $\mathcal{G}_\infty$. When unspecified, assume that DMVP refers to DMVP for a single agent.

1.2 Main Results

Our results are summarized in Table 1. We show that DMVP in $\mathcal{R}$ is NP-hard to approximate within any factor, when the underlying graph G is restricted to a star or tree of max degree 3. We show that in $\mathcal{B}$ this problem is NP-hard to approximate within any factor less than Δ, when G is restricted to a spider or tree of max degree 3. We show that in $\mathcal{P}$, DMVP is NP-complete when $p = 1$, and that there is a nontrivial class of graphs for which $p = 2$ is NP-hard, but $p = 1$ is not.

We show that in $\mathcal{R}$, DMVP is solvable in $O(T)$ when G is a path, $O(Tn)$ when G is a cycle, and $O(Tn^3 + n^2 2^n)$ for general graphs, where T is the duration of $\mathcal{G}$, as defined in Sect. 2. Furthermore, in $\mathcal{R}$, DMVP is fixed parameter tractable when G is an m-leaf $O(1)$-almost tree, and poly-time solvable when $m = O(\lg n)$. In $\mathcal{B}$, we demonstrate a tight Δ-approximation for trees, and a 2Δ-approximation for general graphs. We demonstrate a class of problems which are NP-hard in $\mathcal{B}$, but solvable by an online algorithm in $\mathcal{P}$. We show that DMVP in $\mathcal{P}$ is solvable in polynomial time when G is a spider, for fixed p, and we show that when $p = 2$, DMVP is solvable in linear time for general trees.

Table 1. DMVP separations and results by TVG class and graph class

DMVP separations			
TVG class	spiders	max-degree 3 trees	general trees
$\mathcal{R}$	no approx.	no approx.	no approx.
$\mathcal{B}$	tight Δ-approx.	tight Δ-approx.	tight Δ-approx.
$\mathcal{P}$	in P, for fixed p	$O(n)$ exact, for $p = 2$	$O(n)$ exact, for $p = 2$
$\exists$ graph class s.t. DMVP NP-hard in $\mathcal{P}$ with $p = 2$, easy with $p = 1$.			

Complexity of exact algorithms in $\mathcal{R}$				
path	cycle	general graphs	m-leaf c-almost trees	$O(\lg n)$-leaf c-almost trees
$O(T)$	$O(Tn)$	$O(Tn^3 + n^2 2^n)$	in FPT	in P

The remainder of this paper is organized as follows: preliminaries (2), lower bounds (3), upper bounds (4), open problems and discussion (5). Details of all missing proofs can be found in the full version of this paper [2].

2 Preliminaries

For the minimization problem DMVP($\mathcal{G}, S$) and the corresponding decision problem DMVP($\mathcal{G}, S, t$), input is viewed as a sequence of graphs G_i each represented as an adjacency matrix, with an associated integer duration t_i, i.e. $\mathcal{G} = (G_1, t_1), (G_2, t_2), ..., (G_m, t_m)$, where G_1 appears initially at time zero. Let $T = \sum_{i=1}^{m} t_i$. Note that since each t_i can be encoded in $O(\lg t_i)$ space, it is possible for T to be exponential in the size of $\mathcal{G}$. The following observation is required to show that the number of time steps of $\mathcal{G}$ that need to be considered for DMVP is in fact polynomial in the size of $\mathcal{G}$.

Observation 1. *When computing DMVP over $\mathcal{G}$, it is not necessary to consider each static temporal subgraph (G_i, t_i) for more than $2n - 3$ time steps.*

The idea is that on a static graph anything that can be accomplished in more than $2n-3$ steps can be accomplished in $2n-3$ steps or fewer. By Observation 1, for any $t_i > 2n - 3$, when computing DMVP, all time steps after the first $2n - 3$ can be ignored (skipped). DMVP over $\mathcal{G}$ can be computed by computing DMVP over $\mathcal{G}' = (G_1, \min(t_1, 2n - 3)), ..., (G_m, \min(t_m, 2n - 3))$, and adding back the cumulative time skipped before completion. $\mathcal{G}'$ can clearly be derived from $\mathcal{G}$ in $O(m)$ time. The total duration of $\mathcal{G}'$ is $T' = \sum_{i=1}^{m} \min(t_i, 2n - 3) < 2nm - 3m$, which is polynomial in $|\mathcal{G}|$. Let $\epsilon(\tau)$ be the time skipped through time τ. $\epsilon(\tau)$ can be simply calculated for all $\tau \leq T'$ in $O(T')$ time. A similar $O(T')$ preprocessing step can be run to associate each time $\tau \in T'$ with the corresponding available static graph G_i, enabling $O(1)$ edge presence lookups $\rho(e, \tau)$.

Since all of the algorithms we present run in $\Omega(T')$ time, we can run these preprocessing steps for every instance of DMVP and not affect the asymptotic running time. Therefore, for the sake of simplicity, for the rest of our results we assume that this preprocessing has taken place, i.e., we think of $\mathcal{G}$ as $\mathcal{G}'$ and T as T', thereby avoiding the exponential nature of T. Note also that for the case of $\mathcal{P}$, the constraint of periodicity implies that it is only necessary to look at p consecutive time steps of the input.

3 Lower Bounds

As motivation for many of the results in this paper, it is important to note that MVP for a single agent is solvable in linear time on trees [1]. To characterize the difficulty of DMVP in $\mathcal{R}$, we first show inapproximability over stars. A similar theorem was independently discovered in [25].

Theorem 1. *DMVP for a single agent in $\mathcal{R}$ is NP-hard to approximate within any factor, even when the underlying graph is a star.*

This inapproximability also holds over the restriction of underlying graphs to trees of max-degree 3, in particular, combs.

Theorem 2. *DMVP for a single agent in $\mathcal{R}$ is NP-hard to approximate within any factor, even when the underlying graph is a comb.*

We have a similar set of lower bounds for the case of $\mathcal{B}$, but with *some* ability to approximate. We later show (Theorem 11) that these approximation bounds are indeed tight for all trees.

Theorem 3. *DMVP for a single agent in $\mathcal{B}$ is NP-hard to approximate within any factor less than Δ, even when the underlying graph is a spider, $\forall \Delta > 1$.*

Theorem 4. *DMVP for a single agent in $\mathcal{B}$ is NP-hard to approximate within any factor less than Δ, even when the underlying graph is a comb, $\forall \Delta > 1$.*

As is shown in Sect. 4, there is a much greater potential for tractability of DMVP in $\mathcal{P}$ than in $\mathcal{B}$ or $\mathcal{R}$. However, the next result follows immediately via reduction from hamiltonian path by simply restricting t to $n - 1$.

Theorem 5. *DMVP for a single agent in $\mathcal{P}$ is NP-complete, when $p = 1$.*

DMVP in $\mathcal{P}$ for $p = 1$ is then also NP-complete for all classes of graphs for which hamiltonian path is NP-complete, in particular, planar graphs of maximum degree 3, bridgeless undirected planar 3-regular bipartite graphs, and 3-connected 3-regular bipartite graphs [4]. To show that $\mathcal{P}$ is an interesting dynamics class for DMVP in its own right, it is important to show that DMVP yields different hardness results over $\mathcal{P}$ than over static graphs. Thus, we construct a class of graphs for the following result:

Theorem 6. *There is an infinite class of graphs C such that DMVP for a single agent in $\mathcal{P}$ over graphs in C is NP-complete when $p = 2$, but trivial when $p = 1$.*

4 Upper Bounds

In this section, we map out a class of graphs over which DMVP in $\mathcal{R}$ is solvable in polynomial time. We first start with a very useful lemma. Note that a related observation (about turning around on a ring) was made in [20].

Lemma 1 (Turning around lemma). *There is always an optimal solution J that never turns around at a degree 2 vertex of the edge-induced subgraph of J in G.*

See Fig. 1. The idea is that if an agent turns around at such a vertex, that vertex must also be reached at some other time in J. We apply Lemma 1 to get the following solvability results for restricted classes of underlying graphs.

Theorem 7. *DMVP for a single agent in $\mathcal{R}$ on a path is solvable in $O(T)$ time.*

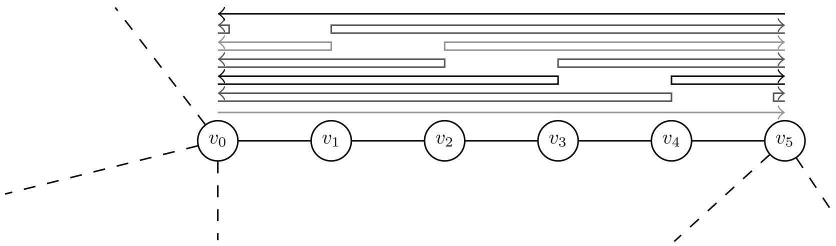

Fig. 1. The 7 ways, satisfying Lemma 1, of covering the vertices of a length 5 path with degree 2 intermediate nodes.

Theorem 8. *DMVP for a single agent in $\mathcal{R}$ on a cycle is solvable in $O(Tn)$ time.*

Now we show that despite the severe inapproximability of DMVP over $\mathcal{R}$, we can always compute an optimal solution in exponential time.

Theorem 9. *DMVP for a single agent in $\mathcal{R}$ is solvable in $O(Tn^3 + n^2 2^n)$ time.*

Proof. The proposed algorithm first computes all-pairs-all-times-foremost-journey for input TVG $\mathcal{G}$, using a straightforward dynamic programming algorithm, then uses this information to run another dynamic programming algorithm, conceived along the lines of a standard method for TSP [6].

Let d_{uv}^t be the length of the foremost journey from u to v, starting at time t. Algorithm 1 computes d_{uv}^t for all vertex pairs (u, v), and times $t \in \mathcal{T}$ for a given TVG $\mathcal{G}$.

At all times t, for all vertices $u \in V$, d_{uu}^t is clearly 0. At time T, the time limit has been reached, so an agent cannot move to another vertex in any guaranteed time, and thus we set $d_{uv}^T = \infty$ for all $u \neq v$. For all $T - 1 \geq t \geq 0$, in the worst case an agent can wait at u for one step, and take the foremost journey to v starting at time $t + 1$. If there is a better journey than this, it must consist of not waiting, rather taking one of the edges available at time t from u to some vertex k. Subsequently taking the foremost journey from k to v starting at time $t+1$ results in an optimal journey through k. Algorithm 1 clearly runs in $O(Tn^3)$ time, and uses $O(Tn^2)$ space.

Algorithm 2 uses the d_{uv}^t values computed by Algorithm 1 to compute the cost of a minimal solution to DMVP for a single agent in $\mathcal{R}$. Let $V' \subseteq V$ and $c(V', v)$ be the minimal time it takes to visit all vertices in V' starting at vertex s at time 0 and ending at vertex $v \in V'$.

After initializing the minimal costs for visiting subsets up to size 2, the algorithm repeatedly uses the minimal costs for size i subsets to calculate $c(V', v)$ for each size $i + 1$ subset V' and $v \neq s \in V'$. Once computed up to size n, the algorithm returns the minimal cost among journeys that cover all vertices. This is an optimal solution to DMVP as it is the minimum cost of taking foremost journeys between vertices that results in a complete cover. There are 2^n subsets

Algorithm 1. All-pairs-all-times-foremost-journey($\mathcal{G}$)

```
for all $u, v \in V \times V$ do                    ▷ Initialize base case for $t = T$.
    if $u = v$ then
        $d^T_{uv} = 0$
    else
        $d^T_{uv} = \infty$                         ▷ Since input ends at $T$, agent cannot move.
for $t = T - 1, ..., 0$ do                          ▷ Work backwards until start time $t = 0$.
    for all $u, v \in V \times V$ do
        if $u = v$ then
            $d^t_{uv} = 0$
        else
            $d^t_{uv} = d^{t+1}_{uv} + 1$           ▷ In worst case, just wait at $u$.
            for all $k \in V$ do
                if $\rho((u, k), t) = 1$ then       ▷ Check for better route.
                    $d^t_{uv} = \min(d^t_{uv}, d^{t+1}_{kv} + 1)$
```

Algorithm 2. *DMVP*($\mathcal{G}, \{s\}$)

```
$c(\{s\}, s) = 0$                                   ▷ Initialize subset of size 1.
for all $v \neq s \in V$ do                         ▷ Initialize subsets of size 2.
    $c(\{s, v\}, v) = d^0_{sv}$
for i = 3,...,n do                                  ▷ Build up to subsets of size n.
    for all $S \subseteq V s.t. |S| = i$ do
        for all $v \neq s \in V$ do
            $c(V', v) = \min_{u \neq s \in V' \setminus \{v\}}(c(V' \setminus \{v\}, u) + d^{c(V' \setminus \{v\}, u)}_{uv})$
  return $\min_{v \neq s \in V}(c(V, v))$
```

of V, and so $n2^n$ subset-vertex pairs of the form (V', v). For each of these, the algorithm computes the minimum of $O(n)$ values. So, Algorithm 2 has running time $O(n^2 2^n)$. Since it saves one cost for each subset-vertex pair, Algorithm 2 also uses $O(n2^n)$ space. Sequentially running Algorithm 1 followed by Algorithm 2, we have a complete algorithm for DMVP for a single agent in $\mathcal{R}$, with combined running time $O(Tn^3 + n^2 2^n)$. $\square$

We use Theorem 9 to generalize Theorems 7 and 8 with the following:

Theorem 10. *DMVP in $\mathcal{R}$ is fixed parameter tractable, when G is an m-leaf c-almost-tree, for fixed parameter m, and c constant.*

Proof. First, consider the restricted case where G is an m-leaf tree. Since every leaf must be visited, and visiting all leaves implies coverage of the entire tree, there is a minimal solution that can be thought of as an ordering of the set of leaves of G, and the foremost journeys between them. In this case, there is only one *way* to visit any node, namely, on the way to a leaf. Using this observation and Algorithm 2 from the proof of Theorem 9, we see that we only need to consider all orderings of leaves, instead of all orderings of vertices, yielding a run time of $O(Tn^3 + m^2 2^m)$, which is indeed fixed parameter tractable for parameter m.

Suppose the underlying graph G of $\mathcal{G}$ is an m-leaf c-almost-tree. Consider all edges e such that removing e from G results in a $(c-1)$-almost-tree. Each of these edges lies on some path P such that removing any edge of P will similarly result in a $(c-1)$-almost-tree, and every intermediate vertex on the path has degree 2. Suppose P is the path $v_0 ... v_l$. Since G is an m-leaf c-almost-tree, there are $O(m)$ paths of this type. The edge-induced subgraph G' of the underlying walk of an optimal covering of $\mathcal{G}$ can be any $(c-c')$-almost-tree $\subseteq G$, for $0 \leq c' \leq c$. For each c', a solution involves selecting c' paths, each of $O(n)$ length, from which to remove an edge. So, there are $O(m^{c'} n^{c'})$ possible choices of $(c-c')$-almost-trees, and thus $O(\sum_{c'=0}^{c}(m^{c'} n^{c'})) = O(m^c n^c)$ choices for G'. Every G' has no more than $m+2c$ leaves. Since every edge of G' is covered, by Lemma 1, there are at most 2 ways to cover each of the remaining $O(m)$ paths $v_0 ... v_l$ of intermediate vertex degree 2, namely: entering at v_0 and exiting at v_l, or entering at v_l and exiting at v_0. Augment the set of leaves to be ordered in a solution with the selected ways of covering these paths, that is, select one of the consecutive subsequences $v_0 v_l$ or $v_l v_0$ to be in the ordering. With this augmentation, we still have a set of $O(m)$ elements to be ordered, the optimal ordering of which can be computed via Theorem 9 in $O(Tn^3 + m^2 2^m)$ time. The minimum over all ways of covering G' can then be computed in $O(2^m)O(Tn^3 + m^2 2^m) = O(Tn^3 2^m + m^2 2^{2m})$. The overall minimum cost for covering $\mathcal{G}$ can then be computed by taking the minimum cost over all $O(m^c n^c)$ edge-induced subgraphs in $O(m^{c'} n^c) O(Tn^3 2^m + m^2 2^{2m}) = O(Tn^{3+c} f(m))$ time. $\square$

The following result follows immediately for the case when $m = O(\lg n)$.

Corollary 1. *DMVP in $\mathcal{R}$ is solvable in polynomial time, if $\mathcal{G}$ is an $O(\lg n)$-leaf c-almost-tree, for c constant.*

We conjecture (see Sect. 5) that the maximal class of graphs over which DMVP in $\mathcal{R}$ is poly-time solvable is the class of all graphs with polynomially many spanning trees, all of which have $O(\lg n)$ leaves.

Since DMVP in $\mathcal{B}$ is bounded by $2\Delta n$, the running time of the algorithm in Theorem 9 on TVGs over $\mathcal{B}$ reduces to $O(\Delta n^4 + n^2 2^n)$. We also see that we are able to greatly improve on approximation from $\mathcal{R}$ to $\mathcal{B}$:

Theorem 11. *DMVP in $\mathcal{B}$ over a tree can be Δ-approximated in $O(n)$ time. This approximation is tight.*

Theorem 12. *DMVP in $\mathcal{B}$ can be 2Δ-approximated by any online spanning tree traversal of G.*

These approximation upper bounds derive from static solutions [1], but waiting at most $\Delta - 1$ steps for each edge to appear. Theorems 3 and 4 show the tightness of Theorem 11. Here, $\mathcal{B}$ is starkly differentiated from $\mathcal{R}$ in that we have at least some ability to approximate in $\mathcal{B}$. See Sect. 5 for a further discussion of the relationship between these two classes.

Similar to the case for $\mathcal{B}$, DMVP in $\mathcal{P}$ is bounded by $2pn$, so the running time of the algorithm in Theorem 9 reduces to $O(pn^4 + n^2 2^n)$. To exemplify the differences between $\mathcal{P}$ and $\mathcal{B}$, and motivate interest in the tractability of DMVP over $\mathcal{P}$, we first give the following simple example:

Theorem 13. *For any p, there is a class of problems over combs, for which DMVP in $\mathcal{B}$ is NP-hard, but in $\mathcal{P}$ is solvable by the online algorithm: take arms when you get to them.*

The quality of $\mathcal{P}$ we take advantage of above is that if the fastest journey between two nodes takes d steps, the foremost journey can take no longer than $d+(p-1)$, while in $\mathcal{B}$ it can be as bad as $d\Delta$. We again harness this effect in the following result, a stronger theorem in the context of our inapproximability results for $\mathcal{R}$ and $\mathcal{B}$ (Theorems 1 and 3):

Theorem 14. *DMVP in $\mathcal{P}$ over a spider is solvable in polynomial time, for fixed p.*

Proof Sketch: Each arm can be classified into one of $O(p^3)$ equivalence classes, based on the return time and cost above fastest of taking that arm for all time $t \equiv i \mod p$. A solution is an ordering of arms by when they are traversed. Suppose S is an optimal solution. Every length p subsequence of S must contain a shortest subsequence (called a *pattern*) that begins and ends at an equivalent time $t, t' \equiv j \mod p$. Patterns can be moved to any location beginning at some $t'' \equiv j \mod p$ without changing the cost of the solution. We can then cluster patterns by start time, without changing the cost of the solution, so that pairs of consecutive clusters are separated by no more than $p-1$ arms. There are only $O((p-1)!p^5 n^{(p^3)^{p+1}}) = O(n^{(p^3)^{p+1}})$ solutions of this form, one of which must be minimal. $\square$

This polynomial runtime can be significantly improved for the case of $p = 2$.

Theorem 15. *DMVP in $\mathcal{P}$ over a tree is solvable in $O(n)$ time, when $p = 2$.*

Proof Sketch: Consider the problem for an agent starting at root o at time 0. We can show by induction that there is always an optimal solution that never enters any of the subtrees of o's children more than once. We then show each subtree is of one of three types: (11) fastest coverage is always available, (10) fastest coverage is available at even times, and (01) fastest coverage is available at odd times, with both 01 and 10 subtrees taking odd time to cover fastest, and 11's taking even time. Alternating between 10's and 01's, and then taking the remaining subtrees in any order, before ending at a *furthest* leaf, results in an optimal solution, as we maximize how many subtrees are traversed optimally. Using dynamic programming, we can compute bottom-up the type and cost of the maximal subtree rooted at each node v, in $O(\deg(v))$ time for each. $\square$

5 Open Problems and Discussion

This paper presents significant advances towards isolating the maximal class of graphs over which DMVP in $\mathcal{R}$ is solvable in polynomial time. We conjecture that this maximal class is the class of all graphs with polynomially many spanning trees, all of which have $O(\lg n)$ leaves. Furthermore, we conjecture that this class is equivalent for $\mathcal{R}$ and $\mathcal{B}$. But we are very interested in expanding this class with respect to $\mathcal{P}$, motivated by our solvability results for $\mathcal{P}$ over subclasses of trees. We have shown that for the case of $p = 2$, DMVP for a single agent over general trees can be computed in linear time. This result relies on the fact that we know how to optimally piece together patterns with period 2. New methods for finding optimal pattern sequences could greatly reduce computation for cases of $p > 2$. We are hopeful that DMVP in $\mathcal{P}$ will be shown to be poly-time solvable over arbitrary trees or at least bounded degree trees, for greater values p both fixed and not fixed.

Considering $\mathcal{B}$ and $\mathcal{R}$, $\mathcal{B}$ is clearly differentiated from $\mathcal{R}$ in that we have at least some ability to approximate in $\mathcal{B}$. There remains, however, an important open question: Is there any class C of underlying graphs such that DMVP is NP-hard over C in $\mathcal{R}$, but not in $\mathcal{B}$? We are particularly interested in whether or not DMVP in $\mathcal{B}$ is NP-hard when the underlying graph is a star and Δ is fixed, in particular, when $\Delta = 2$. Note: The proof of Theorem 1 implies it is hard when Δ is some relatively small function of the input. We conjecture that even for $\Delta = 2$ this problem is NP-hard, but the highly-restricted nature of the input makes an answer to this problem more elusive than some of the others we have results for. Towards an answer to this question, we give the following observation:

Observation 2. *DMVP in $\mathcal{R}$ over a spider with arms of uniform length l, e.g., a star (when $l = 1$), can be decided in polynomial time, when t disallows waiting, i.e., $t = 2n - l - d$, where d is topological distance from s to c.*

Proof Idea: We can reduce this problem to the problem of finding a perfect bipartite matching between arms and blocks of time, for which there are many known efficient polynomial time algorithms, e.g., [18].

Overall, our results show some instances where DMVP is tractable as well as showing that DMVP faces difficult computational challenges for some natural classes of underlying topologies and dynamics. These challenges motivate research into online, multi-agent solutions to the problem, since in many cases having a complete global view of the present and future does not appear to be very helpful; moreover, in agent-oriented applications ranging from software agents to mobile robots, the information available to teams of agents can be bounded both temporally and geographically, and such online, multi-agent approaches could be well suited to agent dynamics without diminishing tractability.

References

1. Aaron, E., Kranakis, E., Krizanc, D.: On the complexity of the multi-robot, multi-depot map visitation problem. In: IEEE MASS, pp. 795–800 (2011)
2. Aaron, E., Krizanc, D., Meyerson, E.: DMVP: foremost waypoint coverage of time-varying graphs (2014). http://arxiv.org/abs/1407.7279

3. Ahr, D., Reinhelt, G.: A tabu search algorithm for the min-max k-Chinese postman problem. Comput. Oper. Res. **33**(12), 3403–3422 (2006)
4. Akiyama, T., Nishizeki, T., Saito, N.: NP-completeness of the Hamiltonian cycle problem for bipartite graphs. J. Inf. Process. **3**(2), 73–76 (1980)
5. Baumann, H., Crescenzi, P., Fraigniaud, P.: Parsimonious flooding in dynamic graphs. Distrib. Comput. **24**(1), 31–44 (2011)
6. Bellman, R.: Dynamic programming treatment of the travelling salesman problem. JACM **9**(1), 61–63 (1962)
7. Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, B., Raghavan, P., Sudan, M.: The minimum latency problem. In: Proceedings of 26th STOC, pp. 163–171 (1994)
8. Casteigts, A., Flocchini, P., Mans, B., Santoro, N.: Deterministic computations in time-varying graphs: broadcasting under unstructured mobility. In: Calude, C.S., Sassone, V. (eds.) TCS 2010. IFIP AICT, vol. 323, pp. 111–124. Springer, Heidelberg (2010)
9. Casteigts, A., Flocchini, P., Quattrociocchi, W., Santoro, N.: Time-varying graphs and dynamic networks. IJPED **27**(5), 387–408 (2012)
10. Avin, C., Koucký, M., Lotker, Z.: How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 121–132. Springer, Heidelberg (2008)
11. Choset, H.: Coverage for robotics: a survey of recent results. Ann. Math. AI **31**, 113–126 (2001)
12. Correll, N., Rutishauser, S., Martinoli, A.: Comparing coordination schemes for miniature robotic swarms. In: Springer Tracts in Advanced Robotics, vol. 39, pp. 471–480 (2008)
13. Easton, K., Burdick, J.: A coverage algorithm for multi-robot boundary inspection. In: Proceedings of ICRA, pp. 727–734 (2005)
14. Edmonds, J., Johnson, E.: Matching, Euler tours and the Chinese postman problem. Math. Program. **5**, 88–124 (1973)
15. Fakcharoenphol, J., Harrelson, C., Rao, S.: The k-traveling repairman problem. In: Proceedings of 39th STOC (2007)
16. Flocchini, P., Mans, B., Santoro, N.: On the exploration of time-varying networks. Theor. Comput. Sci. **469**, 53–68 (2013)
17. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, New York (1979)
18. Hopcroft, J., Karp, R.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM J. Comput. **2**(4), 225–231 (1973)
19. Ilcinkas, D., Wade, A.M.: On the power of waiting when exploring public transportation systems. In: Fernàndez Anta, A., Lipari, G., Roy, M. (eds.) OPODIS 2011. LNCS, vol. 7109, pp. 451–464. Springer, Heidelberg (2011)
20. Ilcinkas, D., Wade, A.M.: Exploration of the T-interval-connected dynamic graphs: the case of the ring. In: Moscibroda, T., Rescigno, A.A. (eds.) SIROCCO 2013. LNCS, vol. 8179, pp. 13–23. Springer, Heidelberg (2013)
21. Karp, R.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum, New York (1972)
22. Kuhn, F., Lynch, N., Oshman, R.: Distributed computation in dynamic networks. In: ACM Symposium on Theory of Computing (2010)
23. Kuhn, F., Oshman, R.: Dynamic networks: models and algorithms. ACM SIGACT News **42**(1), 82–96 (2011)

24. Mans, B., Mathieson, L.: On the treewidth of dynamic graphs. In: Du, D.-Z., Zhang, G. (eds.) COCOON 2013. LNCS, vol. 7936, pp. 349–360. Springer, Heidelberg (2013)
25. Michail, O., Spirakis, P.G.: Traveling salesman problems in temporal graphs. In: Csuhaj-Varjú, E., Dietzfelbinger, M., Ésik, Z. (eds.) MFCS 2014, Part II. LNCS, vol. 8635, pp. 553–564. Springer, Heidelberg (2014)
26. Wagner, A., Lindenbaum, M., Bruckstein, A.: Distributed covering by ant-robots using evaporating traces. IEEE Trans. Robot. Autom. **15**(5), 918–933 (1999)
27. Xuan, B., Ferreira, A., Jarry, A.: Computing shortest, fastest, and foremost journeys in dynamic networks. IJ Found. Comput. Sci. **14**(02), 267–285 (2003)