Dynamic Control of Data Ferries under Partial Observations

Chi Harold Liu[†], Ting He[‡], Kang-won Lee[‡], Kin K. Leung[†], and Ananthram Swami[§]
[†]Imperial College, London, UK
[‡]IBM T. J. Watson Research Center, Hawthorne, NY, USA
[§]Army Research Laboratory, Adelphia, MD, USA
[†]{chiliu, kin.leung}@imperial.ac.uk, [‡]{the, kangwon}@us.ibm.com, [§]aswami@arl.army.mil

***Abstract*—Controlled mobile helper nodes called data ferries have recently been proposed to bridge communications between disconnected nodes in a delay-tolerant manner. While existing work has explored various trajectory designs for the data ferry by assuming either static nodes or full observations at the data ferry, the problem remains open when the nodes are mobile and the ferry only has partial observations. In this paper, we investigate the problem of dynamic ferry mobility control under limited-range sensing. Assuming the data ferries are capable of sensing node presence within certain range and adjust their movements dynamically, we aim to design control policies that maximize the number of effective contacts. We provide a comprehensive model of the control framework using Partially Observable Markov Decision Process (POMDP), based on which we study the structure of the optimal policy and propose an efficient heuristic policy which shows significant improvement over the predetermined benchmark. To the best of our knowledge, this is the first data ferry control mechanism that can handle both stochastic node mobility and incomplete ferry observations.**

I. Introduction

Continuing advances in sensor technologies and pervasive computing brings in new perspectives to solving challenging communication problems. Consider a network of mobile nodes moving in rough terrains (*e.g.,* with obstacles or danger zones), as illustrated in Fig. I. Due to terrain constraints and/or mission requirements, the network may be partitioned into smaller groups from time to time, and although communications within a group can be enabled via appropriate mobile ad hoc or delay-tolerant routing protocols, communications across groups can become difficult. Yet, such inter-partition communications are frequently needed for mission success. Applications of this kind can be found in military coalition networks, emergency response scenarios, and other networks in challenging environments. In such circumstances, helper nodes mounted on controllable mobile platforms such as UAVs [1] have been proposed to assist with the communications in a *load-carry-and-deliver* manner. If the sensing range of the data ferry covers the entire domain (defined as the activity area of one group), the ferry will have full knowledge of the network layout and the problem becomes straightforward [1], [2]. In practice, however, complete sensing coverage may not always be possible due to ground obstacles, vast network area, limitations of sensor, or simply because of the need of keeping the UAVs from being exposed to the adversary. In this paper, we study in detail how to bridge communications in such challenged scenarios using dynamically controlled, autonomous data ferries.

Research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

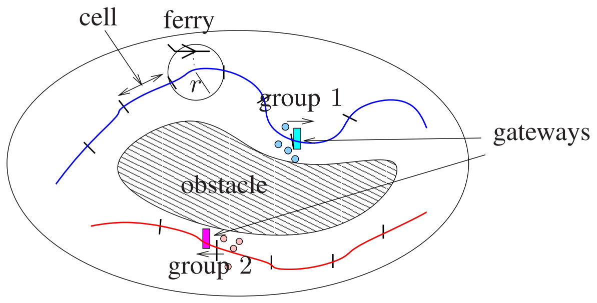

Fig. 1. Example: using a UAV-mounted data ferry to connect two groups of nodes (via gateways) moving along disjoint routes. r: ferry sensing radius.

Consider the case where the sensing range of a data ferry only covers a subset of the entire region (referred to as the *partially* observable case), and thus it does not know the exact locations of nodes once they are out of range. For control purpose, we partition the network region of each group of nodes into cells (in 1-D case, a cell refers to a segment as shown in Fig. I) such that the data ferry can sense and communicate with a node anywhere within the cell. Each data ferry is equipped with certain sensing, communications, and storage capabilities, and most importantly, with a programmable control logic which can navigate it among the cells. Periodically, the data ferry senses the presence of nodes and uploads/downloads messages upon contact, after which it will move to the next cell specified by the control logic and repeat the process. In the mean while, the nodes may move among cells of their local region constantly according to their mission needs. Assuming each group has a gateway node that aggregates all the inter-group messages, our goal is to control the mobility of the ferry to efficiently transport messages between gateways. In the sequel, we will only focus on the interaction between the ferry and the gateways, and all references to "nodes" mean such gateway nodes.

Although it is possible to infer certain statistical properties of node movements, it is often impractical to accurately predict their movements due to run-time randomness. The question we investigate is: how should the data ferry move given certain prior knowledge of node mobility patterns and its run-time (partial) observations? To our knowledge, this is the first effort in data ferry control to address both run-time randomness (at nodes) and incomplete observations (at the ferry)[1].

A. Summary of Contributions and Limitation

We are interested in the design of control policies for autonomous data ferries in DTNs. To address the challenges of run-time randomness and incomplete observations, we take the approach of dynamic control, where instead of designing fixed trajectories, we design control policies that dynamically map available information to navigation actions at run time. Our specific contributions are three-fold:

Comprehensive control framework: We develop a comprehensive framework for the design of control logic using the tool of Partially Observable Markov Decision Process (POMDP) [4]. The framework incorporates both the prior knowledge of node movements, modeled by Markov chains on the partitioned space, and the run-time sensing results, modeled by binary observations. It also allows for a flexible control objective, modeled by a payoff function. As a concrete example, we choose to maximize the number of effective ferry-node contacts with exponential discounts, although other objectives can also be used.

Efficient policy computation algorithm: Due to the well-known curse of history and dimensionality, POMDP problems are generally difficult to solve exactly. We address this issue by developing an efficient policy computation algorithm based on belief space quantization. Although the proposed algorithm contains design parameters that entail a performance-complexity tradeoff, our simulation shows that the performance of the computed policy is robust to these parameters, and it is possible to achieve a low complexity without compromising much performance.

Numerical studies: The proposed policy is evaluated by simulations under random walk mobility models. The results show that by adapting to node movements at run time, the proposed policy can achieve a higher contact rate than the predetermined alternative, and the improvement is more significant for nodes with random mobility patterns in a large field (*e.g.,* 100% improvement for symmetric mobility in fields with more than 10 cells; see Section V).

B. Related Work

Recently, the idea of using designated mobile nodes to support communications in poorly connected networks has emerged [2], [5]–[8]. If sufficiently many helper nodes are available, a mobile backbone can be constructed to cover all the regular nodes [7]. Otherwise, the helper nodes will have to physically move between regular nodes to transport messages, and proper control of their mobility becomes crucial to the overall communication performance. Depending on the type of communications, such helper nodes can act as mobile sinks in sensor networks [8] or data ferries in ad hoc networks [2], [5], [6]. Existing works on data ferry/mobile sink control assume that nodes are either static [2], [5], [8] or can notify the ferry of their current locations via long-range radio [6], in which cases the control problem is reduced to design predetermined routes for the ferry to follow. In contrast, we aim to relax these assumptions to handle the challenging scenarios of highly mobile nodes and limited ferry observations.

Technically, our problem belongs to the family of stochastic control problems with partial observations, first studied in [4]. Although extensively studied in the fields of control and robotics, to our knowledge its application on mobility control in communication networks has not been explored before.

The rest of the paper is organized as follows. Section II presents the control framework based on POMDP. Section III formulates the problem and presents the optimal control policy. Hardness results and an efficient alternative policy are presented in Section IV, which are evaluated numerically in Section V. Section VI concludes the paper.

II. Control Framework

A. Network Model

Given a number of disconnected network partitions, where each partition contains a group of nodes moving in a disjoint local region according to certain group mobility patterns, we select one node per group to perform as the gateway for communications across groups, as illustrated in Fig. I. We assume that nodes have sufficient contacts within a group, and the selected gateways have sufficient storage to buffer messages while awaiting contacts with the data ferry. In the sequel, all references to "node" mean such gateways.

Each data ferry is assigned to serve multiple domains, and the assignment is disjoint for different ferries[2], which makes it suffice to focus on one ferry. The ferry periodically senses node presence within a certain range, exchanges messages with nodes upon contacts, and buffers messages in between. Moreover, it has a controller which can dynamically navigate the ferry among cells.

The rest of the subsections specify the framework of control based on the concept of POMDP.

B. State Space and Mobility Model

To facilitate control, we partition the network field of each node into cells, labeled $0, \dots, n_i$ for node i, so that the data ferry is able to sense a node and exchange messages with it once they are in the same cell. In this paper, we keep

[1]Although some other works involving randomness and incomplete information exist in a similar context, *e.g.,* [3] studies node power management under incomplete knowledge of ferry schedules, they do not focus on controlling ferry mobility and are thus out of the scope of this paper.

[2]Coordinated service by multiple ferries per domain will be explored in future work.

a minimum state space where the controller represents the network state only by the cell index of the node it is trying to contact in the current slot to focus on the issue of stochastic movements and partial observations, *i.e.,* if the current node in slot t is node i, the state is $s_t \in \mathcal{S}_i = \{0, \dots, n_i\}$. Other network characteristics, such as traffic demands, quality of service requirements, and buffer size constraints, are also information of interest and will be explored in future work.

We model the mobility of each node by a Markov chain on the quantized space derived from the above partition. Let $P^i = (P^i_{jk})_{j,k\in\mathcal{S}_i}$ be the transition matrix for node i, where each element $P^i_{jk} \triangleq \Pr\{s_{t+1} = k | s_t = j\}$. For simplicity, we will assume i.i.d. mobility for nodes and drop the superscript i in the sequel, but the framework can be easily amended for heterogeneous cases.

C. Action Space

The action u_t specifies the cell the data ferry will move to in the coming slot, and the action space defines the set of movements feasible to the data ferry. In general, the action space is the union of all the cells nodes may visit, i.e., $u_t \in \bigcup_{\forall i} \mathcal{S}_i$, which may grow linearly with the number of nodes. To keep the policy scalable, we consider a hierarchical design where the action only specifies where to move in the region of the node the ferry is trying to contact, and the order of nodes is controlled by an upper layer policy. For example, one can use existing ferry route design algorithms [5] to obtain a node sequence that optimizes certain performance metrics.

Under such an hierarchical design, the action space is divided into two subsets: “follow” actions $\mathcal{U}_f = \{\text{“follow” } 0, \text{“follow” } 1, \dots, \text{“follow” } n\}$ and “switch” actions $\mathcal{U}_s = \{\text{“switch” } 0, \text{“switch” } 1, \dots, \text{“switch” } n\}$, where “follow” j means to navigate to cell j of the current region to follow the current node, and “switch” j means to switch to cell j of a new region specified by the upper layer policy. The reason to allow for switching is that occasionally, nodes may wander away from their usual cells, which will cause consecutive misses, and it may be more efficient to move on to other regions first and revisit this node later. In the case where skipping a node is undesirable (*e.g.,* messages have hard deadlines), one can simply remove the switch actions from the action space. The set of feasible policies allowing switching includes the set of policies not allowing switching, and thus the optimal performance of the former gives an upper bound on the latter. The overall action space is given by $\mathcal{U} = \mathcal{U}_f \cup \mathcal{U}_s$.

D. Observation Model

The onboard sensor produces a binary observation per slot. Let $z_t \in \mathcal{Z} = \{0, 1\}$ denote the observation in slot t, where $z_t = 1$ means “contact” and $z_t = 0$ means “miss”. Then under perfect sensing, $z_t = 1$ if and only if $u_{t-1} = s_t$, *i.e.,* the cell the ferry decides to navigate to (one slot earlier) coincides with the cell the node moves to. Note that in general, the data ferry may miss a node even if it is within the sensing range, which can be modeled by a randomized observation model $z_t = 1$ with certain probability if $u_{t-1} = s_t$.

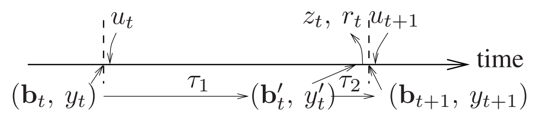

Fig. 2. The order of action, state transition, and observation during one slot (i.e., one sensing period).

E. Payoff Function

The payoff function represents the goal of control. Since the job of a data ferry is to ferry messages between different nodes, an intuitive goal is to maximize the total number of *effective contacts*, which is defined as the first contact after switching. Note that this effectively prevents the trivial case of being stuck with the same node since only the first of each run of consecutive contacts with the same node counts. Under this payoff, we note that it is not sufficient to only know whether the data ferry meets the targeted node or not, but we also need to know whether the next contact is effective or not. Let $y_t \in \{0, 1\}$ denote such an indicator, where $y_t = 1$ means effective contact and $y_t = 0$ otherwise. The payoff function $r(z_t, y_t)$ is given by $r(z_t, y_t) = z_t \cdot y_t$, which gives a unit reward if and only if there is an effective contact.

Based on the payoff function, we can calculate the cumulative payoff over a *design horizon* T, which is a period of time the controller is optimized upon. In this paper, we consider the discounted rewards: $R_T \triangleq \mathbb{E}[\sum_{t=1}^T \gamma^t r(z_t, y_t)]$, where $\gamma \in (0, 1]$ is a discount factor specified by the application.

III. Problem Statement and Optimal Policy

Given the framework developed in Section II, the problem is to design a control policy for dynamic mobility control. A policy π is a sequence of mappings that map all the available information, including past actions and observations, to a new action in each slot, i.e., $\pi = (\pi_t)_{t=1}^T$, and

$$\pi_t\big(\mathbf{z}_{1:t-1},\ \mathbf{u}_{1:t-1}\big) = u_t, \tag{1}$$

where we use $\mathbf{x}_{t_1:t_2}$ to denote the vector $(x_{t_1}, \dots, x_{t_2})$ ($t_1 \le t_2$). Denoting the expected payoff under a policy π by

$$R_T^\pi \triangleq \mathbb{E}\Big[\sum_{t=1}^T \gamma^t r(z_t,\ y_t)\big| u_t = \pi_t\big(\mathbf{z}_{1:t-1},\ \mathbf{u}_{1:t-1}\big)\Big], \tag{2}$$

where the expectation is taken over all possible node movements and observations, the problem is to find a policy π that maximizes R_T^π over all feasible policies for a predetermined horizon T. We are particularly interested in stationary policies, i.e., $\pi_t \equiv \pi$ for all t, which maximizes the long-term payoff as $T \to \infty$.

A. Belief Updates

It is known that the sufficient statistics for the past information is the *belief vector* (also called belief state) $\mathbf{b}_t$, which is the posterior distribution of the (current) node given all past observations, i.e., $\mathbf{b}_t = \big(\Pr\{s_t = j | \mathbf{z}_{1:t-1},\ \mathbf{u}_{1:t-1}\}\big)_{j\in\mathcal{S}}$. Enriched with the indicator y_t, we obtain a new state $(\mathbf{b}_t,\ y_t)$ which is the state the controller will act upon.

To understand the control process, let us zoom in to one slot starting with some initial state $(\mathbf{b}_t,\, y_t)$, as illustrated in Fig. III-A. At the beginning of the t^{th} slot, the data ferry chooses the next action based on its policy $u_t = \pi_t(\mathbf{b}_t,\, y_t)$ and moves accordingly. At the same time, the node also moves according to its own mobility pattern represented by state transition matrix P, and reaches a new belief state $P^T\mathbf{b}_t$ by the end of the slot. However, if the data ferry chooses to switch to a new node, then the "current node" changes, and its belief of the new node is reset to the limiting distribution $\mathbf{b}_0$ (assume it exists) while the indicator y_t is also reset to 1. Combining these two cases give the state transition specified by $(\mathbf{b}'_t,\, y'_t) = \tau_1(\mathbf{b}_t,\, y_t|u_t)$ as:

$$\tau_1(\mathbf{b}_t,\, y_t|u_t) = \begin{cases} (P^T\mathbf{b}_t,\, y_t) & \text{if } u_t \in \mathcal{U}_f, \\ (\mathbf{b}_0,\, 1) & \text{if } u_t \in \mathcal{U}_s. \end{cases} \tag{3}$$

At the end of the slot, the data ferry takes an observation z_t and earns a unit reward if effective contact occurs. The expected payoff is given by

$$r_t = r(\mathbf{b}'_t,\, y'_t,\, u_t) = y'_t b'_t(u_t). \tag{4}$$

In addition, the data ferry will update its belief vector according to the Bayesian rule. Let $\mathbf{e}_u$ denote the unit vector with 1 at the u^{th} element and 0 elsewhere, and $[\mathbf{b}'_t]_{\setminus u}$ the belief vector derived by setting the u^{th} element of $\mathbf{b}'_t$ to 0 followed by normalization. Then the Bayesian update $(\mathbf{b}''_t,\, y''_t) = \tau_2(\mathbf{b}'_t,\, y'_t|u_t,\, z_t)$ is given by:

$$\tau_2(\mathbf{b}'_t,\, y'_t|u_t,\, z_t) = \begin{cases} (\mathbf{e}_{u_t},\, 0) & \text{if } z_t = 1, \\ ([\mathbf{b}'_t]_{\setminus u_t},\, y'_t) & \text{if } z_t = 0. \end{cases} \tag{5}$$

The updated state is then used as the new state for the next slot $(\mathbf{b}_{t+1}, y_{t+1}) = (\mathbf{b}''_t,\, y''_t)$, and the process repeats. Note that instead of one belief update per step (which combines the Bayesian update at the end of a slot with the state transition in the next slot) as in classic POMDP, there are two updates in our problem because the update reflecting state transition (τ_1) depends on the next action and thus has to be deferred to the next slot.

B. Optimal Policy and Value Iteration

Let the *value function* $V_T(\mathbf{b},\, y)$ denote the cumulative payoff over horizon T starting from state $(\mathbf{b},\, y)$. Then the optimal value function must be the solution of the *value iteration* (VI) in the following form [4]:

$$\begin{aligned} V_T(\mathbf{b},\, y) \;&=\; \gamma \max_{u\in\mathcal{U}} \Big[r(\mathbf{b}',\, y',\, u) + \\ &\qquad \sum_z p(z|\mathbf{b}',\, u) V_{T-1}(\mathbf{b}'', y'') \Big], \end{aligned} \tag{6}$$

where $p(z=1|\mathbf{b}',u) = b'(u)$, and $p(z=0|\mathbf{b}',u) = 1-b'(u)$, and the optimal policy must be the one achieving the optimal value function, *i.e.,*

$$\begin{aligned} \pi_T(\mathbf{b},\, y) \;&=\; \arg\max_{u\in\mathcal{U}} \Big[r(\mathbf{b}',\, y',\, u) + \\ &\qquad \sum_z p(z|\mathbf{b}',\, u) V_{T-1}(\mathbf{b}'', y'') \Big]. \end{aligned} \tag{7}$$

For infinite horizon $T = \infty$, it is known that the VI will converge as long as $\gamma < 1$, and the limit V_∞ gives the optimal stationary policy that maximizes the long-term total (discounted) payoff. As stationary policies are easy to implement, and the lifetime of a data ferry is typically long relative to its sensing period, we are particularly interested in designing stationary policies with good payoff over large horizons.

IV. Hardness Result and Efficient Heuristic Policies

It is known that to compute π_T for arbitrary initial state is PSPACE-hard [9]. Without loss of generality, we assume the initial state is always $(\mathbf{b}_0,\, 1)$ (*i.e.,* the data ferry is aware of limiting distribution of node movement), then it can be shown that the complexity is reduced to $O(|\mathcal{U}|^T)$, where $|\mathcal{U}|$ denotes the number of actions[3]. Therefore, in order to compute the optimal policy, the main difficulty comes from the fact that there is a large number of reachable belief vectors which grows exponentially with T.

To speed up the policy computation, we adopt the method of grid-based value iteration [10] which limits VI (6) to only a subset of belief vectors called *grid points* that are sampled from the belief simplex. Based on this idea, we develop an approximate policy-computing algorithm shown in Algorithm 1. The algorithm tries to approximate the value function iteratively, where in each iteration (lines 2–16), it mimics the exact VI (6) at each grid point (lines 6–13) with the simplification that values at non-grid points are approximated by those at the nearest grid points (line 9)[4]. The iteration repeats until the approximated value function converges (line 16). Given the approximated values computed by Algorithm 1, we can extract the control action for any given belief by plugging the approximated values into (7), called *lookahead controller*. Alternatively, we can let the algorithm record the best action at each grid point (line 13) and directly use the action for the grid point closest to the current belief, called *direct controller* [10].

How well this algorithm performs depends on the selection of grid points $\mathcal{B}$. Ideally, $\mathcal{B}$ should represent all the reachable belief vectors. Nevertheless, since the number of reachable belief vectors grows exponentially over time, certain approximation is needed, and it is desirable to select only a small number of representative belief vectors as grid points. This process of representing the belief simplex by grid points introduces certain inaccuracy, which can be reduced by increasing the number of grid points $|\mathcal{B}|$ at the cost of increased complexity. Fortunately, we observe that the control performance is insensitive to $|\mathcal{B}|$ (see Fig. 7).

[3]This is because the number of reachable states grows as $|\mathcal{U}|^{T-1}$ when value function (6) iterates at each step, and we need to optimize value functions over $|\mathcal{U}|$ actions.

[4]We use 2-norm to calculate the distance between two belief vectors, but other norms can also be used.

Algorithm 1 : Approximate VI based on belief sampling

```
1:  Initialize: $T = 0$, $\hat{V}_T \leftarrow 0$
2:  repeat {approximate value iteration}
3:      $T \leftarrow T + 1$
4:      for all $\mathbf{b}_k \in \mathcal{B}$, where $k = 0, 1, \ldots, |\mathcal{B}| - 1$ do
5:          for all $y \in \{0, 1\}$ do
6:              for all $u \in \mathcal{U}$ do
7:                  compute $\mathbf{b}''_{k,z}$ in (5) for $z = 0$ and $z = 1$
8:                  if $\mathbf{b}''_{k,z} \notin \mathcal{B}$ then
9:                      $\hat{\mathbf{b}}''_{k,z} \leftarrow \arg\min_{\mathbf{b}_j \in \mathcal{B}} \|\mathbf{b}_j - \mathbf{b}''_{k,z}\|$
10:                 end if
11:                 $Q_T(\mathbf{b}_k, y, u) = r(\mathbf{b}'_k, y', u) + \sum_z p(z|\mathbf{b}'_k, y')\hat{V}_{T-1}\left(\hat{\mathbf{b}}''_{k,z}, y''\right)$
12:             end for
13:             $\hat{V}_T(\mathbf{b}_k, y) \leftarrow \max_{u \in \mathcal{U}} \gamma Q_T(\mathbf{b}_k, y, u)$
14:         end for
15:     end for
16: until $\max_{\mathbf{b}_k, y} |\hat{V}_T(\mathbf{b}_k, y) - \hat{V}_{T-1}(\mathbf{b}_k, y)| \leq \epsilon$
17: Return: policy $\pi_T$.
```

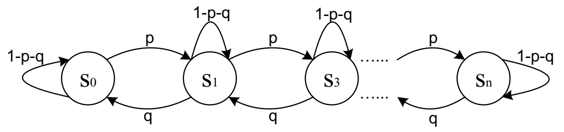

Fig. 3. The state transition diagram for the random walk mobility model used in the simulation.

V. Numerical Results

We assess the performance of the proposed controller by a case study. We simulate N nodes moving on disjoint 1-D paths, each of which is partitioned into n cells such that each cell can be covered by the sensing range of the data ferry in one time slot. Runtime randomness of the nodes' mobility is modeled by i.i.d. random walks as shown in Fig 3 with parameters p, q. The default parameters are $N = 3$, $n = 5$, $T = 20$, $p = q = 0.3$, $|\mathcal{B}| = 50$, $\gamma = 1$ unless otherwise noted.

We first illustrate our simulation settings by two example trajectories in Fig. 4, which shows how nodes move over time and how our controller navigates the ferry. We simulate two mobility patterns: (a) symmetric mobility ($p = q$) and (b) biased mobility ($p \neq q$). The symmetric mobility represents scenarios where the nodes move randomly without preference in either direction; the biased mobility, on the other hand, represents cases where nodes have a general direction of movement but also minor setbacks due to random events in the field. As the plot shows, our controller adapts ferry movements more aggressively in the symmetric case (Fig. 4 (a)) and deterministically hops between the most probable cells of different nodes in the biased case (Fig. 4 (b)), suggesting that the proposed controller is indeed able to adapt to different node mobility patterns.

We then simulate the proposed algorithm under different design parameters. Fig. 5–6 shows the performance over time under different discount factor γ. Although the value

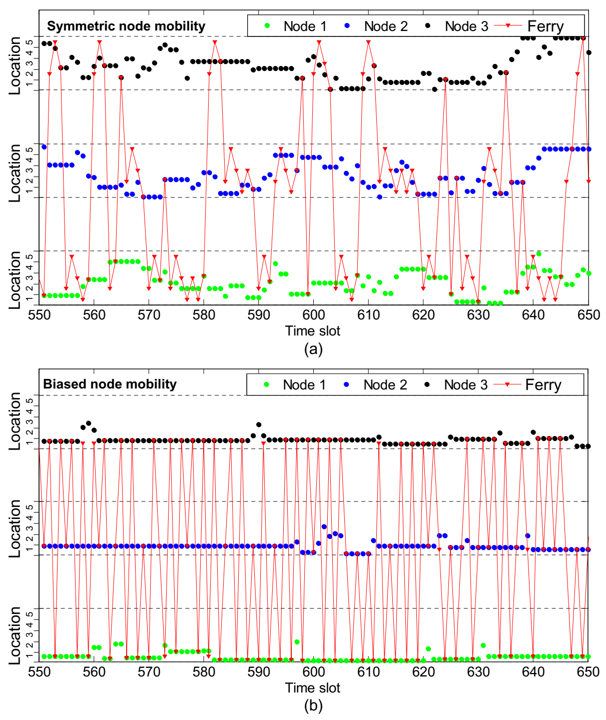

Fig. 4. Two illustrative examples of node and ferry trajectories. Two different mobility patterns: (a) $p = q = 0.3$ and (b) $p = 0.1, q = 0.8$.

of γ has immediate impact on the convergence rate and the value of discounted reward (Fig. 5), we observe that the actual number of effective contacts is rather insensitive to γ. The implication of this result is that one can afford to use a small γ for fast convergence without compromising the actual ferry performance. Fig. 7 shows the effect of using different types of controllers and different numbers of grid points. We simulate two types of controllers: direct controller and lookahead controller (as explained in Section IV). As a benchmark, we also simulate a predetermined policy that lets the ferry keep switching among the most likely cells of different nodes, called the *switching policy*. While the direct controller can be improved by selecting more grid points, the lookahead controller appears indifferent to varying $|\mathcal{B}|$. Among the three controllers, the lookahead controller outperforms the direct controller, especially at small grids, and they both significantly outperform the deterministic controller using the switching policy. This result implies that when using lookahead controller, we can afford to use a small grid for fast computation without compromising the performance. All the other simulations use lookahead controller. We generate grid points by randomly sampling the belief simplex; it is possible to use other methods (see [10]), but we do not observe significant differences in the results.

To evaluate the gain of dynamic control, we compare the proposed policy with the switching policy in Fig. 8–9. Fig. 8 shows that the proposed policy achieves a larger number of contacts than the switching policy, and the gap increases over time, *i.e.,* the contact rate is larger. The amount of improvement, however, depends on the specific mobility parameters, and we observe that it is larger for symmetric mobility and

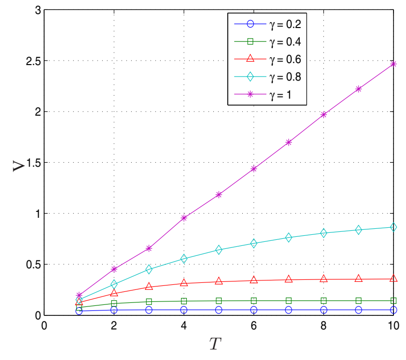

Fig. 5. Effect of γ: discounted reward over time.

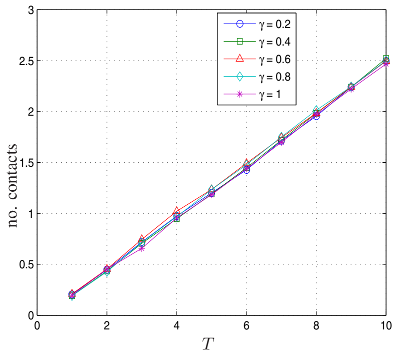

Fig. 6. Effect of γ: number of contacts over time.

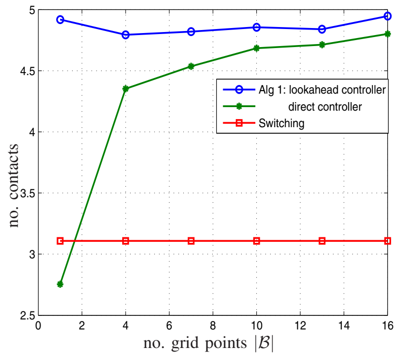

Fig. 7. Effect of grid points.

becomes smaller as the mobility becomes more biased (*e.g.,* our policy has 68% more contacts for $p = q = 0.3$ but only 8% more for $p = 0.2$, $q = 0.4$). Fig. 9 shows the performance of both policies as the number of cells n increases. The increase of n models either a smaller sensing range of the data ferry for a fixed network area or a larger network area for a fixed sensing range. In either case, it becomes more difficult for the ferry to meet nodes as the level of uncertainty increases, and indeed the numbers of contacts decrease for both policies. Our policy shows a constant improvement over the switching policy for all values of n under symmetric mobility. The improvement diminishes as the node mobility becomes biased because in this case, the ferry will directly meet the node at its most likely cell with high probability, and thus there is no need of sophisticated control. In the symmetric case, although the absolute amount of improvement is not large, the relative improvement can be significant for large n (*e.g.,* 100% at $n = 12$).

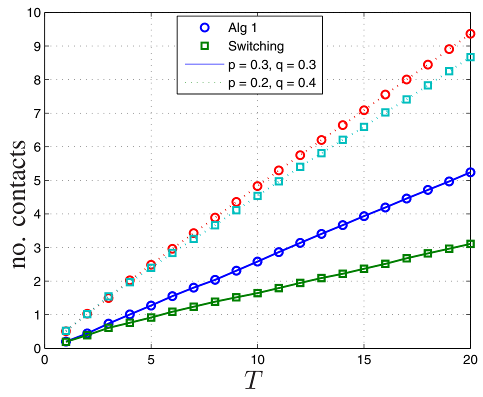

Fig. 8. Proposed policy vs. switching policy: number of contacts over time.

VI. Conclusions

In this paper, we revisit the problem of data ferry control under the new challenges of stochastic node movements and incomplete ferry observations. To handle these challenges in a systematic manner, we explore the approach of stochastic control using POMDP. We propose a comprehensive model of the control framework, based on which we develop both a mathematical representation of the optimal control policy and an algorithm to approximate it efficiently. Initial simulations show promising improvements in the contact rate compared with predetermined control mechanism, especially for nodes with (close to) uniform steady-state distribution and relatively large fields.

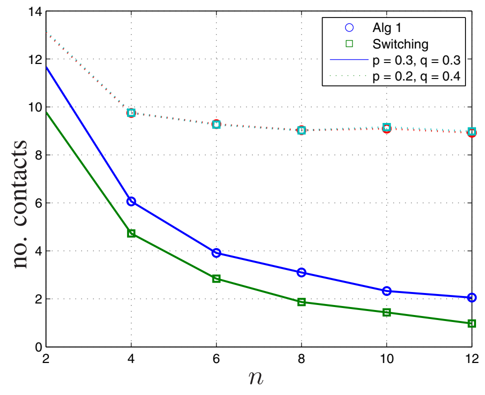

Fig. 9. Proposed policy vs. switching policy: number of contacts over number of cells.

References

[1] T. X. Brown, B. Argrow, C. Dixon, S. Doshi, R.-G. Thekkekunnel, and D. Henkel, "Ad hoc uav-ground network (augnet)," in *AIAA 3rd Unmanned Unlimited Technical Conference*, Chicago, IL, Sept. 2004.

[2] D. Henkel and T. Brown, "On controlled node mobility in delay-tolerant networks of unmannned aerial vehicles," in *International Symposium on Advance Radio Technologies*, 2006.

[3] H. Jun, W. Zhao, M. Ammar, E. Zegura, and C. Lee, "Trading Latency for Energy in Densely Deployed Wireless Ad Hoc Networks Using Message Ferrying," *Ad Hoc Networks*, vol. 5, no. 4, pp. 444–461, May 2007.

[4] E. Sondik, "The optimal control of partially observable markov decision processes," Ph.D. dissertation, Stanford University, CA, 1971.

[5] W. Zhao, M. Ammar, and E. Zegura, "Controlling the mobility of multiple data transport ferries in a delay-tolerant network," in 24*th IEEE INFOCOM 2005*, Miami, FL, March.

[6] ——, "A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks," in *ACM MobiHoc*, Roppongi, Japan, May 2004.

[7] A. Srinivas, G. Zussman, and E. Modiano, "Mobile backbone netowkrs—construction and maintenance," in 7*th ACM MobiHoc 2006*, Florence, Italy, May.

[8] M. Ma and Y. Yang, "Sencar: An energy efficient data gathering mechanism for large scale multihop sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, no. 10, October 2007.

[9] C. Papadimitriou and J. Tsitsiklis, "The complexity of markov decision processes," *Mathematics of Operations Res.*, no. 3, pp. 441–450, 1987.

[10] M. Hauskrecht, "Value-function approximations for partially observable markov decision processes," *Journal of Artificial Interlligence Research*, pp. 33–94, 2000.