Self-monitoring Reinforcement Metalearning for Energy Conservation in Data-ferried Sensor Networks

Ben Pearre and Timothy X. Brown
University of Colorado, Boulder, CO, USA
{benjamin.pearre,timxb}@colorado.edu

***Abstract*—Given multiple widespread stationary data sources such as ground-based sensors, an unmanned aircraft can fly over the sensors and retrieve their data via a wireless link. When sensors have limited energy resources, they can reduce the energy used in data transmission if the ferry aircraft is allowed to extend its flight time. Complex vehicle and communication dynamics and imperfect knowledge of the environment confound planning since accurate system models are difficult to acquire and maintain, so we present a reinforcement learning approach that allows the ferry aircraft to optimise data collection trajectories and sensor energy use *in situ*, obviating the need for system identification. We address a key problem of reinforcement learning—the high cost of acquiring sufficient experience—by introducing a metalearner that transfers knowledge between tasks, thereby reducing the number of flights required and the frequency of significantly suboptimal flights. The metalearner monitors the quality of its own output in order to ensure that its recommendations are used only when they are likely to be beneficial. We find that allowing the ferry aircraft to double its range can reduce sensor radio transmission energy by 60% or better, depending on the accuracy of the aircraft's information about sensor locations.**

***Keywords*-Sensor networks; data ferries; energy optimisation; reinforcement learning; metalearning**

I. INTRODUCTION

We consider the problem of collecting data from widespread energy-limited stationary data sources such as ground-based sensors. Our approach uses a fixed-wing unmanned aircraft (UA) to fly over the sensors and gather the data via a wireless link [1]. We assume that the UA has a known range limit and can be recharged/refuelled at a base station, and that the sensors may continuously generate data over long periods, so that the UA needs to ferry the data to a collection site over repeated flights. The goal is to trade energy used by the UA against energy saved by the sensor nodes. The system is difficult to model, so the challenge is to develop a model-free approach that can quickly learn to minimise the sensors' radio transmission energy subject to the UA's range constraints.

The problem may be subdivided into the following pieces: *Aircraft trajectory optimisation* seeks to discover a flight path over the sensor nodes (a so-called *tour*) that minimises some mission cost. We decompose this piece as follows:

- *Tour Design* decides in what order to visit sensor nodes of known location, or establishes a search pattern when the locations are unknown.
- *Trajectory Optimisation* finds a sequence of waypoints the UA should follow in order to visit the sensor nodes.
- *Vehicle Control* translates the waypoints into control surface and engine commands.

Radio energy optimisation consists of the following:

- *Radio Design* chooses radio hardware and protocols to support high-efficiency communication.
- *Power Management* varies the transmission power of nodes' radios during interaction with the ferry aircraft.

This paper focuses on the Aircraft Trajectory Optimisation and Radio Power Management layers. We assume that the tour is given and that the nodes' locations are known only approximately as when, for example, the sensors have been deployed from an aircraft. Vehicle control to track a set of waypoints requires an autopilot, whose behaviour is a complex function of the waypoints, weather, aircraft dynamics, and the control models within the autopilot. Similarly, communication system performance is a complex function of the radio protocols, antenna patterns, noise, and interference. We assume autopilot and communication systems are black boxes whose specific functionality is unknown to the upper layers. Only aggregate performance of the ferry system is reported to the learner.

In [2], we examined model-free minimisation of UA trajectory length. Here we extend the technique: since network lifetime or maintenance costs may depend on the energy reserves of the sensor nodes, we seek to minimise their transmission energy cost per bit.

Data ferries can be highly effective for reducing radio energy requirements. Jun et al. [3] compare ferry-assisted networks with hopping networks in simulation and finds that a ferry can reduce node energy consumption by up to 95% (further gains would have been possible with a broader configuration space). Tekdas et al. [4] reach a similar conclusion on a real toy network in which wheeled robots represent ferries. Anastasi et al. [5] consider the total energy requirement per message including overhead associated with turning a node's radio on in order to search for a fixed-trajectory ferry. Ma and Yang [6] optimise the lifetime of nodes by choosing between multi-hop node-to-node routing and changing the ferry's route and speed. Optimal solutions under the trade-off between energy use and latency have been examined for

 ISBN: 978-1-61208-207-3

a fixed ferry trajectory [7], and adaptively with a known radio model and simple flight dynamics [8]. In [9], a node learns whether to transmit to the ferry or wait depending on the anticipated trajectory; range affects transmission power. Anastasi et al. [10] review techniques for energy minimisation for both ferried and general sensor networks.

Past work on ferried networks has made various assumptions about ferry and communication system dynamics, which we broadly categorise as follows. In *Visit* models, the ferry exchanges all data upon visiting a node [11]–[14]. *Communication radius* models exchange all data instantaneously below a certain range [4], [6], [15]–[18]. A learning variant to this model is found in [19]: planning assumes a communication radius, but the ferry also exchanges data opportunistically, possibly allowing improvements to the trajectory. *Variable rate* models base rate on communication range [20], [21]. Stachura et al. [22] assume probabilistic packet loss based on distance, achieving the same effect. Mobile nodes are treated in [23], which uses a model of a UA equipped with a beamforming antenna to plan trajectories that maximise the signal-to-noise ratio (SNR) to each node.

Models are necessarily approximate, but inaccuracies can lead to poor performance in the field. The difficulty of generating and updating sufficiently accurate models under possibly changing conditions motivates our question: can a sensor network system simultaneously learn to optimise both the aircraft's flight path and the sensors' radio policies, in a reasonable time, directly on a radio field? The goal is to provide a UA with approximate information about the geometry of a sensor network, and to have it improve its performance rapidly and autonomously. We focus on minimising the number of flight tours required to get a good solution, since computational costs are comparatively minor.

Our contributions are as follows:

- We demonstrate the feasibility of a reinforcement learning approach for rapid discovery of energy-saving network policies that trade UA flight time for sensor energy. The policies are learned without a system model and despite potentially inaccurate sensor node locations, unknown radio antenna patterns, and ignorance of the internals of the autopilot.
- We introduce a reinforcement metalearner that learns to speed up and stabilise the performance of the learner, and transfers acquired knowledge about the policy optimisation process to unseen problems.
- A poorly trained metalearner may degrade the learner's performance, and should not be allowed to influence learning. We introduce a method of monitoring the performance of the metalearner without allowing it to affect network behaviour before it has been trained sufficiently.
- We show our two independent optimisers—waypoint location and transmission power policy—can operate simultaneously on the same sampling flights.

Our learning framework quickly produces trajectories that exploit the limits of ferry endurance. For example, when the allowable flight length is twice that of a handcoded reference trajectory and accurate sensor location information is available, the system learns to reduce sensor communication energy by roughly 60% after a few dozen flights, and when the sensor location information is approximate, the learners do even better. More important than this specific result is the development of a general technique that allows data-ferry networks to efficiently optimise their performance *in situ*.

Section II describes our radio model. Section III describes how our simulated autopilot control policies interpret waypoints. Section IV reviews the learning algorithm we use and describes how we apply it to our sensor energy optimisation problem. In Section V, we develop our metalearner. Section VI presents our results. Section VII concludes.

II. Radio environment

Our goal is to evaluate the use of model-free optimisation in a complex, unknown radio environment. We introduce a radio model that incorporates several complicating factors that are rarely considered: variable-rate transmission, point noise sources, and directional antennas. This model extends that introduced in our previous work [2] only by giving the nodes dipole antennas.

The signal to noise ratio at node a from node b is given by

$$\mathrm{SNR}_{ab} = \frac{P(a,b)}{N + \sum_i P(a,n_i)} \tag{1}$$

$P(a,b)$ is the power received by node a from node b, N is background noise from electronics and environment, and n_i are noise sources. The power between a and b is usually computed as

$$P(a,b) = \frac{P_{0,a} d_0^{\epsilon}}{|X_a - X_b|^{\epsilon}} \tag{2}$$

for reference transmit power $P_{0,a}$, reference distance $d_0 = 1$, distance between transmitter and receiver $|X_a - X_b|$, and propagation decay ϵ. However, antenna shape and radio interactions with nearby objects make most antennas directional, so the orientations of the antennas affect power. We model the aircraft's antenna as a short dipole with gain $1.5 = 1.76$ dBi oriented on the dorsal-ventral axis of the aircraft, yielding a toroidal antenna pattern. We model the nodes' fields similarly with random fixed orientations, so we adjust the power computation in (2) to:

$$P'(a,b) = \sin^2(\xi_{ab}) \sin^2(\xi_{ba}) P(a,b) \tag{3}$$

where ξ_{xy} is the angle between antenna x's pole and the direction to y. This depends on the vector between the UA's and node's positions ($\in \mathbb{R}^3$), the aircraft's orientation ($\in \mathbb{R}^3$), and the transmitter's orientation, although the latter is assumed not to change. Here we consider only constant-altitude trajectories.

In order to evaluate (3), we require the UA's position and orientation. A full dynamical simulation of the aircraft is unnecessarily complex for our purposes, so we assume that course and heading ϕ are the same (yaw = 0), pitch = 0, and roll $\psi = \frac{\pi}{2} \tanh 2\dot{\phi}$, which varies between 0 for a straight course and $\pm 54°$ for our maximum turning rate of $\dot{\phi} \simeq 0.347$ rad/s (i.e. the UA flies a complete circle in 20s).

 ISBN: 978-1-61208-207-3

We use the Shannon-Hartley law [24] to compute the data transmission rate between transmitter a and receiver b:

$$R_{ab} = \beta \log_2(1 + \mathrm{SNR}_{ab}) \quad (4)$$

This assumes that data rate varies continuously. The hardware may use discrete rates that are chosen according to current SNR conditions, but [21] indicates that the difference in trajectories and performance outcomes between continuously variable and the discrete rates of 802.11g may be negligible for this type of problem.

This model ignores many characteristics of a real radio environment such as occlusion, reflection, higher-order directionality, and multipath propagation. Moreover, we do not simulate obvious protocol modifications that would allow other sensor nodes to cease transmission and thereby reduce interference with the active node. However, in part due to the latter omission, the model produces fields that have irregularities similar to those that occur in real radio environments, and thus it meets our aim of having a complex simulation environment within which we can test whether the aircraft can learn *in situ.*

III. Autopilots

The aircraft is directed to follow some trajectory through an autopilot control policy.

A. Reference autopilot control policy

The *Reference autopilot* is borrowed from [21] and does not learn. It assumes that the aircraft has an estimate of the locations of the sensor nodes (although these can be difficult to discover [25]). While communicating only with the target node the aircraft flies at constant speed v towards the tangent of a circle of minimum turning radius about the node's nominal location, circles it at the maximum turning rate ω until D bytes are received, and then proceeds to the next node. The result is the Reference trajectory.

B. Learning autopilot control policy

The *learning autopilot* places a GPS waypoint at the assumed location of each sensor node (we will assume that node identities and approximate locations are known during tour initialisation, although the assignment could instead occur on the fly as nodes are discovered). The aircraft collects data from any node opportunistically while flying towards the tangent of a minimum-turning-radius circle about the waypoint, and then if necessary circles the waypoint exchanging data only with the assigned node until it has collected sufficient data. The true node location and the waypoint location may differ; final waypoint positions are learned as described next.

IV. Learning

Policy gradient reinforcement learning (PGRL) [26] consists of a family of techniques for model-free optimisation of control policies. Key elements are a policy gradient estimator, a reward function, and a policy representation. We introduce these by describing and applying the well-known "episodic REINFORCE" estimator to our waypoint-placement problem, and then introduce the more sophisticated radio transmission power policy.

A. Policy Gradient Reinforcement Learning (PGRL)

A policy $\pi(s, u; \theta) = \Pr(u|s; \theta)$ defines the probability of choosing action u given state s under the policy parametrised by $\theta \in \mathbb{R}^n$. The expected reward averaged over all states and actions under a policy $\pi(s, u; \theta)$ is denoted $J(\pi(s, u; \theta))$ or abbreviated as $J(\theta)$. PGRL techniques are ways of estimating the gradient of the expected reward: $\widehat{g_\theta} = \widehat{\nabla_\theta J(\theta)}$.

Our task can be broken down into distinct "trials", each consisting of a complete execution of $\pi(\theta)$ over some bounded time (e.g., the aircraft flying a complete tour τ) and receiving reward r at the end. During a trial, the policy defines a probability distribution over the action chosen at any point. Assume that the controller makes some finite number H of decisions u_k at times $t_k, k \in 1 \ldots H$ during a trial; discretising time in this manner makes it possible to compute the probability of a trajectory under a policy as the product of the probabilities of each (independent) decision at each time t_k. So $\Pr(\tau|\theta) = \prod_{k=1}^{H} \Pr(u_k|s_k; \theta)$.

To optimise θ, we estimate the gradient using stochastic optimisation's "likelihood-ratio trick" [27] or reinforcement learning's "episodic REINFORCE" (eR) [26], [28] with non-discounted reward. Each element ∇_{θ_i} of the gradient is estimated as:

$$\widehat{g_{\theta_i}} = \left\langle \left(\sum_{k=1}^{H} \nabla_{\theta_i} \log \Pr(u_k|s_k; \theta) - \mu_{\sum_\nabla} \right) \left(\sum_{k=1}^{H} \gamma^{t_k} r_k - b_i \right) \right\rangle \quad (5)$$

where $\mu_{\sum_\nabla}$ is the mean over trials of the $\sum \nabla_{\theta_i}$ terms, b_i is a "reward baseline" for element θ_i, computed as the inter-trial weighted mean of rewards, using the per-trial weight $\left(\sum_{k=0}^{H} \nabla_{\theta_i} \log \Pr(u_k|s_k; \theta)\right)^2$ [26], and $\langle \cdot \rangle$ is the average over some number N of trajectories. The term $\mu_{\sum_\nabla}$ is the mean over trials of the $\sum \nabla_{\theta_i}$ terms; this term does not appear in [26] but is included here in order to reduce the variance of the "characteristic eligibility"; we found it to further improve the gradient estimation process. We postpone discussion of the temporal discount factor γ to §V-C; for now consider $\gamma = 1$.

Once a policy gradient estimate $\widehat{g_\theta} = \widehat{\nabla_\theta J}$ for episode e is obtained, we take a step of some length α in that direction,

$$\theta_{e+1} = \theta_e + \alpha \frac{\widehat{g_\theta}}{|\widehat{g_\theta}|} \quad (6)$$

thus creating a new policy. The gradient estimation and update may be repeated until a design requirement is met, until convergence to a local maximum, or forever to track a time-varying environment. The theoretical guarantee of convergence to a locally optimal policy is only available if α decreases over time. That guarantee is useful, but does not directly apply to learners operating in a changing environment and is unnecessary for our tasks.

B. Reward

The reward function (or its additive inverse, the cost function) expresses the desiderata of solutions as a scalar quantity.

 ISBN: 978-1-61208-207-3

We seek the policy that leads to the lowest total transmission power subject to a maximum desired tour length $d_{\max}$, which indicates that we are nearing the endurance limit of the aircraft. We use the following:

$$r = -\left(\max(0, d - d_{\max})^{\varrho} + \sum_{j \in \text{nodes}} \varphi_j \sum_{k=1}^{H} P_{jk} \Delta t\right) \quad (7)$$

d is the current trajectory path length, $d_{\max}$ is the soft maximum range of the aircraft, ϱ controls the severity of the soft maximum distance penalty, P_{jk} is the transmission power of node j at timestep k of length Δt, and φ_j is a weighting for the value of energy for node j. Note that d is not penalised until the aircraft exceeds $d_{\max}$.

The aircraft's autopilot may be programmed to return to base by some hard length constraint $d_{\text{hardmax}} > d_{\max}$, and a cost for underrunning data-collection objectives could be included in (7). This produces similar results in our simulations but complicates our explanation.

C. Waypoint placement

We consider a sequence of nodes that need to be visited in some order $\{a, b_1, \ldots, b_n, c\}$ that was determined by a higher-level planner [13], [17]. The aircraft must fly a trajectory that starts at a and ends at c and allows exchange of D_j bytes of data with each of the n sensor nodes b_1 to b_n. Thus we seek the path $a \to c$ that minimises (7). That sufficient data are collected from each node is guaranteed by the autopilot policies (§III).

Trajectory policies for the autopilot are implemented as sequences of constant-altitude waypoints. So for m waypoints, the waypoint policy's parameter vector $\theta = [x_1\ y_1\ x_2\ y_2 \ldots x_m\ y_m]^T$. In order to be used by (5) the controller adds noise such that $\Pr(\tau|\theta)$ can be computed. In a real system, actuator noise or autopilot error E can be used for this purpose if $\nabla_\theta \log \Pr(u + E \mid \theta)$ can be computed, but in our simulations we simply add Gaussian noise directly to the waypoint locations at the beginning of each tour:

$$u \sim \mathcal{N}(\theta, \Sigma) \quad (8)$$

$$\nabla_\theta \log \Pr(u|s;\theta) = \frac{1}{2}\left(\Sigma^{-1} + \Sigma^{-1\,T}\right)(u - \theta) \quad (9)$$

D. Radio transmission power

The data-ferrying approach allows sensors to communicate with distal base stations without the need for high-powered radios, but the energy that nodes spend in communicating with the ferry is still non-negligible [3], [4]. Recall the data rate from Section II:

$$R_{ab} = \beta \log_2(1 + \text{SNR}_{ab}) \quad (10)$$

The derivative of $\frac{\text{rate}}{\text{power}}$

$$\nabla_P \frac{\beta}{P} \log_2\left(1 + \frac{P}{N}\right) = \frac{\beta}{N\,P\,\log(2)\,\left(1 + \frac{P}{N}\right)} - \frac{\beta \log\left(1 + \frac{P}{N}\right)}{P^2 \log(2)} \quad (11)$$

is negative whenever

$$\frac{P}{P+N} < \log\left(\frac{P+N}{N}\right) \quad (12)$$

which is true except at $P = 0$. So while reducing power results in a lower energy cost per bit, lower transmission rates require longer trajectories. Given an externally defined trade-off between ferry trajectory length and node energy savings, when should a sensor transmit, and at what power?

The difficulty of predicting the SNR between transmitter and aircraft again suggests reinforcement learning. We assume that at each timestep a sensor can transmit with power $P \in [0, P_{\max}]$, that it occasionally sends short probe packets at $P = P_{\max}$, and that the aircraft's radio can use this to measure the current maximum SNR and provide instructions to the node. These packets are too brief to transmit sensor data or use much power, so we do not model them explicitly. Here, too, the learning approach will silently optimise around such quirks of real hardware.

The *power policy* is a learned function that controls the power a node uses to transmit given a reported SNR, given in dB. Our desired behaviour is to transmit at a target power $P_{\text{target}} \le P_{\max}$ whenever the probed SNR exceeds some threshold R_T. So, the policy has action $u = P$, state $s = \text{SNR}_{\text{probed}}$, and is parametrised by $\theta = [P_{\text{target}}, R_T]$. Reinforcement learning requires exploration noise, so at each timestep the policy π draws the actual transmission power P from a Gaussian (truncated on $[0, P_{\max}]$) whose mean is taken from a sigmoid of height P_{target}:

$$\begin{aligned} \pi(s, u; \theta) &= \Pr(u|s;\theta) \\ &\sim \mathcal{N}\left(\frac{P_{\text{target}}}{1 + e^{\phi(R_T - s)}},\ \sigma\right), \text{ truncated on } [0, P_{\max}] \end{aligned} \quad (13)$$

When $s = R_T$, the mean transmit power is 50% of P_{target}, going to 100% as s increases above R_T and vice versa, thus implementing the desired behaviour with exploration. ϕ controls the sigmoid's width, and σ controls Gaussian exploration. For example, when $P_{\text{target}} = P_{\max}$ and R_T and σ are small, the policy mimics the full-power Reference policy.

The policy's derivatives are:

$$\nabla_\theta \log \pi(s, u; \theta) = \begin{bmatrix} \frac{u - \frac{P_{\text{target}}}{1 + \mathrm{e}^{\phi(R_T - s)}}}{\sigma^2\left(1 + \mathrm{e}^{\phi(R_T - s)}\right)} \\ -\frac{P_{\text{target}}\,\phi\,\mathrm{e}^{\phi(R_T - s)}\left(u - \frac{P_{\text{target}}}{1 + \mathrm{e}^{\phi(R_T - s)}}\right)}{\sigma^2\left(1 + \mathrm{e}^{\phi(R_T - s)}\right)^2} \end{bmatrix} \quad (14)$$

Unlike the waypoint-placement policy, this one is closed-loop: sensing packets detect SNR, which informs the choice of action u (transmission power) at each timestep. Thus we use the full loop-closing capabilities of the episodic REINFORCE algorithm of §IV-A. This policy and the waypoint-placement one run in parallel, using the same flights to estimate their reward gradients.

As a concession to practicality, the power policy incorporates a failsafe mechanism: when the aircraft's soft maximum

 ISBN: 978-1-61208-207-3

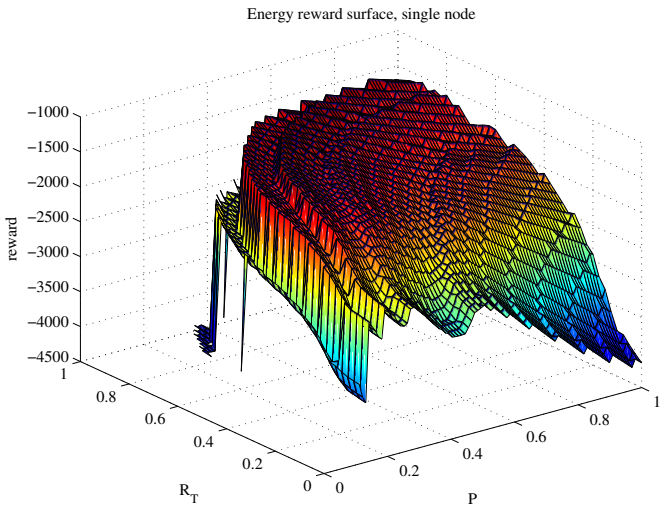

Fig. 1. Energy reward landscape for an example single node, with fixed waypoint position. As transmission power P and threshold SNR R_T change, energy savings may lead to greater reward up to a certain point. But the high cost of exceeding the aircraft's range constraint creates a steep "cliff" in the reward landscape.

range has been exceeded, radio power is set to 100%, ensuring that the UA does not become stuck in nearly infinite loops.

V. METALEARNING

Waypoint location optimisation is fairly straightforward [2]. However, our energy reward function (7) is highly nonlinear with respect to the power policy parameters in the vicinity of the optimal solution. Figure 1 shows a portion of the reward landscape for trajectories looping a typical node. When R_T is small and P is near 1, the gradient is not difficult to estimate—exploration noise will generally average over the ridges and valleys. However, in the vicinity of the optimal solution (the crest of the hill), there is a steep cliff: if R_T becomes too high or P becomes too low and the aircraft must remain near the sensor for a long time in order to collect all the data, which activates the aggressive length-overrun term of the reward function.

Conventional PGRL repeatedly estimates the reward gradient near the current policy and takes a hillclimbing step. Near the optimum, hillclimbing updates can result in the learner taking a step off the cliff, or "cliff-jumping". Furthermore, the cliff contains local regions in which a problematic reverse-sloped ledge structure is apparent—it is possible for a local gradient estimate to suggest a step further off the cliff. Confidence regions can mitigate this problem, but they generally fail to re-use information acquired during past steps (but see [29] for a counterexample). The problematic structure in the reward landscape motivated the development of a technique to encode knowledge of the process of optimising on reward landscapes like ours.

Consider the intuition: when a trial leads to an overly long trajectory, it is generally helpful to increase power or allow transmission at lower SNR. Conversely, for a trajectory that does not use the aircraft's full range node energy can be reduced by using lower power. The PGRL gradient estimation finds policy updates that, on average, tend to obey these heuristics, but the microstructure and the abrupt cliff near the global optimum frequently lead to poor updates. Our goal in developing a metalearner is to give the UA the ability to use experience with past problems to improve learning speed and robustness on new problems and automatically capture such heuristics. We investigate the following questions:

- Can a metapolicy that encodes knowledge about optimising policies in this domain be learned through experience?
- Can such a metapolicy transfer knowledge between problems?
- Can we monitor the quality of the metapolicy's recommendations in order to prevent a poor metapolicy from adversely affecting the optimisation process?
- Can the metapolicy be used to speed or stabilise the learning of energy-saving policies for sensor networks?

A. Metapolicy

Our energy "metapolicy" examines each trajectory and produces a guess as to the best update, $\Delta\theta$, to the base power policy's parameters $\theta = [P_{\text{target}}, R_T]$. For each element θ_i of θ, we use a simple neural network, a so-called single-layer perceptron (see [30]) with one input—the fraction of allowed aircraft range used—and two outputs—suggested changes to the base policy's two parameters:

$$s_\mu = \frac{d}{d_{\max}} \tag{15}$$

$$u_\mu = \Delta\theta_i \tag{16}$$

$$\pi_\mu(s_\mu, u_\mu; \Theta_i) = \Pr(u_\mu | s_\mu; \Theta) \tag{17}$$

$$= \mathcal{N}\left(\tanh\left(\Theta_{1,i} s_\mu + \Theta_{2,i}\right), \quad \sigma_\mu\right)$$

$$\nabla_{\Theta_i} \log \pi(s_\mu, u_\mu; \Theta) = \begin{bmatrix} s_\mu Z \\ Z \end{bmatrix} \tag{18}$$

$$\text{where } Z = \tag{19}$$

$$\sigma_\mu^{-2}\left(u_\mu - \tanh\left(s_\mu \Theta_{1,i} + \Theta_{2,i}\right)\right)\left(1 - \tanh(s_\mu \Theta_{1,i} + \Theta_{2,i})^2\right)$$

If the perceptual space is enriched with other inputs or requires a richer representation, other models can be used. Note, however, that more complex models with more parameters increases the number of runs necessary for learning a good metapolicy.

B. Metareward

The metalearner's objective is to learn a metapolicy that takes as input a policy operating on a trajectory, and outputs an "action" consisting of an improvement of the base policy's parameters. So as our metareward r_μ we choose the reward improvement between trials:

$$r_\mu = r_i - r_{i-1} \tag{20}$$

where r_i is the base reward received on trajectory i.

 ISBN: 978-1-61208-207-3

C. Time-discounted credit

The metalearner receives μ-reward (20) after every μ-action, and each μ-action also—to a lesser extent—affects future μ-states and thus potential μ-rewards, so it would be appropriate to use a time-discounted eligibility ($\gamma < 1$ in (5)). But further improvements are to be gained by using a more sophisticated gradient estimator, which we introduce here:

1) G(PO)MDP: (Here we drop the μ-prefix, as this section describes a well-known general technique.) In reinforcement learning, when an action u is taken at time t_u and a reward r is received at future time t_r, the action is assigned *credit* for the reward based on an estimate of how important the action was in producing the reward. In eR (§IV-A), greater weight may be given to rewards received early in the episode than on those received later, modulated by the term $\gamma^{t_k},\ 0 < \gamma \leq 1$ in (5). G(PO)MDP [31] uses the separation in time between t_u and t_r to assign credit in proportion to $\gamma^{t_r - t_u},\ t_u < t_r$. We use G(PO)MDP as described in [26]. The gradient estimator is related to (5):

$$\widehat{g_{\theta_i}} = \left\langle \sum_{p=0}^{H} \left(\sum_{k=0}^{p} \nabla_{\theta_i} \log \Pr(u_k | s_k; \theta) \right) \left(\gamma^{t_k} r_k - b_i \right) \right\rangle \qquad (21)$$

We use the optimal baseline b_i shown in [26].

2) Sliding trajectory windows: As presented above, (21) learns from rewards received early in the trajectory but not later, since γ^t drives the value of later rewards to 0. Therefore we break a trajectory into sequences of $\langle s_\mu, a_\mu, r_\mu \rangle$, with one sequence starting at each timestep, and present those as separate trajectories to (21). Sequence length n is chosen such that $\gamma^n \geq 0.05 > \gamma^{n+1}$: the terms beyond this increase computational burden without significantly improving accuracy.

D. PGRL+μ: *Combining gradient and metapolicy updates*

Changes to the base policy's θ can come from the base PGRL estimator (§IV-D) after every epoch, or from the metapolicy (17) via u_μ after every trial. When the base PGRL estimator produces an estimate, we use it to adjust θ. But we can also pretend that it came from π_μ, and use it as u_μ for the computation of $\nabla_\theta \log \pi(s_\mu, u_\mu; \Theta)$. Thus both the PGRL and the π_μ updates and metarewards can be used to form the μ-trajectory for (21).[1] Although tuning can improve performance, for expository simplicity we set the magnitudes of all updates to be the same.

E. Autodetect metalearner: *Monitoring metapolicy quality*

Early in training, the metalearner can give poor advice, leading to high-cost policies. If ample non-mission training time is allocated, then such runs do not pose a problem. In our single-node example, roughly 50 runs of 100 trials each were required before the metapolicy reliably improved upon PGRL. Our hope is that any knowledge encoded by the metalearner can be transferred between tasks and that therefore metapolicy training time for any actual scenario may be low, but here we show how the metalearner can be trained, tuned, and tested without significantly interfering with the performance of a live network.

After each epoch, PGRL adjusts θ by some amount $\Delta_\nabla \theta$, and the following trial yields a μ-reward (20). If instead it had been the metalearner that had recommended the same update $\Delta_\mu \theta = \Delta_\nabla \theta$, the same reward (on average) would have been observed. Moreover, when the two recommend "opposite" changes to θ, it is more often the case than not that the reward earned by the metalearner would at least have had the opposite sign to that actually seen. Thus it is possible to make an educated guess as to whether the metapolicy's output would have led to an improvement in the base policy, without actually making the suggested change.

We define the *update disagreement* δ as the unsigned angle between $\Delta_\nabla \theta$ and $\Delta_\mu \theta$. We estimate metapolicy quality by tracking the average disagreement between base policy and metapolicy updates. For each policy update, we score the comparison between the metapolicy to PGRL as follows:

∇ good:	$\delta \leq \frac{\pi}{2}$	1
$r_\mu > \tau$	$\delta \geq \frac{3\pi}{2}$	-1
∇ bad:	$\delta \leq \frac{\pi}{2}$	-1
$r_\mu < -\tau$	$\delta \geq \frac{3\pi}{2}$	1

Our estimate of metapolicy quality is the mean of the trial-by-trial scores during a run, which is roughly equivalent to the negative of the slope of a linear regression of metareward vs. disagreement, but offers better numerical properties: the result is bounded, which eliminates the effect of outliers due to unusual exploration noise; it discards the ambiguous cases in which the PGRL and δ differ by around $\frac{\pi}{2}$; and it produces a valid result when the metapolicy and policy [dis]agree perfectly. τ was roughly hand-tuned to yield a positive metareward quality at, on average, the same time as the standard PGRL+μ metalearner started to outperform PGRL, yielding $\tau = 1000$. We perform exponential smoothing (base 0.9) across runs on the result in order to use more available information.

VI. Results

We generate random data-ferrying *problems* each of which consists of a random position and orientation for each sensor. At each *timestep* the aircraft flies some distance towards the next waypoint, measures the current SNRs via probe packets, and requests some data from a node at the power indicated by the power policy. A *trial* is a single complete flight over the radio field. An *epoch* is a small number of trials, after which we estimate the policy gradients and update the policies. A *run* is an attempt to optimise radio power and waypoint position policies for a given *problem*, and here consists of 100 trials. For each problem we generate a new random radio field and re-initialise the policies, but not the metapolicies. Although it is possible to learn as soon as we have enough trials to produce a gradient estimate, for simplicity we instead hold

[1] While the base gradient update provides a legitimate value for u_μ and hence $\nabla_\theta \log \pi(s_\mu, u_\mu; \Theta)$, it means that the forward roll-out is off-policy. This is theoretically interesting, but in practice, for our application, not problematic.

 ISBN: 978-1-61208-207-3

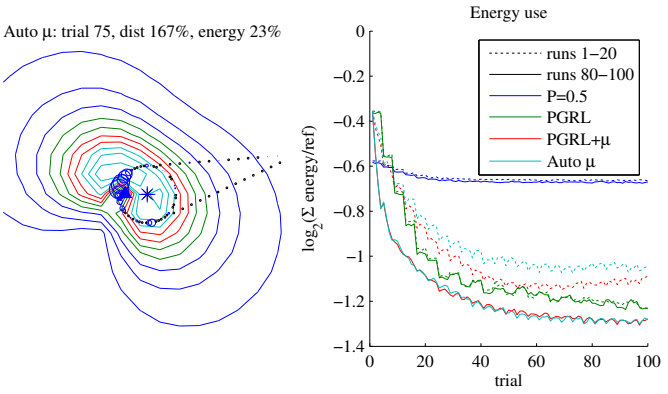

Fig. 2. Learning to minimise energy use for the single-node case. *Left:* sample trajectory plotted in space, superimposed over reference rate contours that show what the aircraft would see in flat level flight (not what it actually sees as it steers and banks). The aircraft starts at ⋆; the waypoint is at ✳. Circle size is proportional to data rate. *Right:* energy use for the three algorithms for the current experiment, averaged over the last few runs, which measures the performance gains possible from a well-trained metalearner.

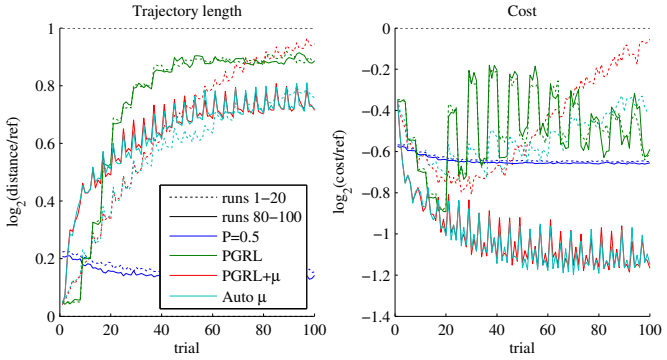

Fig. 3. Further learning details for the 1-node case. *Left:* trajectory length. *Right:* aggregate cost vs. trials relative to Reference (averaged over the last few runs of all experiments).

the metapolicy's parameters Θ constant during each run. An *experiment* is a set of 100 runs during which the metalearners have the opportunity to adapt. For each experiment we re-initialise the metapolicy parameters. To generate the graphs, we average over 50 experiments.

We will compare the metalearners to two non-learning approaches and to PGRL:

Reference is the non-learning autopilot defined in §III-A.

Half-power learns waypoint placement as described in §IV-C, but instead of learning the power policy, sets $P = \frac{P_{\max}}{2}$. This is guaranteed to increase trajectory length by a factor ≤ 2.

PGRL uses the Learning autopilot and the conventional PGRL approach described in §IV-D, without the metalearner.

Parameters: The aircraft flies at speed $v = 1$ at altitude $z = 3$. The maximum turning rate of $\omega = 20°/s$ yields a turning radius $r = \frac{v}{\omega} \approx 2.9$. Radios use $P_{\max} = 150$ and bandwidth $\beta = 1$, and the background noise $N = 2$. Each sensor's data requirement *req* = 20. These parameters do not qualitatively affect the results, and can be replaced by appropriate values for any given hardware. For waypoint placement the learning rate $\alpha = 0.5$ and the exploration parameter $\sigma = 1$. The power policy uses $\alpha = 0.3$, $\sigma = 0.2$, $\phi = 1, \varphi_j = 1 \forall j$. Policy gradient estimates for waypoint placement and energy are computed and applied every 4 trials (one epoch) for reasons described in [2]. Metapolicy gradient estimates are computed and applied after each run as described in §V-C. The metalearner's temporal reward discount $\gamma = 0.25$.

A. Learning from base gradient, Metalearning

Figure 2 shows the energy use of our learners on a single node over the course of 100 trials. The behaviour of the metalearners' early learning is illustrated by performance plots over runs 1–20. We contrast this with the performance of well-trained networks over runs 80–100. All performance graphs show $\log_2$ of the ratio of performance compared to Reference.

In our scenarios, **Half-power** (or "P=0.5") consistently reduces sensor energy requirements to about $2^{-0.65} \simeq 65\%$ of Reference in exchange for a 10% length increase, although this varies with data requirement—for high requirements (e.g., req > 100) Half-power reduces energy use by only 20% with a 60% increase in trajectory length. Despite the fixed power policy, energy use decreases slightly over time as waypoint positions are optimised.

Figure 3 shows how the learners' behaviours change during each run as the waypoint and energy policies are refined. After 20 or so trials, **PGRL** performs very well (Figure 3, Cost), but as it nears the optimal solution it falls into a cycle of discovering and rediscovering the cliff when random exploration steps take trajectory length over $d_{\max}$, resulting in frequent high-cost trajectories. It still outperforms Reference and untrained (runs 1–20) PGRL+μ, but Half-power is superior. In contrast, the trained (runs 80–100) metalearners outperform PGRL both early in each problem (the first few trials) during which the metapolicies rapidly push the policies towards lower energy use, and later, where they almost completely eliminate cliff-jumping.

The higher-level time-varying behaviour shown by the metalearners can be seen in the difference between runs 1–20 and 80–100, and more explicitly in Figure 4. As the metalearners observe the learners solving new problems, they refine their μ–policies, yielding performance that improves from run to run. Figures 2 and 3 show snapshots of per-trial performance averaged over runs 1–20 and 80–100, and the improvement of average per-trial performance over runs can be seen in Figure 4.

PGRL+μ usually outperforms the non-meta approaches by a significant margin after about 30 runs, allowing discovery of policies that use only 40% of the sensor energy of Reference and 65% that of Half-power, while seldom exceeding the aircraft's soft range limit. Perhaps surprisingly, even with such a simple representation we found that handcoding a metapolicy that outperforms the learned one was not easy. However, Figure 4 shows that early in metapolicy training, especially for the first 20 runs, PGRL+μ performs poorly, producing

 ISBN: 978-1-61208-207-3

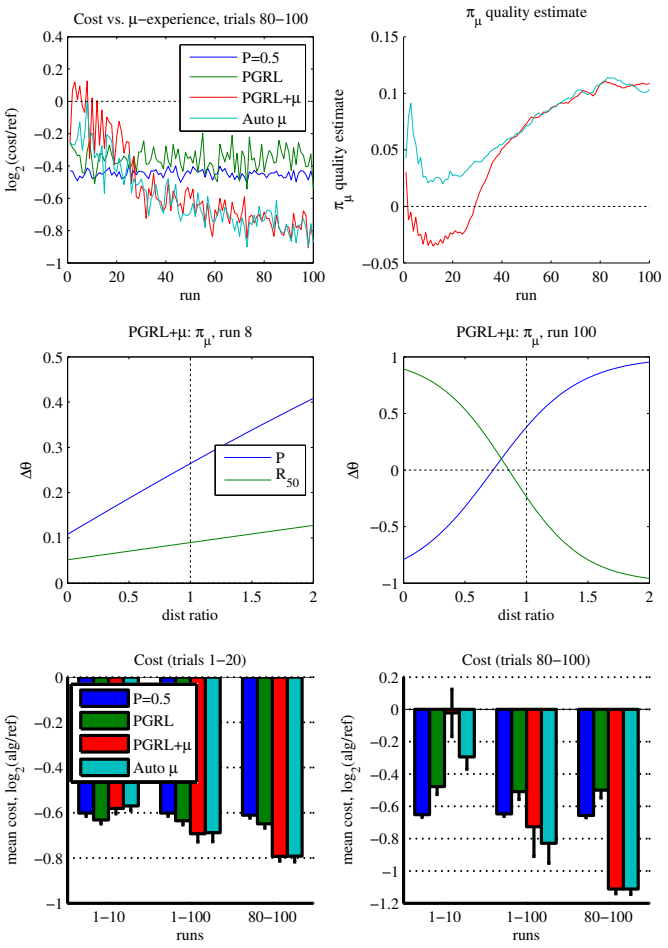

Fig. 4. Metalearning on 1 node. *Top left:* Trained (trials 80–100) performance of the algorithms over runs, during which the metalearners improve with experience. *Right:* The estimate of the quality of the metalearner's output through time, both for PGRL+μ, which does not use the information, and for Autodetect, which does. *Middle:* representations of the metapolicies' average actions $u_\mu = \Delta\theta$ vs. the observation of the previous trajectory, "dist ratio" $s_\mu = \frac{d}{d_{\max}}$, shown early in μ–training (run 8) and late (run 100). *Bottom:* Average performance of the algorithms relative to Reference at key points during learning and metatraining. Shown are the values measured across the runs indicated on the horizontal axis for *left:* just the first 20 trials and *right:* (same legend) the last 21 trials in each run. Error bars show problem-to-problem standard deviation 1σ of solution quality compared to Reference.

trajectories that are on average no better than Reference and are frequently far worse.

This problem is alleviated by **Autodetect**: in early runs, before its metapolicy has been well-trained, the quality measure often prevents the metapolicy from being used. As can be seen from the π_μ quality graph in Figure 4, the quality measure is on average above 0 (the threshold for use of the metapolicy's output), but in individual experiments it drops below 0 at the appropriate times and thus disables the use of the metapolicy. The Cost vs. μ–experience and bar graphs show the consequence: Autodetect's performance tracks that of PGRL early in training, and as the metalearner's experience with different problems in the domain grows, it surpasses PGRL and performs as well as PGRL+μ.

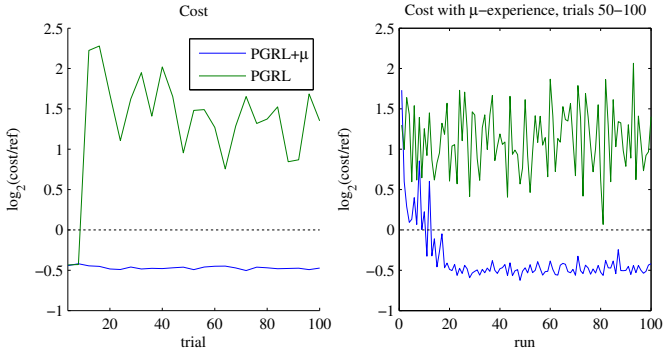

Fig. 5. Compensating for poorly chosen learning rates: PGRL vs. Autodetect with $\alpha_w = 0.97^{\text{trial}}$ and energy policy learning rate $\alpha_e = 1$ — a value only twice one that performs well. (10-experiment average)

The bar graphs in Figure 4 break down performance over the two learning timescales in order to show how quickly learning progresses without (runs 1–10) and with (runs 80–100) a well-trained metalearner. Performance over trials 1–20 shows initial learning speed, while performance over trials 80–100 shows what may be expected as the network matures. This further emphasises Autodetect's ability to nearly match the performance of the better of PGRL or PGRL+μ.

Another interesting feature of the Autodetect learner, visible in Figure 4's π_μ quality graph, is that the measured metapolicy quality progresses differently from that of PGRL+μ: the quality estimate stays higher initially, but begins to track that of PGRL+μ around run 40. The two metapolicies' parameters evolve differently due to the differences in training: while PGRL+μ always sees meta-actions from both the metapolicy and PGRL, Autodetect sees only the latter until it has proven itself, thus obtaining fewer training examples drawn from a different distribution. Raising Autodetect's threshold makes it less likely that the metapolicy gives bad advice to the learner, but reduces μ–learning speed. The effect that this has on our metapolicy quality estimate through time is intriguing, but we leave investigation as future work.

Much of the training time shown in Figure 4 may be required only once in a "lifetime" due to the transferability of the trained metapolicies. Once their metapolicies are trained, the metalearners facilitate the discovery of a good policy extremely rapidly—*after only a few trials*. They aggressively push policy changes that they have found in the past result in higher performance: quickly reducing energy use until nearing the UA's range limit and then backing off without requiring further exploration of the cliff's high-cost trajectories.

B. Sensitivity to learning rates

The base PGRL learner can achieve stable results if the learning step sizes such as α in (6) are chosen carefully. If learning rates are small, learning is slow, but a larger values exacerbate the learner's tendency to cliff-jump. Therefore it is typical in reinforcement learning to have α decrease over the course of a run—for example $\alpha = \alpha_0 x^e, x < 1$ for the update after epoch e. Ideally $\alpha \to 0$ as the policy nears the optimal.

 ISBN: 978-1-61208-207-3

For our task, this both provides a convergence guarantee for the hillclimber and reduces cliff-jumping, but it requires hand-tuning of x and α_0. Furthermore it eliminates the system's ability to adapt to a slowly changing environment; e.g., foliage growth or physically shifted sensors.

A poor choice of α_e interferes with the learners' ability to remain near the optimum. Here we have set $\alpha_e = 1$. We set the waypoint-placement learning rate $\alpha_w = 0.97^{\text{trial}}$ in order to allow both policies to converge more quickly, which is unnecessary in our other tests but helpful here. Figure 5 shows results: while $\alpha_e = 0.3$ (as shown in §VI-A) produces reasonable results, $\alpha_e = 1$ results in the exploration of many high-cost trajectories. PGRL performs extremely poorly, but PGRL+μ learns to anticipate the large jumps that result from the poor choice of α_e, resulting in costs worse than with a carefully chosen α_e but still significantly better than PGRL. The ability of the metalearner to stabilise learners with less hand-tuning may be a great asset in real-world systems.

C. Knowledge transfer with metapolicy initialisation

While the learned energy and waypoint policies are highly problem-specific, the metapolicy is more broadly applicable. This is shown in §VI-A by the gains on new problems after training on previous ones. But can a metapolicy trained for our single-node scenario be used to accelerate policy learning for problems drawn from a broader domain? The question of metapolicy transferability motivates us to modify our problem space as follows:

- For each problem, we place 5 sensor nodes randomly on a 40×40 field. This yields a great variety of radio field shapes since the nodes interfere with each other.
- For each problem, the data requirement for each node is drawn randomly from $[10 \ldots 30]$.
- We have corrupted the aircraft's information about the nodes' locations with Gaussian noise with $\sigma = 3$.

Figure 6 shows training results: for "Transfer" the metapolicy is set to the final Autodetect value learned during the generation of Figures 2–4:

$$\Theta = \begin{bmatrix} 1.70 & -1.16 \\ -1.61 & 1.34 \end{bmatrix} \tag{22}$$

Figure 6 shows the results in the extended domain: the earlier metapolicy achieves better than a 60% reduction in sensor radio transmission energy over Reference, and again offering speed and stability improvements over non-meta PGRL. The ability of single-node metapolicies to generalise to a broader problem configuration space is encouraging, but it is not perfect. For example, if we allow node data requirements to be drawn randomly in $[1 \ldots 10]$ the metapolicy is not cautious enough in its recommendations for policy updates due to the short contact times. Over multiple runs the metapolicy does continue to adapt based on policy learning from the new problems, but due to space constraints we do not study multi-node metapolicy learning here.

We show progress to 200 trials for this scenario. Figure 6 shows that for Transfer this is overkill, but PGRL has not yet converged. Performance gains in this scenario, even with a repurposed metapolicy, are slightly greater than in our first example, with Transfer achieving a 62% savings and even Half-power effecting a 45% savings. This difference is primarily due to the sensor position misinformation: Reference does not understand the error and so its performance degrades sharply as the information is compromised, whereas the learners—even Half-power—modify their trajectories to work around the misinformation.

VII. Conclusion and future work

We have demonstrated the feasibility of a reinforcement learning approach for autonomous discovery of energy-saving behaviours for sensor networks. With a soft trajectory length limit of twice a handcoded reference and a single sensor, the UA tended to fly 75% further than under Reference while sensors reduced communication energy by 60%. When the UA's knowledge of sensor positions is degraded by even a modest amount, the advantage of the learning approaches increases rapidly.

The UA does not model the environment, but learns directly from experience, saving the costs of locating the sensors and building system models, and eliminating the effects of modelling errors. We use a trajectory encoding that is well-suited to the task of interfacing between learning algorithms and proprietary autopilots. Additionally, we concurrently optimise both power and waypoint-placement policies.

Our reinforcement metalearner uses experience with the process of learning data-ferrying policies in order to accelerate and stabilise a conventional PGRL system, and transfers acquired knowledge about the policy-optimisation process to a range of unseen problems. This furnishes a new mechanism for approaching the global optimum of an unseen data-ferrying scenario extremely quickly while sampling few high-cost trajectories.

Much work remains to be done. For example, the metalearner reduces the necessity of hand-tuning the system by compensating for poor choice of parameters. A poorly chosen exploration rate frequently produces trajectories that greatly exceed the aircraft's range limit. Preliminary results show that as the metalearner gains experience with the base optimiser it learns to compensate, modulating the problematic policy updates and keeping trajectory costs lower, but a full investigation into the range and limits of this capability, both in the data-ferrying domain and for general model-free optimisation, remains to be done. Also, we have examined only learning in the vicinity of a few nodes. While preliminary results on the transfer of this metaknowledge to multi-node problems is promising, efficiently scaling the metalearning mechanism to sensor networks of many nodes is important future work. Finally, a more detailed characterisation of when and to what extent the metaknowledge is transferable will be key to understanding the broader applicability of the technique.

References

[1] A. Jenkins, D. Henkel, and T. Brown, "Sensor data collection through gateways in a highly mobile mesh network," in *Proc. IEEE Wireless*

 ISBN: 978-1-61208-207-3

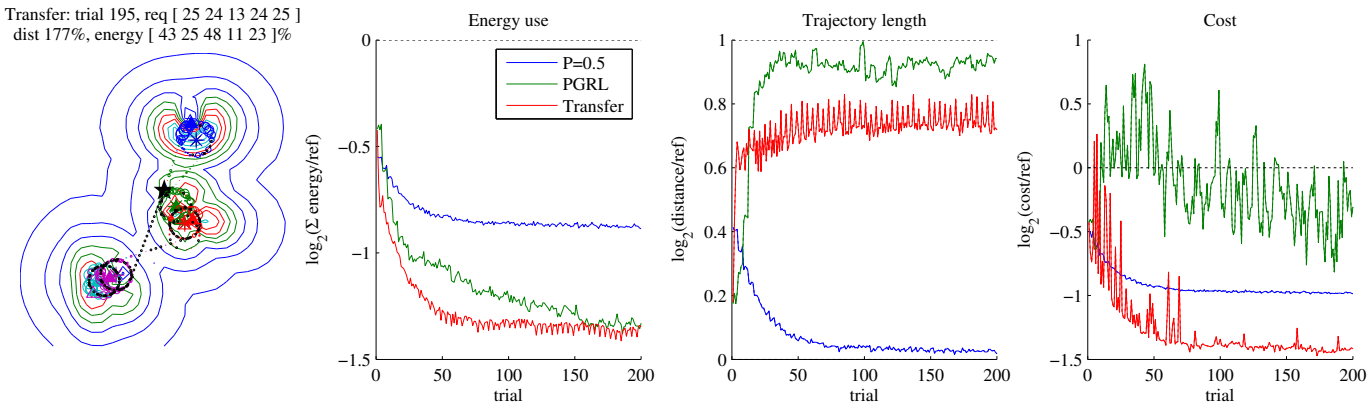

Fig. 6. Transferring the metapolicy to a larger field with varying data requirements and incorrect node position information. See Figures 2–4 for descriptions of the graphs. *Left:* the nodes are at ▲; their assumed positions, to which the waypoints were initialised, are at △; and "energy" is relative to Reference.

Communications and Networking Conference (WCNC). Hong Kong: IEEE, 2007, pp. 2784–2789.

[2] B. Pearre and T. X. Brown, "Fast, scalable, model-free trajectory optimization for wireless data ferries," in *IEEE International Conference on Computer Communications and Networks (ICCCN)*, 2011, pp. 370–377.

[3] H. Jun, W. Zhao, M. H. Ammar, E. W. Zegura, and C. Lee, "Trading latency for energy in densely deployed wireless ad hoc networks using message ferrying," *Ad Hoc Netw.*, vol. 5, pp. 444–461, May 2007.

[4] O. Tekdas, J. Lim, A. Terzis, and V. Isler, "Using mobile robots to harvest data from sensor fields," *IEEE Wireless Communications special Issue on Wireless Communications in Networked Robotics*, vol. 16, pp. 22–28, 2008.

[5] G. Anastasi, M. Conti, and M. Di Francesco, "Reliable and energy-efficient data collection in sparse sensor networks with mobile elements," *Perform. Eval.*, vol. 66, pp. 791–810, December 2009.

[6] 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*, vol. 18, pp. 1476–1488, 2007.

[7] R. Sugihara and R. K. Gupta, "Optimizing energy-latency trade-off in sensor networks with controlled mobility," in *IEEE INFOCOM Mini-conference*, 2009, pp. 2566–2570.

[8] D. Ciullo, G. Celik, and E. Modiano, "Minimizing transmission energy in sensor networks via trajectory control," in *IEEE Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2010, pp. 132–141.

[9] L. Bölöni and D. Turgut, "Should i send now or send later? a decision-theoretic approach to transmission scheduling in sensor networks with mobile sinks," *Wireless Communications and Mobile Computing*, vol. 8, no. 3, pp. 385–403, 2008.

[10] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: a survey," *Ad Hoc Netw.*, vol. 7, pp. 537–568, May 2009.

[11] Y. Gu, D. Bozdag, R. W. Brewer, and E. Ekici, "Data harvesting with mobile elements in wireless sensor networks," *Computer Networks 50*, vol. 17, pp. 3449–3465, 2006.

[12] A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava, "Mobile element scheduling with dynamic deadlines," *IEEE Transactions on Mobile Computing*, vol. 6, no. 4, pp. 395–410, 2007.

[13] D. Henkel and T. X. Brown, "Towards autonomous data ferry route design through reinforcement learning," in *Proceedings of the 2008 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WOWMOM)*. Washington, DC, USA: IEEE, 2008, pp. 1–6.

[14] T. He, K. won Lee, and A. Swami, "Flying in the dark: Controlling autonomous data ferries with partial observations," in *Proceedings of the eleventh ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*. New York, NY, USA: ACM, 2010, pp. 141–150.

[15] W. Zhao and M. H. Ammar, "Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks," in *Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems*, ser. FTDCS '03. Washington, DC, USA: IEEE, 2003, pp. 308–314.

[16] M. Dunbabin, P. Corke, I. Vasilescu, and D. Rus, "Data muling over underwater wireless sensor networks using an autonomous underwater vehicle," in *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, 2006, pp. 2091–2098.

[17] M. M. Bin Tariq, M. Ammar, and E. Zegura, "Message ferry route design for sparse ad hoc networks with mobile nodes," in *MobiHoc: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2006, pp. 37–48.

[18] R. Sugihara and R. K. Gupta, "Improving the data delivery latency in sensor networks with controlled mobility," in *Proc. 4th IEEE international conference on Distributed Computing in Sensor Systems*, ser. DCOSS. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 386–399.

[19] ——, "Path planning of data mules in sensor networks," in *ACM Trans. Sen. Netw.*, vol. 8, no. 1. New York, USA: ACM, Aug. 2011, pp. 1–27.

[20] D. Henkel and T. X. Brown, "On controlled node mobility in delay-tolerant networks of unmanned aerial vehicles," in *International Symposium on Advance Radio Technolgoies*, 2008, pp. 7–16.

[21] A. Carfang, E. W. Frew, and T. X. Brown, "Improved delay-tolerant communication by considering radio propagation in planning data ferry navigation," in *Proc. AIAA Guidance, Navigation, and Control*. Toronto, Canada: AIAA, August 2010, pp. 5322–5335.

[22] M. Stachura, A. Carfang, and E. W. Frew, "Active sensing by unmanned aircraft systems in realistic communication environments," in *IFAC Workshop on Networked Robotics*, 2009, pp. 62–67.

[23] F. Jiang and A. L. Swindlehurst, "Optimization of uav heading for the ground-to-air uplink," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 5, pp. 993–1005, 2012.

[24] C. E. Shannon, "Communication in the presence of noise," in *Proc. Institute of Radio Engineers*, vol. 37, no. 1, 1949, pp. 10–21.

[25] N. Wagle and E. W. Frew, "A particle filter approach to wifi target localization," in *AIAA Guidance, Navigation, and Control Conference*. Toronto, Canada: AIAA, August 2010, pp. 2287–2298.

[26] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, no. 4, pp. 682–697, 2008.

[27] P. Glynn, "Likelihood ratio gradient estimation: An overview," in *Proceedings of the 1987 Winter Simulation Conference*, 1987, pp. 366–375.

[28] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 1992.

[29] J. Z. Kolter, Z. Jackowski, and R. Tedrake, "Design, analysis and learning control of a fully actuated micro wind turbine," in *Proceedings of the 2012 American Control Conference (ACC)*, 2012.

[30] J. A. Hertz, A. S. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*. Perseus Books, 1991.

[31] J. Baxter and P. L. Bartlett, "Infinite-horizon policy-gradient estimation," *Journal of Artificial Intelligence Research*, vol. 15, pp. 319–350, 2001.

 ISBN: 978-1-61208-207-3