Spring 1-1-2015

Cascaded Optimization for a Persistent Data Ferrying Unmanned Aircraft

Anthony John Carfang
University of Colorado at Boulder, carfang@colorado.edu

Cascaded Optimization for a Persistent Data Ferrying Unmanned Aircraft

by

Anthony Carfang

B.S., Aerospace Engineering, Illinois Institute of Technology, 2008

B.S., Computer Science, Illinois Institute of Technology, 2008

M.Eng., Aerospace Engineering, University of Colorado, 2012

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Aerospace Engineering Sciences

2015

This thesis entitled:
Cascaded Optimization for a Persistent Data Ferrying Unmanned Aircraft
written by Anthony Carfang
has been approved for the Department of Aerospace Engineering Sciences

Prof. Eric W. Frew

Prof. Brian Argrow

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Carfang, Anthony (Ph.D., Aerospace Engineering)

Cascaded Optimization for a Persistent Data Ferrying Unmanned Aircraft

Thesis directed by Prof. Eric W. Frew

This dissertation develops and assesses a cascaded method for designing optimal periodic trajectories and link schedules for an unmanned aircraft to ferry data between stationary ground nodes. This results in a fast solution method without the need to artificially constrain system dynamics. Focusing on a fundamental ferrying problem that involves one source and one destination, but includes complex vehicle and Radio-Frequency (RF) dynamics, a cascaded structure to the system dynamics is uncovered. This structure is exploited by reformulating the nonlinear optimization problem into one that reduces the independent control to the vehicle's motion, while the link scheduling control is folded into the objective function and implemented as an optimal policy that depends on candidate motion control. This formulation is proven to maintain optimality while reducing computation time in comparison to traditional ferry optimization methods.

The discrete link scheduling problem takes the form of a combinatorial optimization problem that is known to be NP-Hard. A derived necessary condition for optimality guides the development of several heuristic algorithms, specifically the Most-Data-First Algorithm and the Knapsack Adaptation. These heuristics are extended to larger ferrying scenarios, and assessed analytically and through Monte Carlo simulation, showing better throughput performance in the same order of magnitude of computation time in comparison to other common link scheduling policies. The cascaded optimization method is implemented with a novel embedded software system on a small, unmanned aircraft to validate the simulation results with field experiments. To address the sensitivity of results on trajectory tracking performance, a system that combines motion and link control with waypoint-based navigation is developed and assessed through field experiments. The data ferrying algorithms are further extended by incorporating a Gaussian process to opportunistically learn the RF environment. By continuously improving RF models, the cascaded planner can continually improve the ferrying system's overall performance.

Acknowledgements

This dissertation has been made possible with the support, guidance and encouragement of many significant people others. The partnership under NSF IIP-1161029 with Brigham Young, Air Force Research Labs, United Technologies Research Corporation, Northrop Grumman, and others provided an incredible amount of intellectual, as well as financial support for this work. The aircraft, sensor and autopilot systems are the direct result of many years of development within the Research and Engineering Center for Unmanned Vehicles (RECUV) at the University of Colorado. The support and instruction from Jack Elston, Maciej Stachura, James Mack, Brian Argrow and Eric Frew have been paramount in this regard. Dr. Argrow and Dr. Frew have been inspirational in directing and motivating my research. This research has been incredibly rewarding, bridging several broad disciplines, including control systems, optimization, aircraft flight regulation, communication systems, software engineering, and discrete math. In enabling my combination of such a vast array of fields, I would like to thank my advisor, Dr. Frew, along with Drs. Timothy Brown and Derek Kingston, for the valuable discussions, guidance, and resources that they have provided to me. Along with Dr. Dale Lawrence and Dr. Argrow, my committee's critical feedback has been key to making this research as fruitful as possible. I am very appreciative of the numerous other students in RECUV for their support with flight experiments and technical discussions, in particular Jason Durrie, Neeti Wagle, and Kevin Rauhauser. Finally, I would like to thank my parents, Anthony and Carol; my sister, Laura; my brothers, Andrew and Robert; and my amazing wife Jennie, for their never-ending patience, moral support, love, and inspiration throughout my entire Ph.D. career.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

INTRODUCTION

This dissertation addresses the problem of optimizing a persistent data ferrying system that uses a small unmanned aircraft (UA) to transfer data from source nodes to a destination. By combining motion control with delay-tolerant networking, effective communication range is increased, and average throughput of the system can be improved. This optimal control problem involves both trajectory optimization and communication link scheduling, a complex problem challenged further by nonlinear vehicle kinematics of the UA, and even more with the complicated nonlinear radio frequency (RF) dynamics involved in wireless communications between nodes and aircraft.

This work shows that the system dynamics exhibit a unique cascaded structure between the vehicle and RF dynamics. The central thesis of this work is that this structure can be exploited to reformulate the nonlinear ferry optimization problem into one that reduces the independent variables to the vehicle's motion, with link scheduling determined by an optimal policy within the cost function of the trajectory optimization. Though most literature constraint system dynamics for tractability, the novel formulation in this thesis reduces the problem's computation time without restricting the system dynamics. This work further reduces computational burden through the development of fast, near-optimal link scheduling policies, and assesses the cascaded solution through simulation and field experiments. The development of this thesis results in a detailed solution methodology to quickly solve data ferry problems that account for complex system dynamics effectively. This fast, scalable and near-optimal solution methodology enables even more complex ferrying problems to achieve faster solutions with higher performance.

1.1 Motivation

A key aspect to any sensor network is the ability to communicate between nodes. However, wireless sensor networks must cope with uncertain RF environments, interference from uncooperative transmitters, sparse sensor distribution, and even intermittent connectivity. The mobility of an unmanned aircraft can be a great asset to sparse sensor networks where communication between sensors can be difficult. The aircraft can be controlled specifically to collect and deliver data from these sensors, acting as a data ferry [86]. Unmanned aircraft can provide long range communications [64], and their autonomous nature makes them an ideal choice for persistent sensing, which focuses on periodic trajectories of the ferries [28, 64]. This section presents prior and on-going work related to the data ferry problem, which incorporates delay-tolerant networking, mobility control, radio frequency (RF) environment modeling, and nonlinear and combinatorial optimization techniques.

1.1.1 Mobility in Wireless Sensor Networks

Including node mobility within wireless sensor networks has been shown to enhance communication performance of the network by taking advantage of spatial variation found in RF environments [33]. The ability for mobile relays to adapt to RF environments in order to improve network throughput was further demonstrated with autonomous underwater vehicles [24, 37] and unmanned aircraft [8]. Many systems have since incorporated networking equipment on unmanned aircraft to exploit this point [17, 54, 63, 81]. However, these systems require instantaneous end-to-end connectivity, and result in the UA remaining at a static location within the sensor network. For sparse sensor networks, instantaneous end-to-end connections may not be guaranteed. Instead, by allowing for delay in end-to-end connectivity, relay nodes can make use of store-and-forward paradigms to deliver data [85].

Routing within delay-tolerant networks (DTN) with mobile nodes has been extensively studied. Epidemic routing [74] is required when node movement may be random or unknown, where each node forwards packets to other nodes as they are encountered, hoping to increase the probability that the packet will eventually reach its destination. Instead of relying on sensor nodes to flood the network and relay data, it

is beneficial to introduce dedicated mobile relay nodes, termed Data Ferries [65]. Routing efficiency can be further improved with forwarding heuristics based on knowledge of the ferry routes [4, 60], such as vehicles on a highway [15], city buses [82] and commercial aircraft [48].

Data Ferrying can enable delayed communication over a large area with little infrastructure. This can be useful following natural disasters that have damaged network infrastructure, or enabling last-mile communication to connect rural areas to major cities [80]. Rather than communicate opportunistically, this work focuses on the optimal design and deployment of a dedicated data ferrying unmanned aircraft. Optimizing a data ferrying system requires controlling the ferry's motion in addition to determining a communication link schedule with each node. At the same time, this system must balance acceptable delay with the achievable gains in throughput, and in a way that fairly meets each node's data requirements.

In a delay-tolerant sensor network with known data needs, specific mobility and routing control can result in strong throughput within the DTN. By restricting ferry motion to straight lines, and assuming communication occurs at a fixed rate only when the ferry reaches a sensor, the ferry system is abstracted to a graph where the ferry traverses edges of a graph to "visit" nodes to communicate with them. This paradigm allows Traveling Salesman Problem (TSP) techniques to be used [44]. The edges of the graph can be weighted based on data needs of each node in addition to distances [14, 66], or based on fairness and data needs [80]. Relaxing the dynamics restrictions slightly, variations of the TSP have been explored to include Dubins-Vehicle dynamics [19, 55], or allowing the ferry to communicate at a fixed rate within a fixed radius of the nodes, using the TSP with Neighborhoods paradigm [6].

For persistent ferrying where the infinite horizon solution is composed of repeated ferrying cycles, graph theory is used to find Hamiltonian Cycle solutions, [66], potentially including strategic subtours (visiting nodes multiple times within a cycle) [49]. Frequencies of the subtours can be adjusted to better meet the data needs of the nodes as well [64]. Though strategic subtours gain up to 40% improvement over Hamiltonian Cycles, optimality still cannot be guaranteed. Further, The TSP is well known to be NP-Hard [70], for which optimal solutions cannot be efficiently found. Hence, TSP-based solutions focus on the performance of heuristics. This work derives methods to quickly determine optimal ferrying solutions.

1.1.2 Importance of Communication Models

Significantly, graph-theoretic techniques assume the ferry moves in straight lines, and that data transfer occurs at a fixed rate between ferry and node only when the two are within some proximity of each other. This disk model assumption is pervasive in mobile wireless network literature [6, 14, 36, 40, 48, 87], even in work that incorporates nonlinear trajectory optimization techniques for vehicle control [25, 42]. Communication rates, instead, vary with signal strength, and RF environments have been shown experimentally to be spatially and temporally much more complex than the disk model can represent [10, 17, 53, 51, 68, 76]. Graph-theoretic assumptions may be appropriate and give reasonable results when node spacing far exceeds the lengthscales of vehicle and communication dynamics. However, as the aircraft nears a node, or in networks where nodes are only somewhat sparsely spaced, these dynamics become a significant concern. Between unmanned aircraft and ground nodes, position changes on the order of 10 m have been shown to drastically change how an RF signal is received [16].

To handle the well-known challenge of such signal characteristics, the IEEE 802.11 wireless communication protocol [39] defines multiple modulation schemes that vary error-correcting techniques in order to increase robustness in areas of weak signal strength, and increase average throughput in areas of strong signal strength. With such complex environments, and protocols to accommodate them, the applicability of disk-model-based solutions is significantly weakened [9].

This dissertation relaxes the assumptions implicit in TSP solutions, and instead includes dynamics within the ferrying system. This, of course, requires knowledge of the RF environment. Much work has developed theoretical and statistical models of RF environments, including the effects of distance and antenna characteristics [61], as well as shadowing and fading [31, 51]. Methods have also been developed to adapt to interferers and dynamic environments [16], as well as continually learn and improve models on-line [11, 68, 76]. (The impact of incorporating such methods with data ferrying is investigated in Chapter 6.)

1.1.3 Novel Ferry Solutions

Some initial data ferrying work does include a realistic fading RF environment [27, 28]. A computationally intensive Mixed-Integer Linear Program (MILP) is formed to optimize bandwidth allocation along

the ferry's trajectory. However, trajectory dynamics are limited to straight lines between points, also forming Hamiltonian Cycles that have been shown to be suboptimal [49]. The goal of this dissertation is to solve the data ferrying problem while accounting for complex aircraft and communication dynamics, increasing the fidelity and applicability of the results. To address the tractability of the optimization problem, a cascaded optimization solution is derived, distinguishing motion control from communication control.

The communication control is a combinatorial optimization problem determining the ferry's communication link schedule. This link scheduling allocation is related to servicing machines in the Restless Bandit Problem [45]. Gittins [30] and Whittle indices [78] determine frequencies that each sensor node must be serviced by the ferry, and the ferry's motion is to move at these frequencies to *draining regions*, where the communication rate to each node is greater than the rate at which the sensors gather data. Though the existence of draining regions is necessary and sufficient to keep the sensors' buffers stable, the results say little toward optimizing the ferry's motion.

Rather than focusing on trajectories that just stabilize, this work characterizes optimal trajectories. Optimal trajectories that consider both vehicle and communication dynamics have been examined [42, 58], but such work is limited to the Data Gathering Problem (DGP), gathering a fixed amount of data rather than ferrying for persistent sensing. The DGP assumes unlimited buffer size on the aircraft, and lacks the switching dynamics of also delivering data involved in the ferrying problem. Additionally, link scheduling in these works is fixed with the ferry suboptimally communicating with each sensor equally at all time, rather than controlling the allocation variable. This dissertation seeks to optimize the persistent ferrying problem, where data delivery is essential, and it optimizes over both the vehicle's path and the allocation parameter to do so.

1.2 Detailed Problem and Solution

The problem considered in this dissertation is persistent data ferrying with an unmanned aircraft. Instead of solving complicated ferrying scenarios with significant computational resources and time, this work examines properties of a fundamental ferrying problem in order to develop smarter strategies for ferry optimization. The discussion is initially limited to a system of one UA ferrying data from a single

stationary source node to a single stationary destination, and the UA is tasked with flying a periodic time-constrained route with the goal of maximizing the network's Quality of Service (Qos) 1.1. This leads to a clear, rich understanding of the ferry problem, and strategies developed from this fundamental scenario are then extended to scenarios of multiple nodes and data flows.

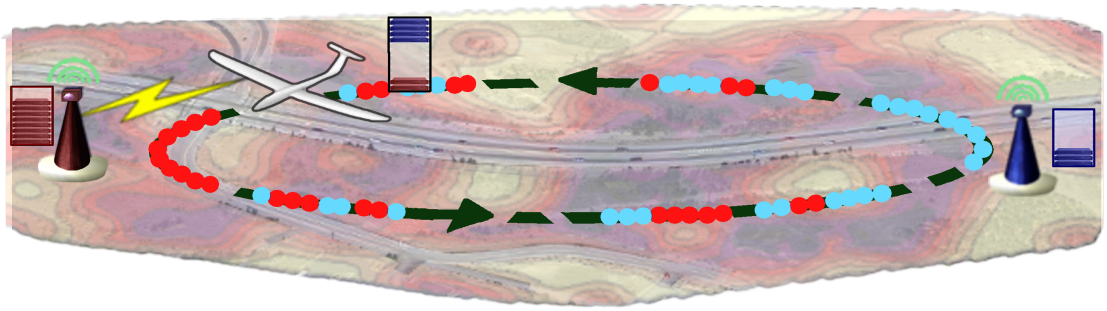


Figure 1.1: The fundamental ferrying problem, optimizing the unmanned aircraft's trajectory as well as communication link scheduling to transfer data between two communication-limited sensors.

One assumption made to solving the ferry problem is that the RF environments between all nodes and the UA are known. Surveys of radio frequency environments have shown them to be complicated and time-varying in nature ([16, 53, 76]). Thus there is a time-sensitive need to model the RF environment before the ferry system can be optimized. Techniques for learning and modeling RF environments are integrated with ferrying, demonstrating the ability for the ferry's plan to be updated as knowledge of the environment improves. The need to solve the ferrying problem in the same time scale of changing RF environments restricts computation time available for the ferry optimization; however, the discretized link-scheduling problem is a constrained combinatorial optimization problem of a form known to be NP-Hard in the resolution of the discretization, presenting further challenge and the need for fast, near-optimal solutions.

The goal of this dissertation is to develop a framework for optimizing data ferry systems that considers both the nonlinear vehicle kinematics of an unmanned aircraft, along with the complex dynamics exhibited by wireless communications. Given this goal, the framework and associated algorithms developed, implemented, and assessed in this dissertation have the following unique features:

- Handle complicated system and communication dynamics,
- Reduce computational effort over conventional approaches,

- Extend to more complex ferry systems.

Achieving these objectives is done by exploiting a specific structure of the ferry problem uncovered from an in-depth analysis of the fundamental problem scenario, distinguishing motion dynamics from communication dynamics. A cascaded optimization problem formulation is derived where motion control is optimized at the higher level, and cascading down to the communication link-scheduling optimization. Using a genetic algorithm to solve discretized versions of the original and the cascaded nonlinear optimization problems demonstrates the computational savings of the cascaded method. This optimization framework reduces the ferry problem dimension, and extends to complicated ferrying scenarios (e.g. more source nodes, multiple destination nodes, mobile networks, etc.).

In order to further reduce computation time, this work develops two heuristic policies to quickly implement the link-scheduling problem (LSP). A necessary condition of optimality is derived, and forms an equality constraint in the LSP. The Most-Data-First (MDF) policy is based on allocating discrete segments to one node in a greedy order, but alternates between nodes in order to minimize error in the equality constraint. An additional policy manipulates the equality constraint and relates the new form to the Knapsack Problem ([41]). Heuristics from the well-studied Knapsack Packing Problem are then applied to the ferry's link-scheduling problem. Through Monte Carlo Simulation, both the MDF and Knapsack heuristics are shown to be near-optimal and faster than other heuristics commonly used in ferrying literature. These trends are further validated through flight experiments as well.

1.3 Contributions

This dissertation makes three key contributions: (1) derivation and analysis of a novel cascaded solution method to the persistent ferry problem; (2) development and characterization of fast; near-optimal solutions to the link scheduling problem; and (3) implementation of a data ferrying unmanned aircraft system and its related experiments and assessments. Each of these central contributions contain multiple novel developments and sub-contributions, the details of which are as follows.

(1) **Derivation and analysis of a novel cascaded solution to the persistent ferry problem.** This work derives a novel cascaded solution method that separates vehicle motion control from communication link scheduling, and further develops a policy for link scheduling based on a given vehicle trajectory. Two significant sub-contributions result from analyzing this cascaded method:

   (a) the developed link scheduling policy is proved to retain optimality of the overall solution while reducing problem dimension.

   (b) Unlike other ferrying solution methods, this cascaded method eliminates the need to restrict system dynamics for tractability, and is further shown in simulation to improve computational costs over traditional nonlinear solvers.

(2) **Development of near-optimal solutions to the link scheduling problem.** The optimal link scheduling policy is an NP-Hard combinatorial optimization problem. This work develops several fast, near-optimal solutions to this combinatorial optimization problem, namely an adaptation of the Greedy heuristic to the Knapsack problem, and the Most-Data-First algorithm. There are three facets to characterizing these algorithms:

   (a) The algorithms are assessed both theoretically and in simulation in terms of run time, ferry performance, and robustness to trajectory discretization factors, resulting in obtaining near-optimal solutions orders of magnitude faster than binary integer programming methods.

   (b) These algorithms are extended to the case of multiple source nodes, with theoretical and simulation analysis also showing fast, near-optimal performance. The Most-Data-First algorithm has the added benefit of an inherent fairness with respect to number of source nodes.

   (c) These algorithms are compared in simulation to several common link scheduling heuristics, demonstrating similar computation times, but superior ferrying performance.

(3) **Implementation of a data ferrying unmanned aircraft system.** There are two key aspects required for optimal performance of any ferrying solution that present significant challenges in fielding

a system: trajectory tracking, and knowledge of the communication environment. This work address each of these significant challenges in the following ways:

(a) Trajectory tracking. This work implements a method combining waypoint following and coordinated link switching for ferry guidance and link control. The contribution of flight tests is three-fold: (i) the merits and challenges of this method is evaluated; (ii) the simulation results of the cascaded ferry solution will be validated with true flight data; and (iii) the robustness of each link scheduling policy to errors in trajectory tracking are experimentally assessed.

(b) Knowledge of the communication environment. A system is developed for measuring the throughput between an unmanned aircraft and ground node, showing the complex spatial and temporal characteristics of surveyed throughput fields. To ferry effectively through such environments, a method for integrating opportunistic learning with data ferrying is developed, enabling the ferry path to replan and improve its performance as RF knowledge improves. This system is assessed in simulation, showing two significant results: (i) opportunistic learning is effective and enables rapid improvements to ferry solutions with better throughput performance; and (ii) that non-parametric modeling with a Gaussian Process performs better with more robustness than parameter-based methods.

Through the development and characterization of a fast, scalable, near-optimal data ferrying system, along with the analysis of an implemented and fielded ferrying unmanned aircraft, wireless sensor networks and delay-tolerant networks can benefit greatly from the reliable, quantifiable performance of a data ferrying unmanned aircraft.

1.4 Dissertation Outline

The next chapter formally defines the persistent ferrying problem addressed in this dissertation, a challenging nonlinear optimal control problem. Though the results of this work apply to systems of many nodes, much of the discussion focuses on a fundamental system of one source node and one destination node. This enables a rich and clear understanding of the ferrying problem. Vehicle and communication dynamics

are detailed, with the key point of this work being that the dynamics will not need to be simplified to make the problem tractable. The objective of the persistent ferrying problem is formulated in terms of finding a particular point on a Pareto-optimal front that maximizes average throughput over a given period of time.

At the core of this thesis is the cascaded approach to optimize persistent ferry trajectories, presented in Chapter 3. The system dynamics in Chapter 2 reveal a cascaded structure that allows the problem to be reformulated, where the independent control is reduced to vehicle motion, and link scheduling is folded into the objective function implemented as an optimal policy based on a candidate motion control. This significantly reduces problem dimension, reducing computation time, as demonstrated in simulation with a genetic algorithm used for the nonlinear solver.

The link-scheduling policy is stated as: given a communication environment between source nodes and a destination node, and a discretized periodic trajectory for the UA to ferry data through this environment, determine the optimal allocation of communication links between the source and destination nodes that maximizes the amount of data delivered. Chapter 4 develops two fast, near-optimal policies to solve this NP-Hard combinatorial optimization problem. The Most-Data-First policy is developed from a heuristic based on best satisfying the equality constraint of the link scheduling problem; the greedy knapsack policy adapts heuristics from the knapsack packing problem from discrete mathematics. These policies are compared to several approaches commonly used in ferrying literature, and assessed in simulation in terms of relative throughput and computation time; the benefits are then analyzed with their combined effect on the cascaded ferry solution.

Chapter 5 analyzes the cascaded solution through field experiments. In addition to validating the previous simulation results, field implementation and experiments present challenges to the implicit assumptions present in a data ferrying optimization (including this work's cascaded solution). This chapter develops and assesses an approach to address the issue of perfect trajectory tracking: a waypoint-based controller that is subject to the black-box dynamics of an autopilot. The performance of the cascaded approach and link-scheduling policies are analyzed in terms of robustness to tracking errors.

An additional implicit assumption for data ferry optimization is that the communication environment is constant and known in advance. Much prior work attempts to model RF environments, requiring significant

exploration effort. Chapter 6 presents an opportunistic environmental learning approach to modeling the spatial variation of the RF environments with the use of a Gaussian process (GP). Instead of balancing an explore-versus-exploit issue, here prior models instantiate a data ferrying system, and measurements taken while ferrying are fed to a learner to improve the GP models of the environment. The new models then result in improved data ferry plans, quickly approaching solutions achieved with full knowledge. Such a system enables good data ferrying solutions in complex, potentially unknown, and potentially varying environments.

The contributions of this work are summarized in Chapter 7 , along with the challenges overcome and results achieved. It concludes with a discussion of open questions and directions for future work.

Chapter 2

THE DATA FERRYING PROBLEM

Data ferrying refers to an unmanned aircraft transferring data between stationary ground nodes by augmenting wireless communication with physically moving with the data for some portion of its delivery. This chapter describes the fundamental ferrying problem that is the focus of this work. The vehicle and communication dynamics of the system are first detailed, and the nonlinear modal optimization problem is then formally defined, along with a discussion of assumptions and problem scope.

2.1 Oveview

The general ferrying problem involves stationary source and destination nodes in a communication-challenged environment, where communication is facilitated by relaying data through a mobile ferry. The data flow i describes source node A_i generates data at rates of λ_i to be delivered to destination node B_i, and the priority γ_i places a relative value on the delivered data of flow i. The objective is to control the ferry's motion and communication through the environment to optimize the quality of service among the nodes.

This work primarily focuses on the fundamental case involving a single source A and single destination B, using a fixed-wing unmanned aircraft to ferry the data. (Source A generates data with priority 1 at rate λ.) Examining this specific scenario simplifies the discussion of the ferrying process, while also enabling a richness of understanding. Insights and results from examining this scenario can be extended to more general cases, and considerations and methods to do so are discussed.

2.2 Ferrying Unmanned Aircraft

2.2.1 Vehicle Dynamics

For persistent data ferrying, the ferry's motion is represented by a discretized sequence of time-based poses, p_k, t_k, for $i = 1, ..., n$, where its pose p_k at time t_k consists of the ferry's position and orientation in the environment. The vehicle dynamics can be described as

$$p_k = f_m\left(p_{k-1}, u_{m,k}\right) \tag{2.1}$$

$$t_k = f_t\left(t_{k-1}, u_{m,k}\right) \tag{2.2}$$

The theoretical contributions of this work apply to this general set of vehicle dynamics.

For simulation and experimental setup, an unmanned aircraft is assumed to have an on-board autopilot that provides low-level flight controls of roll, pitch, and yaw for stable flight and trajectory following. With the autopilot managing the low-level dynamic control, this work treats the aircraft and autopilot as a combined system that can be modeled kinematically. Assuming a required fixed altitude and fixed speed, a turn rate controller is considered to manage following non-holonomic aircraft kinematics. A first-order Euler forward difference method describes the kinematics of the ferry's pose of position and course angle, $p_k = [x_k, y_k, \psi_k]$:

$$\Delta p_k = \begin{bmatrix} \Delta x_k \\ \Delta y_k \\ \Delta \psi_k \end{bmatrix} = \begin{bmatrix} v\tau_k \cos(\psi_k) \\ v\tau_k \sin(\psi_k) \\ \omega_k \tau_k \end{bmatrix} \quad \begin{matrix} 0 < \tau_k \leq T_{max} \\ \\ |\omega_k| \leq \omega_{max} \end{matrix} \tag{2.3}$$

$$\Delta t_k = \tau_k = t_k - t_{k-1} \tag{2.4}$$

where ω_k is a turn rate command bounded by aircraft platform parameters, and τ_k is a control input to determine the discretized time dynamics.

2.2.2 Target Platform

The Skywalker X8 Unmanned Aircraft System, a small, rapidly deployable and durable system, is used as the dedicated to a communication relay. The 2.1m delta-wing planform supports a gross weight of

3.5 kg, flies between 12 and 25 m/s, and is software-limited by the autopilot to a 30-degree bank angle. In a steady-state coordinated turn, the turning radius at velocity v and bank angle ϕ is defined as

$$r = \frac{v^2}{g \tan \phi} \tag{2.5}$$

with turn rate determined as

$$\omega = \frac{v}{r_t} \tag{2.6}$$

At a flight speed of 16 m/s, a maximum 30-degree bank angle limits r_{min} to 45 m and ω_{max} to 20 deg/s.

2.3 Wireless Channel Dynamics

2.3.1 Channel Capacity

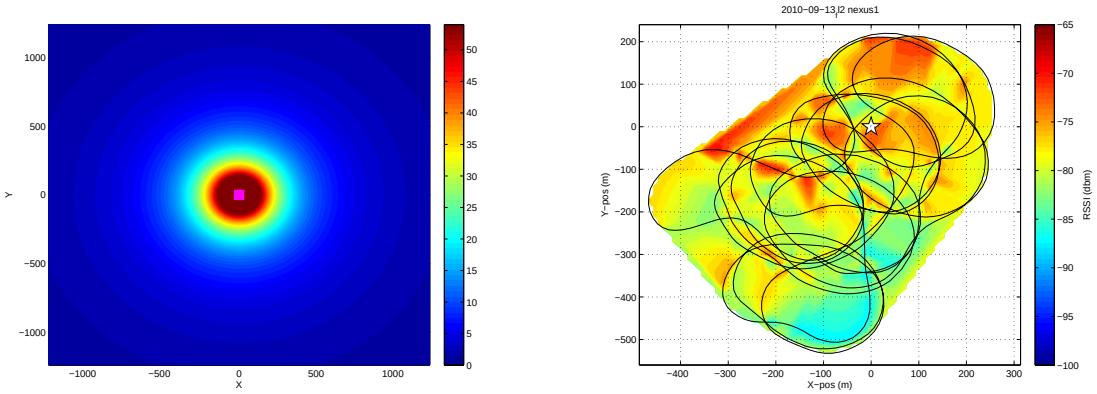
For Radio-frequency (RF) communications The signal power that one node at pose p_k receives from another transmitting node at pose p_j depends on their transmission power P, antenna gains G, and environmental factors such as fading characteristics F and propagation decay rate α [61]:

$$P_{ij} = \frac{P_i G(p_i) P_j G(p_j) F}{||p_i - p_j||^{\alpha}} \tag{2.7}$$

The Signal-to-Noise-plus Interference Ratio (SNIR) incorporates this received power with thermal noise as well as interference from other nodes or other atmospheric phenomena: $S_{ij} = P_{ij}/\eta_j$, where η_j includes noise at the receiver. Given a continuous-time channel with limited bandwidth β and Gaussian noise, the Shannon-Hartley theorem defines the channel's capacity:

$$R_{ij}^{SH} = \beta \log_2 (1 + S_{ij}) \tag{2.8}$$

On a larger scale, wireless communication follows equations (2.7) and (2.8), with a monotonically decreasing communication rate with distance (Fig. 2.1a). Small-scale fading, shadowing and additional environmental effects will lead to more complicated patterns of channel capacity that are both spatially and temporally dependent (Fig. 2.1b).



(a) Ideal communication rates, decreasing cleanly with distance

(b) Actual RF environment between UAV and Nexus Smartphone at Table Mountain, Boulder, Colorado. From Stachura et al.[68]

Figure 2.1: Shadowing, multipath, Gaussian noise and other discrepancies result in drastic differences between ideal and actual RF environments.

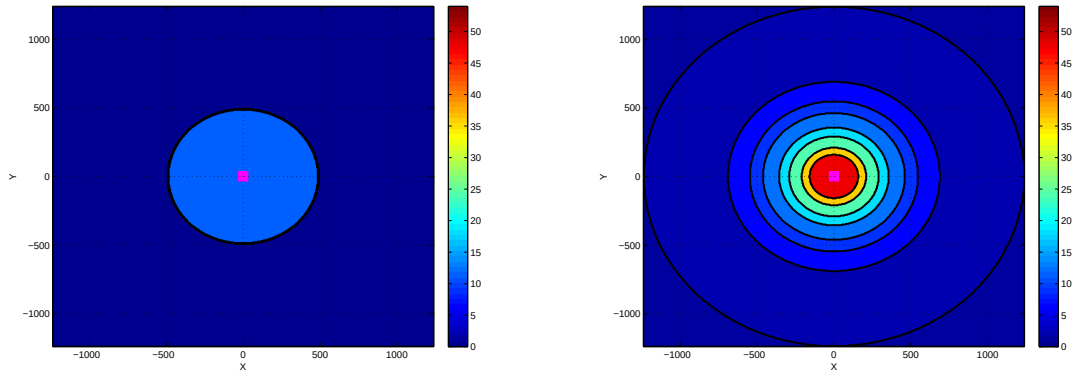
2.3.2 Communication Protocols

In implementation, protocol restrictions can lead to a different actual communication rate R_{ij} between nodes i and j. Rates may be specifically capped with some maximum throughput and some minimum to even establish communication, but with a continuum of rates in between (Fig. 2.1a). Alternatively, and commonly used for its simplicity, is the **disk model**, where nodes have a fixed communication rate, given that some minimum signal strength is received (Fig. 2.2a).

The IEEE 802.11 protocol, however, further defines various modulation schemes to improve the quality of service of the link. At low SNIR, more overhead and larger error-correcting codes increase data reliability, at the sacrifice of *goodput* – the actual useful data being sent over the link. At high SNIR, the data is more likely to transfer error-free, and so modulation schemes reduce overhead and error-correcting codes, improving goodput. Nodes negotiate a **throttled** rate, taking the best feasible option from a fixed set of communication rates $\boldsymbol{R_{thr}}$ [39]. Combined with Eqs. (2.7) and (2.8), rates vary as a step function in proportion to the channel's capacity (shown in Fig. 2.2b):

$$R_{ij} = \begin{cases} \max\left\{r \in \boldsymbol{R_{thr}} \mid r \leq R_{ij}^{SH}\right\}, & R_{ij}^{SH} \geq R_{min}, \\ 0, & \text{otherwise} \end{cases} \tag{2.9}$$

This is analogous to a multi-disk model – while the original disk model is limited to the achievable rate at the minimum acceptable signal strength, the throttled model takes advantage of even higher signal strengths by defining additional faster rates for those higher strengths.



(a) Disk model, with 1 fixed rate within a large enough signal strength

(b) Throttled rates, discrete and decreasing with distance

Figure 2.2: Common communication models based on a signal's log-distance decay

2.3.3 Ferry-Based Communication Models

The nonlinear aspects of a radio environment as well as aircraft dynamics make planning elaborate communications-based missions computationally challenging, particularly when fading and multipath effects occur on a spatial scale significantly shorter than the aircraft's turning radius. For this reason, much work assumes a disk model of communication, and holonomic vehicle dynamics. These assumptions enable graph theory to apply to the ferry problem, where the question of optimal ferrying reduces to solving for the frequency and order of which nodes to visit[66, 36]. In extremely sparse environments where nodes are very far apart, the nonlinear aspects of a node's communication are locally contained, and the disk model seems reasonable. However, on a smaller scale, this has been shown to not only be inaccurate [53], but also significantly reduce the accuracy and performance of the ferrying system [9]. For this reason, ferry planning here includes the full complexity of communication environments.

Much work has gone into complex modeling [31], estimating [68, 11] and predicting [76] the characteristics of an RF environment. Assuming a known RF environment, and with stationary sensor nodes, this

work takes SNIR as a known function of ferry position within the environment:

$$S_a(t_k) = f_s\left(p_k, \zeta_a\right) \tag{2.10a}$$

$$S_b(t_k) = f_s\left(p_k, \zeta_b\right) \tag{2.10b}$$

where p_k is a function of time, and ζ_a, ζ_b are functions of position and time to define the spatio-temporal characteristics of each node's communication channel throughout the environment. Combining this with Eq. (2.8) and the 802.11g mappings of Eq. (2.9) convert this SNIR to communication rates R_{ij} as functions of time. By not overly simplifying the communication environment, the ferry solutions determined in this work maintain high fidelity.

2.4 Buffer Dynamics

Section 2.3.3 describes the rate wireless communication is able to achieve. However, additional system constraints limit communication further.

2.4.1 Link Schedule Allocation Variables

The ferry has a single half-duplex communication channel, and can only send/receive data to/from one node at a time. Its routing status is represented by the communication link allocation vector $\boldsymbol{c}$ of 2 binary variables: $c_a, c_b \in \{0, 1\}$. The first represents the ferry listening to source node A, and the last represents transmitting to destination node B. $\boldsymbol{c}(t)$ defines the link schedule between ferry and nodes. The single channel on the ferry constrains the link schedule variables to

$$c_a + c_b \leq 1 \tag{2.11}$$

(In scenarios with more nodes and data flows, the link allocation vector would contain c_{aj} for each source node, and c_{bj} for each destination, with the sum of all at any given time less than or equal to 1.)

2.4.2 Buffer Definitions

The source node has a limited capacity buffer, or queues, associated with its data transfer needs.

$$0 \leq b_a(t) \leq b_a^{max} \tag{2.12}$$

The destination node is assumed to process incoming data as needed, or to have sufficiently large buffer so that its storage limitations are not a factor. Data successfully delivered is denoted by $b_b(t)$, which is non-decreasing with time. The ferry also has a limited buffer b_f for carrying data from source to destination. With more data flows in a larger ferrying system, the ferry's buffer would be split between each flow, with the value $b_{fj}(t)$ representing the amount of data from source node A_j destined for B_j is on the ferry at time t. The total amount of data on the ferry is the sum of all these buffers, which is limited by b_f^{max}.

$$0 \leq \sum b_{fj}(t) \leq b_f^{max} \tag{2.13}$$

The virtual buffer $b_{\overline{a}} \geq 0$ represents the amount of data that has overflowed the source node's buffer while waiting for the ferry. Denote the full set of buffers as

$$\boldsymbol{b}(t) = [b_a(t), b_b, b_f(t), b_{\overline{a}}(t)] \tag{2.14}$$

Minimizing latency is often the major concern in Delay-Tolerant Networks. Let l_b define the latency metric for data delivered to node B. Ferry performance is typically concerned with average or maximum delivery delay, and latency dynamics for a periodic ferrying system result in these metrics being a function of the ferry's period [35].

2.4.3 Modal Buffer Dynamics

The actual throughput achieved by the nodes additionally depends on the states of their buffers. For example, when the ferry's buffer is full, it cannot collect any data regardless of the channel capacity. Such constraints lead to 8 distinct modes that determine the buffer dynamics purely for A getting data to and from the ferrying aircraft. Denote this system of 8 modes as M_a. These modes are dependent on the ferry buffer, the node's buffer, and the channel's capacity relative to the rate at which the node generates data.

The specific conditions determining each mode are detailed in Table 2.1, where the *dash* represents any state, and *between* represents $(0 < b_f(t) \leq b_f^{max})$ for the ferry's buffer or $0 < b_a(t) < b_a^{max}$ for the buffer on node A.

Table 2.1: Modes of Buffer Dynamics for moving data from A to B

Mode	State Conditions		
	b_f	b_a	channel capacity
I	0	between	— —
	0	0	$c_a R_a < \lambda_a$
II	0	0	$c_a R_a \geq \lambda_a$
III	0	max	$c_a R_a < \lambda_a$
IV	between	between	— —
	between	0	$c_a R_a < \lambda_a$
	between	max	$c_a R_a \geq \lambda_a$
V	between	0	$c_a R_a \geq \lambda_a$
VI	between	max	$c_a R_a < \lambda_a$
VII	max	$b_a < b_a^{max}$	— —
VIII	max	max	— —

Data delivered successfully is limited by the communication rate between nodes, as well as the amount of data the ferry has in its buffer:

$$\Delta b_b(t_k) = \begin{cases} 0, & \text{Modes I, II, III} \\ \int_{t_{i-1}}^{t_k} c_b R_b dt, & \text{otherwise.} \end{cases} \tag{2.15}$$

Note that this buffer is non-decreasing. The rate at which the ferry's buffers change depends on the amount of data the source node has to transmit to the ferry, the amount of data the ferry has to transmit to the destination node, and their respective communication rates:

$$\Delta b_f(t_k) = \begin{cases} \lambda_a \Delta t_k, & \text{Modes II, V} \\ \int_{t_{k-1}}^{t_k} c_a R_a dt, & \text{Modes I, III} \\ \int_{t_{k-1}}^{t_k} c_a R_a - c_b R_b dt, & \text{Modes IV, VI} \\ -\int_{t_{k-1}}^{t_k} c_b R_b dt & \text{Modes VII, VIII.} \end{cases} \tag{2.16}$$

The rate at which the source node's buffer fluctuates depends on its data generation rate, it's buffer status, and the amount of room the ferry's buffer has for its data:

$$\Delta b_a(t_k) = \begin{cases} \int_{t_{k-1}}^{t_i} \lambda_a - c_a R_a dt, & \text{Modes I, IV} \\ \lambda_a \Delta t_i, & \text{Mode VII} \\ 0, & \text{otherwise.} \end{cases} \tag{2.17}$$

Overflow data is dependent on the state of the node's buffer, as well as the rate at which the ferry can get data from that node:

$$\Delta b_{\overline{a}}(t_k) = \Delta t_i \begin{cases} \int_{t_{k-1}}^{t_i} \lambda_a - c_a R_a dt, & \text{Modes III, VI} \\ \lambda_a \Delta t_i, & \text{Mode VIII} \\ 0, & \text{otherwise.} \end{cases} \tag{2.18}$$

Note that like the buffers for delivered data, the overflow buffers are also non-decreasing.

Combining Equations (2.15) - (2.18) results in the following Conservation of Data equation (true for all modes):

$$\lambda_a \Delta t_k = \Delta b_a(t_k) + \Delta b_{\overline{a}}(t_k) + \Delta b_f(t_k) + \Delta b_b(t_k) \tag{2.19}$$

The meaning of this equation is clear by integrating them over a single full period of the ferrying aircraft's trajectory: data generated by the source node in that period must be waiting in the sensor's buffer, overflowed the sensor's buffer, picked up by the aircraft, or delivered to the destination node. These equations will exhibit even more significance in Chapter 3.

2.4.4 Notes on Buffer Dynamics

Buffer dynamics are piecewise continuous - first derivatives are not continuous. The derivatives are discontinuous at boundaries of buffers, going from rates of communication $c_k R_i(t_k)$ to 0 when the buffer is empty, and data generation λ_i to 0 when the node's buffer is full.

Eight modes $M_a(t_k)$ for each data flow in larger systems can similarly be defined. However, the various 8-mode systems are not independent since there is only one ferry buffer. This limits the possible couplings of modes between the modes. For example, if $M_a(t_k)$ is mode II, then $b_a(t_k) = 0$. This implies $M_b(t_k)$ is either modes I, II, or III. Note also that since the link scheduling variables are also binary $c_i(t_k) \in \{0, 1\}$ (since the ferry has only one half-duplex radio), then $M_a(t_k)$ being mode II implies $c_a(t_k) = 1$, and therefore $c_b(t_k) = 0$. So $M_b(t_k)$ can only be modes I or III. For the case of two flows, this yields a total of 20 modes.

2.5 Optimal Control Problem

2.5.1 Ferry Objectives

For relaying data in a Delay-Tolerant Network, there are a variety of relevant objective functions, such as minimizing the number of packets lost to buffer overflows, or minimizing the average latency or delay [66, 80]. Most can be reduced to two Quality of Service objectives: maximize data transfered while minimizing delay. For example, the conservation of data equations (Eq. (2.19)) show that minimizing the rate of data overflowing A's buffer $\dot{b}_{\overline{a}}$ is related to maximizing the rate of data delivered from A to B, $\dot{b}_{\underline{b}}$. The persistent ferry-control problem then aims to optimize some quality of service objective J_{QOS} that is some composite of the two following functions:

$$J_D = \sum_{b_i} \gamma_i \Big(b_{b_i}(t_n) - b_{b_i}(t_0) \Big) \tag{2.20a}$$

$$J_L = \sum_{b_i} \gamma_i l_{b_i}(t_n) \tag{2.20b}$$

where the periodic trajectory's time t_n is a function of the control, the summations are over all destination nodes b_i, and data is weighted by the priorities γ_i. J_D represents the total data delivered at the end of the period – the data metric, and J_L is the maximum delay seen by any packet over the period – the latency metric.

In general there is a direct trade-off between achievable data throughput and the delivery delay associated with it. This trade-off is highlighted by considering the two paradigms for an aircraft to transfer data between two nodes with a smooth continuous communication environment that decreases monotonically with distance as in Fig 2.3, [18, 35]. The aircraft could act as a stationary relay, positioning itself at a point that maximizes the minimum communication rate to each node, indicated by the green dot. Alternatively, the aircraft can fly to A to gather data at a faster communication rate than the stationary relay, take the time to fly to B, and deliver it at a faster rate. Two key points about this trade-off come from analytical derivations [18, 35]:

(1) the mobile ferry always achieves a throughput at least as good as the stationary relay, at the cost of higher delay; and

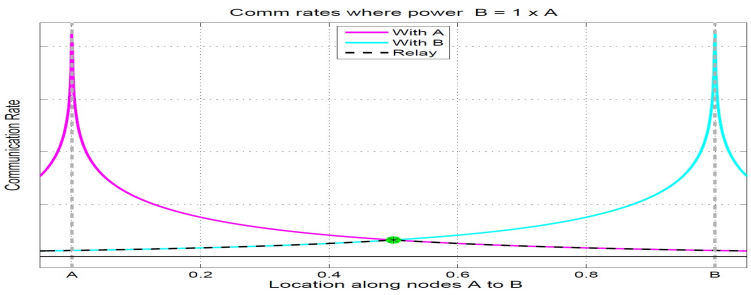


Figure 2.3: Example communication rates between ferry and nodes A and B, where rates decay monotonically and smoothly with distance from each node. A stationary relay with equal time allocation of links to A and B can only achieve a throughput limited to the dotted black line.

(2) the metric for worst-case latency can be parameterized by the duration of the aircraft trajectory's period.

This second point relieves the need to track the dynamics of each individual packet within the system. Instead, minimizing the ferry's period will minimize the worst-case delivery delay.

2.5.2 Problem Statement and Scope

The system states include the sequence of ferry poses, along with the status of the buffers. $\boldsymbol{X} = [\boldsymbol{p}, \boldsymbol{t}, \boldsymbol{b}]$ The input consists of control variables for ferry mobility (including the ferry's initial pose p_0), and the link schedule variables defining to which node the ferry is communicating:

$$\boldsymbol{u}(t) = [\boldsymbol{u_m}(t), \boldsymbol{c}(t)] \tag{2.21}$$

The infinite horizon path for a persistent data ferry is taken as a concatenation of periodic trajectories, as is common throughout persistent ferrying literature [6, 28, 49, 64, 66, 70]. Define $\boldsymbol{t} = \{t_k\}|_0^n$ as the sequence of discrete times over the period, corresponding to the ferry's sequence of poses $\Gamma = \{p_k\}|_0^n$, and $\beta = \{\boldsymbol{b}\}|_0^n$ as the sequence of buffer state values. Since latency is paramaterized by a function of the ferry's motion, the objectives of Eq. (2.20) can be combined to a single function of these two sequences and the corresponding time vector: $J_{QOS}(\Gamma, \beta, \boldsymbol{t})$. (Specific functions for J_{QOS} are presented in Section 3.2.1.) The optimal ferrying problem aims to find the set of controls over some periodic horizon, to maximize the quality of service between source and destination nodes:

Problem 1.

$$[\boldsymbol{u_m}, \boldsymbol{c}]^* = \underset{\boldsymbol{u_m}, \boldsymbol{c}}{\arg\max}\ J_{QOS}\left(\Gamma, \beta, \boldsymbol{t}\right) \tag{2.22a}$$

$$\text{subject to} \quad \text{Eqs. (2.3), (2.10)-(2.18)} \tag{2.22b}$$

$$p_0 = p_n, \tag{2.22c}$$

$$t_0 = 0, \tag{2.22d}$$

$$\boldsymbol{b(0)}, t_n, n \text{ free.} \tag{2.22e}$$

The boundary conditions (Eq. 2.22c) are based on examining a class of ferry solutions that follow periodic trajectories, though initial pose is a control variable. The length of the period and the discretization factor, are not explicitly constrained; they are generally chosen by the optimization solver. The initial buffer states are free variables as a result of examining repeated periodic trajectories; this will be discussed further in Chapter 3.

The cascaded solution approach presented next in Chapter 3 addresses this general persistent ferrying problem, restricted to a class of periodic solutions. The complexities of vehicle dynamics and radio frequency environments are not limited. It is only assumed that the vehicle is controllable, and that the spatial and temporal characteristics of the radio environments are known. (Chapters 5 and 6 assess the sensitivities of this method to errors in trajectory tracking and RF knowledge.)

The simulations and experiments in later chapters analyze the effectiveness of the cascaded solution as applied to a more restricted class of ferrying solutions. In these cases, hemispherical antennas are used on nodes and ferry so that aircraft orientation does not affect the RF signal. It is assumed that for storing data during transfers that memory write speeds are faster than communication rates, and are thus not a limiter in performance either. Additionally, it is assumed that the ferry can switch communication links instantaneously between nodes, with negligible delay to re-establish connections. Further for simplicity, solutions are restricted to periodic trajectories where the ferry flies at fixed speed and fixed altitude.

Chapter 3

CASCADED STRUCTURE OF DATA FERRYING

The problem defined in Chapter 2 is a high dimensional modal nonlinear optimal control problem. Rather than using a time-consuming nonlinear problem (NLP) solver to attempt this challenging problem, this chapter identifies a cascaded structure within the ferry dynamics that can be exploited, reducing complexity and computation time. Further, this cascaded methodology is proven to retain optimality, obtaining solutions as good as the original NLP solver, in less time.

3.1 Cascaded Formulation

3.1.1 Cascaded Structure of System Dynamics

The system dynamics described in Chapter 2 show how the states are coupled with each other. Buffer dynamics are functions of the link schedule control $\boldsymbol{c}$, as well as the communication rates between ferry and nodes – which are a function of aircraft position and orientation. Equation (2.3) shows the pose of the ferry as a function of the motion control $\boldsymbol{u_m}$, but not $\boldsymbol{c}$. Explicitly detailing the functional dependencies, then for an objective function J, the optimal ferrying problem is

Problem 2 (Optimal Ferrying Problem)**.**

$$[\boldsymbol{u_m}, \boldsymbol{c}]^* = \underset{\boldsymbol{u_m}, \boldsymbol{c}}{\arg\max}\ J\Big(\Gamma\left(\boldsymbol{u_m}\right), \beta\big(\Gamma\left(\boldsymbol{u_m}\right), \boldsymbol{c}\big)\Big) \tag{3.1a}$$

$$\text{subject to} \quad \text{Eqs. (2.3), (2.10)-(2.18)} \tag{3.1b}$$

$$p_0 = p_n, \tag{3.1c}$$

$$t_0 = t_n \tag{3.1d}$$

$$\boldsymbol{b}(0), t_n, n \text{ free.} \tag{3.1e}$$

3.1.2 Cascaded Solution Methodology

The separation of the motion dynamics from $\boldsymbol{c}$ in Eq. (3.1a) allows the ferry problem to be decomposed into two layers. Taking advantage of how the motion dynamics cascade through to the communication dynamics, a policy μ_c based on the ferry's path Γ can determine the optimal link schedule $\boldsymbol{c}^*$ for that path:

Problem 3 (Cascaded Ferrying Problem)**.**

$$\boldsymbol{u}_{\boldsymbol{m}}^* = \underset{\boldsymbol{u_m}}{\arg\max}\ J\Bigg(\Gamma\left(\boldsymbol{u_m}\right), \beta\Big(\Gamma\left(\boldsymbol{u_m}\right), \mu_c\big(\Gamma\left(\boldsymbol{u_m}\right)\big)\Big)\Bigg) \tag{3.2a}$$

$$\boldsymbol{c}^* = \mu_c\big(\Gamma\left(\boldsymbol{u}_{\boldsymbol{m}}^*\right)\big) \tag{3.2b}$$

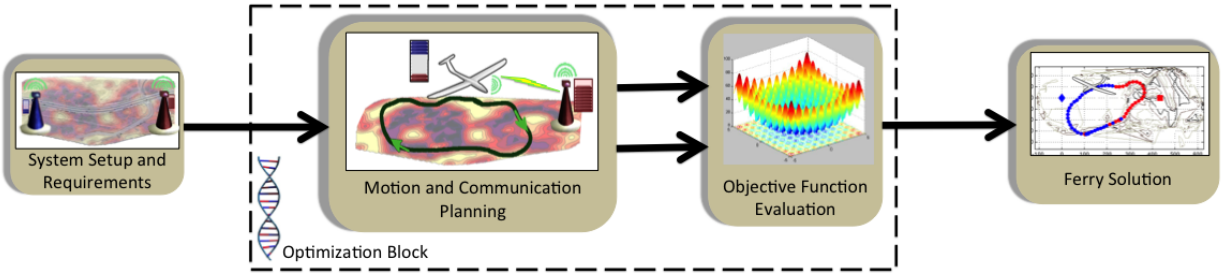
$$\text{subject to} \quad \text{Eqs. (2.3), (3.1c)-(3.1e)} \tag{3.2c}$$

$$\text{where} \quad \mu_c\left(\Gamma\right) = \underset{\boldsymbol{c}}{\arg\max}\ J_C\big(\beta\left(\Gamma, \boldsymbol{c}\right)\big) \tag{3.2d}$$

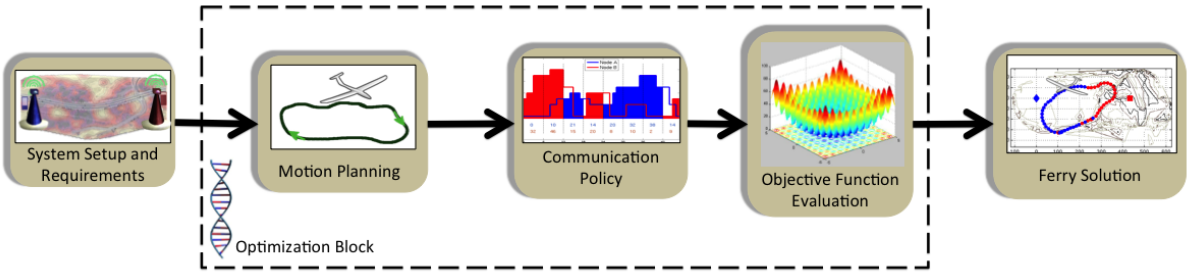
$$\text{subject to Eqs. (2.10)-(2.18).} \tag{3.2e}$$

J_C represents an objective for the ferry's link schedule control, which may differ from the overall objective J in Eq. (3.2a). The approach of these two systems is illustrated in Figure 3.1b. The former optimizes over both $\boldsymbol{u_m}$ and $\boldsymbol{c}$ (Fig. 3.1a), whereas the latter just optimizes over $\boldsymbol{u_m}$, and $\boldsymbol{c}$ is determined from the policy μ_c given $\boldsymbol{u_m}$.

This cascaded method can be thought of having two layers: at the outer layer, the motion trajectory is to be optimized, while at each step of the outer layer, the inner layer is to optimize the link schedule for the given trajectory. Chapter 4 will elaborate on how this method eases the computational burden of solving the ferrying problem.



(a) Standard method



(b) Cascaded method

Figure 3.1: Two approaches to solving the ferry problem. (a) The standard method optimizes over motion and bandwidth control variables at the same time. (b) The cascaded method uses an optimal policy based on a given trajectory to determine bandwidth control.

3.1.3 Differentiation from Coordinate Descent

The Cascaded method of Fig. 3.1b is distinctly different from coordinate descent [3]. In coordinate descent, all but one input (coordinate) is optimized while the rest are held fixed; then that input is fixed at this value while the next coordinate is optimized. The process of iteratively optimizing one free variable at a time while holding the rest fixed is repeated until convergence. For the ferrying problem, coordinate descent would involve fixing initial values for the motion control $\boldsymbol{u_m} = \boldsymbol{u_m^0}$ and iteratively stepping between the

following subproblems:

$$\boldsymbol{c}^k = \arg\max_{\boldsymbol{c}} J\Big(\Gamma\left(\boldsymbol{u_m}\right), \beta\big(\Gamma\left(\boldsymbol{u_m}\right), \boldsymbol{c}\big)\Big) \qquad \text{Given } \boldsymbol{u_m} = \boldsymbol{u_m^{k-1}} \tag{3.3a}$$

$$\boldsymbol{u_m^k} = \arg\max_{\boldsymbol{u_m}} J\Big(\Gamma\left(\boldsymbol{u_m}\right), \beta\big(\Gamma\left(\boldsymbol{u_m}\right), \boldsymbol{c}\big)\Big) \qquad \text{Given } \boldsymbol{c} = \boldsymbol{c}^k \tag{3.3b}$$

subject to the dynamic constraints of Problem 2.

However, coordinate descent may stall at a non-stationary point when dealing with non-smooth systems, and it can only be guaranteed to reach local optima in nonconvex systems. State-of-the-art methods in randomized coordinate descent are being developed for classes of nonconvex systems with up to one linear constraint [57], but not with many modal and nonlinear constraints as in our problem. Further, our cascaded solution exploits specific structure within the link scheduling problem (Eq. 3.3a) that enables efficient policies to uniquely determine $\boldsymbol{c}^k$ for a given motion control $\boldsymbol{u_m^{k-1}}$. These efficient policies are essentially folded into the ferry objective function, and since link scheduling variables $\boldsymbol{c}$ are dependent on motion control, the independent variables of the ferry problem have been reduced to $\boldsymbol{u_m}$.

3.2 Optimality of the Cascaded Method

The link schedule control policy defined in Eq. (3.2d) uses an objective function J_C that may be different from the outer layer objective. Here, the objective of maximizing data delivery along the closed trajectory is used (Eq. (3.4)). Define an arbitrary point p_0 along that trajectory to define the start t_0 and end t_n of the period.

$$\mu_c(\Gamma) = \arg\max_{\boldsymbol{c}} \sum_j \gamma_j b_j(t_n) - \gamma_i b_j(t_0) \tag{3.4}$$

for all destination nodes b_j. This section proves that this single policy will optimize the ferry system for a range of objectives for Problem 2, focusing primarily on the single source, single destination case.

3.2.1 Preliminaries

Consider periodic data ferry trajectories that transfer data from node A to node B, where the trajectory's period is t_n. Without loss of generality, let $t_0 = 0$ and $b_b(0) = 0$. The objective function J of Problem 2 generally takes one of three forms to balance the data and latency trade-offs:

(1) Given a delay or time requirement on the period of the trajectory, **maximize** the data delivered to A.

$$J_1 = b_b(t_n) \mid t_n \leq t_{req} \tag{3.5}$$

(2) Given a data requirement over one period, **minimize** the amount of time to meet it.

$$J_2 = t_n \mid b_b(t_n) \geq b_{req} \tag{3.6}$$

(3) A combination of the two: **maximize** the effective throughput over the period.

$$J_3 = \frac{b_b(t_n)}{t_n} \tag{3.7}$$

Finally, in the cascaded method, the optimal link scheduling policy μ_c **maximizes** the data delivered to A given a trajectory; and Eq. (3.4) becomes

Problem 4 (Link Scheduling Problem)**.**

$$\mu_c(\Gamma) = \arg\max_{\boldsymbol{c}} \; b_b\left(t_n\right) \tag{3.8a}$$

$$\text{subject to} \quad \text{Eqs. (2.10)-(2.18).} \tag{3.8b}$$

3.2.2 Optimality is retained for J_1

This is a straightforward proof from multivariate calculus. J_1 is the same function as the objective for the link scheduling optimization J_C, namely maximizing data delivered. Assuming candidate trajectories satisfy the delay requirement, then by the Principle of Optimality [5], maximizing a function of two variables $[u^*, c^*] = \arg\max_{u,c} J_1(u, c)$ will yield the same maximum as maximizing one given the optimal of the other, $\mu_c^* = \arg\max_{c} J_1(u^*, c)$. Specifically,

$$J_1(\boldsymbol{u_m^*}, \boldsymbol{c^*}) = J_1(\boldsymbol{u_m^*}, \mu_c^*)$$

Further, if the solution is unique, then $\boldsymbol{c^*} = \mu_c^*$; though uniqueness is not necessary to achieve the optimal performance.

3.2.3 Optimality is retained for J_2

Define a trajectory Γ as admissible if and only if there is a link schedule control $\boldsymbol{c}$ that achieves the data requirement:

$$\Gamma \text{ admissible} \Leftrightarrow \exists\, \boldsymbol{c} \mid b_b(t_n) \geq b_{req} \tag{3.9}$$

Note the cost $J_2(\Gamma) = t_n$. Since delay can be parameterized by the trajectory time, the link scheduling optimization step has only an indirect impact on this objective. However, examining this objective yields important insights about optimizing the ferry trajectory.

It is necessary to prove that if some link schedule control $\boldsymbol{c}$ achieves the required data in minimum time (satisfying J_2), then the optimal link scheduling policy $\mu_c(\Gamma)$ will also meet the data requirement in that same time. Let the optimal trajectory and link schedule be $(\Gamma^*, \boldsymbol{c^*})$ with $c^* \neq \mu_c$, and having minimum cost $J_2 = t_n$. Denote

$$b_b^*(t) = b_b(t) \;\text{ given inputs }\; (\Gamma^*, \boldsymbol{c^*}) \tag{3.10a}$$

$$b_b'(t) = b_b(t) \;\text{ given inputs }\; (\Gamma^*, \mu_c\,(\Gamma^*)) \tag{3.10b}$$

to define the data delivered along the optimal path Γ^* at time t, with the optimal $\boldsymbol{c^*}$ and policy μ_c link schedules respectively. Note that for how μ_c is defined (Eq. (3.8)), we have the condition $b_b'(t_n) \geq b_b^*(t_n) \geq b_{req}$. Thus using the link schedule of $\mu_c(\Gamma^*)$ will also meet the data requirement in time t_n and hence be optimal.

Even more, for the minimum time to be achieved, $\mu_c(\Gamma^*)$ **must** be used. Assume for contradiction that it is not used, and instead a different "optimal" policy is used, $\boldsymbol{c^*} \neq \mu_c(\Gamma^*)$ with $J_2 = t_n$. Then

$$b_b^*(t_n) \leq b_b'(t_n) \tag{3.11}$$

Suppose further that $b_b^*(t_n)$ is strictly less than $b_b'(t_n)$. Could the minimum time to meet the data requirement be reduced?

Ignoring the discretization of packet sizes (assuming we can communicate with arbitrarily small packets), then the dynamics of the buffers make $b_b(t)$ continuous and non-decreasing with respect to time. Then the minimum time trajectory will have $b_b^*(t_n) = b_{req}$. However, this means $b'(t_n) > b_{req}$. Because the buffer

is continuous and non-decreasing, then $b'(t_n) > b_{req}$ implies

$$\exists t < t_n \text{ s.t. } b'_b(t) = b_{req} \tag{3.12}$$

for which, $J_2 = t < t_n$, which means the minimum time can be reduced, contradicting our assumption of $J_2(\Gamma^*, \boldsymbol{c}^*) = t_n$ being optimal. Thus we must have $\boldsymbol{c}^* = \mu_c(\Gamma^*)$.

3.2.4 Optimality is retained for J_3

J_3 is in essence a composite of J_1 and J_2. To maximize J_3, the numerator (J_1) should be **maximized** while the denominator (J_2) should be **minimized**. Again, since the delay can be parameterized by the trajectory only, the link schedule optimization does not have a direct impact on the denominator. Section 3.2.2 showed that the decomposition retains optimality for **maximizing** J_1, thus it is also optimal for maximizing J_3.

3.2.5 Optimality of additional objectives with the conservation equations

Another popular objective for J may be to **minimize** the amount of data from B that is lost due to buffer overflow:

$$J_4 = b_{\overline{b}}(t_n) \tag{3.13}$$

Minimizing lost data relates back to the conservation equation (2.19). Integrating over the period and solving for overflow data, and dropping the arbitrary starting values of the buffers,

$$b_{\overline{a}}(t_n) = \lambda_a(t_n - t_0) - b_f(t_n) - b_b(t_n) \tag{3.14}$$

Since λ_a is a system parameter, minimizing $b_{\overline{a}}$ requires maximizing $b_f + b_b$. Note that the ferry's buffer has a fixed capacity, so the data delivered to B must be maximized, which is the objective J_1. Thus, the policy of maximizing data delivery will also minimize overflow data lost.

3.3 Conditions for Optimal Ferry Systems

This section describes specific properties of the ferrying system to derive a necessary condition for optimality. Further, relationships between average ferry throughput and allowable delay enable the ferry

problem to focus on a restricted problem of interest. Together, these enable the link scheduling policy to be formulated as a binary integer linear program (BIP), described in Chapter 4.

For a periodic set of inputs, not all of the states will be periodic. The dynamics of the overflow buffer $b_{\overline{a}}$ and delivery buffer b_b are non-decreasing, and will thus not be periodic as the ferry delivers data. Since the node buffers and ferry buffers are bounded, their infinite-horizon behavior will also be periodic. By expanding on the dynamics of the ferry's constrained buffer for the flow from A to B, a crucial constraint on the link scheduling variables is identified. A corollary is that even if not explicitly bounded, the ferry's buffer will be bounded and periodic in an optimal bandwidth allocation.

Since the ferry's buffer is periodic, $b_f(t_0) = b_f(t_n)$, meaning

$$\sum_{t=t_0}^{t_n} \Delta b_f(t) = 0 \tag{3.15}$$

Expanding the sum through the dynamic modes of Table 2.1,

$$\begin{aligned} \textstyle\sum_{t_0}^{t_P} \Delta b_f(t) \;=\; & \textstyle\sum \Delta t \lambda_a(t) && \text{over Modes II, V} \\ + \; & \textstyle\sum \Delta t c_a R_a && \text{over Modes I, III} \\ + \; & \textstyle\sum \Delta t \left(c_a R_a - c_b R_b\right) dt && \text{over Modes IV, VI} \\ - \; & \textstyle\sum \Delta t c_b R_b && \text{over Modes VII, VIII.} \end{aligned} \tag{3.16}$$

Consolidating terms and combining with Eq. (3.15) results in

$$\sum_{\{II,V\}} \lambda_a \Delta t + \sum_{\{I,III,IV,VI\}} c_a R_a \Delta t = \sum_{\{IV,VI,VII,VIII\}} c_b R_b \Delta t \tag{3.17}$$

The left side of Eq. (3.17) represents the data collected from node A, and the right side represents the amount delivered to node B.

Theorem 1. *Equation* (3.17) *is a necessary condition for a ferry's periodic trajectory to be optimal in the sense of Problem 2 for objectives* J_1 *through* J_4.

Proof. Suppose Eq. (3.17) does not hold. If the left side is greater, the ferry is collecting more data from A than it is delivering to B during a single period, and b_f grows as the trajectory is repeated. In the resulting limit cycle, b_f will grow to b_f^{max} and the system will be in modes VII and VIII, where the ferry can no

longer collect data from A. There are two alternatives that would improve performance: either switch the bandwidth allocation or shorten the trajectory in specific regions (and perhaps both).

First, there is some amount of time t where $c_a(t) = 1$ and therefore $c_f(t) = 0$, but inverting the allocation would better serve the system. If there is a time t where $c_a(t) = 1$, but $R_b(t) > 0$, then switching the allocation to $c_b(t) = 1$ will result in more data delivered (with no increase in period time). Second, with the ferry's buffer filled to b_f^{max}, the system will have some time t where the system is in mode VII or VIII, but $c_a(t) = 1$. Listening to A while the ferry's buffer is full is a poor use of time. If $R_b(t) > 0$, switch the allocation control to deliver data as above. If instead $R_b(t) = 0$, then the ferry's motion can be shortened slightly around this time, shortening the period with no decrease in amount of data delivered.

A similar argument can be made in the case that the right side of Eq. (3.17) is larger, where the ferry is trying to deliver more data to B than it gathered from A. Over repeated cycles, its buffer will be completely empty. The system would be better served with $c_a = 1$ for a longer portion of time, collecting more data from A to then deliver to B, improving throughput. The ferry's motion could also be shortened around a time when the system is in modes I, II, or III, but $c_b = 1$; attempting to deliver data from an empty buffer. This improves system latency with no decrease in data delivered. □

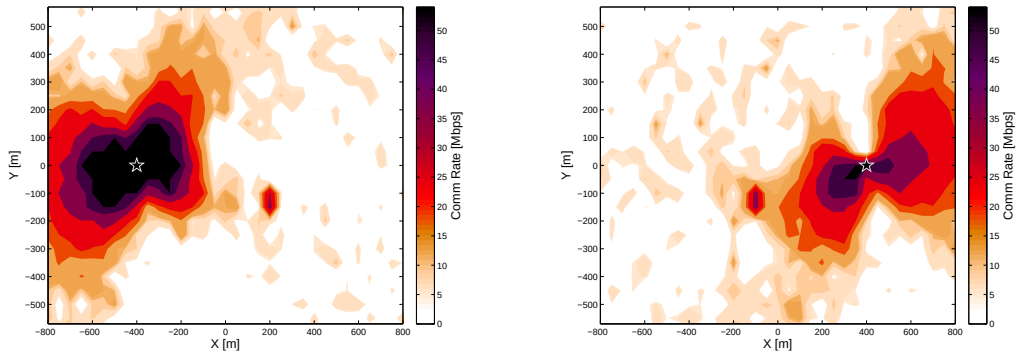
An equivalent argument can be made for additional data flows between more nodes. Theorem 1 is independent of both the vehicle dynamics and communication models. This condition can be added as a constraint to the bandwidth allocation problem (Prob. 4), and in particular allows us to design fast algorithms to solve it.

3.4 Impact on Computing Ferry Solutions

This chapter has so far introduced the cascaded approach to determining ferry solutions, and proving that the global optimal solution is retained in the new approach. Further, derived necessary conditions enable the Link Scheduling Problem (Prob. (4)) to be determined with fast, near-optimal heuristics, developed and analyzed in Ch. 4. This section demonstrates how the cascaded method can save computation time over the traditional, non-cascaded approach to determine data ferrying solutions.

3.4.1 Optimization Setup

In these scenarios, an unmanned aircraft is tasked with ferrying data from A to B over 802.11g channels. The objective function is the effective throughput J_3, limited by a soft constraint of a 90-s-period duration. Though the cascaded approach works regardless of vehicle dynamics, this study fixes the altitude and speed of the aircraft for the sake of visualization of results. The aircraft flies at a constant speed of 25 m/s with a minimum turning radius of 60 meters (similar to the profile of the Tempest unmanned aircraft platform [20]), and at an altitude of 100m above then nodes. The wireless channel characteristics were generated for nodes A and B based on theoretical dipole antennas in an environment with several interferers and additive Gaussian background noise. These channel environments are shown in Fig. 3.2, which, though simulated, reflects environments similarly seen during flight experiments [68].



(a) Node A environment

(b) Node B environment

Figure 3.2: Simulated RF environments between the aircraft and the nodes, based on dipole antennas in a noisy environment.

A genetic algorithm (GA) is used to solve four different problem formulations of the non-convex ferrying problem: (1) Cascaded-LP, (2) Cascaded-Knapsack, (3) Cascaded-MDF and (4) Non-Cascaded. The three cascaded algorithms follow the flow of Fig. 3.1b, while the Non-Cascaded algorithm follows Fig. 3.1a. (The three cascaded policies will be detailed in the next chapter; most relevant here is that they are three different fast heuristic policies to approximate solutions to the Link Scheduling Problem.) Each

algorithm uses the same motion control, composed of 3 parts:

$$\boldsymbol{u_m} = [p_0, \boldsymbol{\omega}, \boldsymbol{\tau}]^T \tag{3.18}$$

controlling the ferry's initial pose, a vector of 8 turn rate commands $\boldsymbol{\omega}$, and the 8 associated times $\boldsymbol{\tau}$ to maintain each turn rate. The ferry's kinematics (Eq. (2.3)) follow a first-order forward difference method. To ensure the closed-path constraints of Eq. (3.1) are satisfied, a shortest-Dubins path [19] is added to connect the ferry's pose at the end of the 8 turn rate commands back to p_0. The ferry's path is divided into 45 segments, corresponding to a maximum of 2 s intervals, each of which having allocation variables $\boldsymbol{c}$ needing to be determined.

The four algorithms differ in how the communication allocation variables are handled. For the cascaded algorithms, the chromosomes of their GA are just the components of $\boldsymbol{u_m}$, each chromosome determining a single ferry path. For each path, the cascaded methods apply a policy to determine the corresponding allocation variables: (1) the Cascaded-LP solves the linear programming relaxation of Problem 5 (Sec. 4.1); (2) the Cascaded-Knapsack implements the Greedy Knapsack heuristic (Sec. 4.3); and (3) Cascaded-MDF implements the Most Data First algorithm (Sec. 4.4) . The Non-Cascaded Algorithm explicitly controls the allocation variables $\boldsymbol{c_a}$ for each segment of the ferry's path, setting $\boldsymbol{c_b} = \boldsymbol{1} - \boldsymbol{c_a}$ to satisfy Eq. (2.11). The GA thus has a much larger chromosome form including all of $\boldsymbol{u_m}$ and $\boldsymbol{c}$.

The specific parameters that control chromosome evolution in the genetic algorithm are summarized in Table (3.1), in particular the rate at which various parts of the chromosomes are mutated and merged (see [77] for further details on the workings of GA). These were manually tuned to values that appear to work well for data ferry trajectory optimization. For consistency among all 4 algorithms, all GAs use population sizes of 50, and are run for 2×10^5 generations. Implemented in MATLAB on a 64-bit Intel Xeon Quadcore CPU at 3.0 GHz each, each run of the GA took between 6 hours and 40 hours, depending on which method was used. Because of the randomness inherent to genetic algorithms, each solver (Cascaded-LP, Cascaded-Knapsack, Cascaded-MDF, and Non-Cascaded) was run multiple times, with performance averaged to obtain an accurate representation of performance.

Table 3.1: Genetic Algorithm Parameters

Cascaded Chromosomes	ω_k, τ_k, p_0
Non-Cascaded Chromosomes	$\omega_k, \tau_k, p_0, c_{a,k}$
Population Size	50
Elite Size	2
Mutation Rates for ω_k, τ_k	0.40
Mutation Rates for p_0, c_{a_k}	0.35
Crossover Rates	0.65

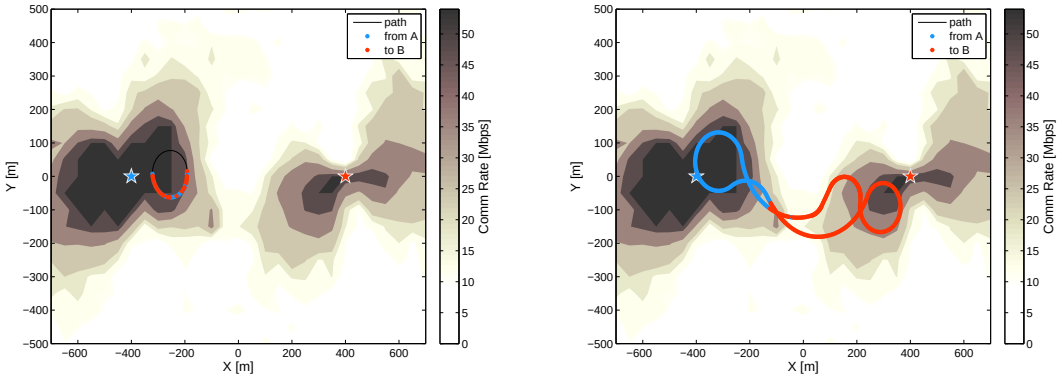
3.4.2 Results

The final ferry trajectories determined by each GA are shown in Figs. 3.3 and 3.4, along with the associated link allocations. Blue segments represent receiving data from A, and red corresponds to transmitting data to B. The three cascaded-based trajectories achieve an effective throughput between 17.7 and 18.5 Mbps, arching through the higher communication regions of the environment to achieve such a high rate. The higher-dimensional Non-Cascaded genetic algorithm was only able to reach a trajectory that achieves 4.9 Mbps within the 2×10^5 generations.

Note that the arched paths determined in the cascaded solutions would not be captured when the dynamics are overly simplified, as in Traveling Salesman-based solutions [6, 37]. Further, because of the highly variable nature of this environment, the best allocation is, as expected, not just a simple Voronoi Partition . Both of the fast heuristics are able to alternate its allocation much more frequently to take advantage of the various spikes in bandwidth. This further illustrates the benefit of communication control that can account for the complexities of such environments, and further take advantage of such complexities.

Figure 3.5a shows how the throughput of the ferrying system improves through the generations of the genetic algorithm. Because the Non-Cascaded method needs to explicitly control the link allocations, it has a much larger set of independent variables. As a result, the solution reaches a throughput of 4.9 Mbps after 150,000 generations, while both cascaded solutions require less than 10 generations to find a much better solution. After just 10,000 generations, the cascaded solution hits over 16 Mbps.

With explicit control over the allocations, evaluating the link scheduling allocation for the Non-Cascaded method runs in $\mathcal{O}(n)$ time. Whereas the MDF algorithm took 1.35 ms to determine and evaluate an allocation, the Non-Cascaded method evaluates an allocation 10 times faster. As a result, each generation



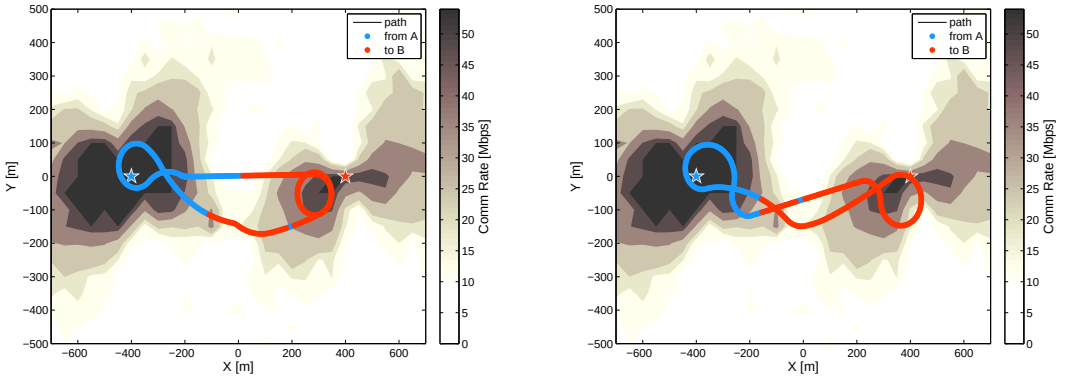
(a) Non-Cascaded (b) Cascaded-LP

Figure 3.3: Ferrying trajectories determined from optimal non-cascaded and cascading methods after 2×10^5 generations.

of the GA takes less time to evaluate for the Non-Cascaded method. The comparison of computation times is shown in Fig. 3.5b. The convergence rate of the Non-Cascaded method, however, is too slow for the evaluation factor to catch up to either cascaded solution. After only 5 s, both the Cascaded-Knapsack and Cascaded-MDF and methods have ferrying solutions already 3 times better than the Non-Cascaded solution at that point. In fact, the Cascaded-MDF solution at 2 s is better than the Non-Cascaded method has found after 20 hours. This highlights the curse of dimensionality for the Non-Cascaded method, and emphasizes the speed of the Cascaded method for data ferrying.

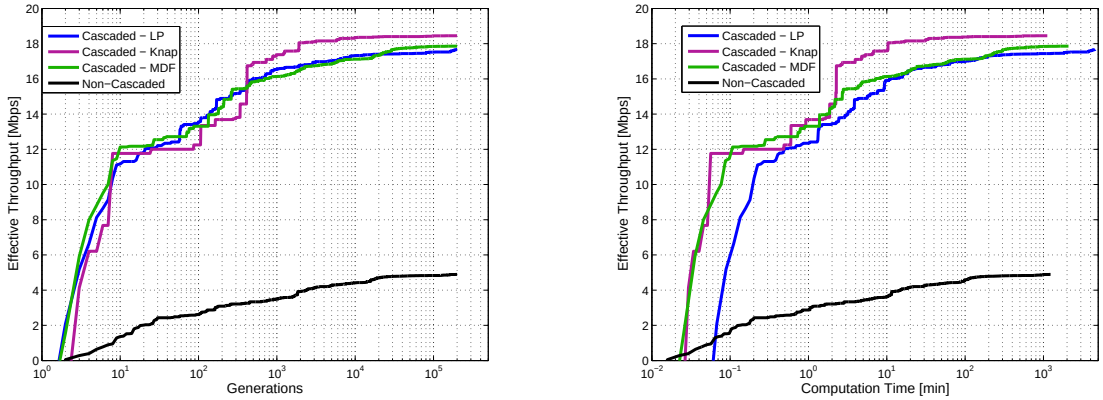
3.4.3 On Optimality within the Simulation Results

Section 3.2 proved that global optimality is retained between cascaded and non-cascaded methods, i.e. that the global optimal solution of both methods achieve the same throughput performance (with the same motion and link scheduling controls in the event of unique solutions). However, the gap between cascaded and non-cascaded solutions is large even after 200k generations of the genetic algorithm, and convergence to the same solution is not apparent. There are two key factors this gap illustrates about the solution approach in this simulation: (i) the dimensionality issues present in the non-cascaded approach leads to much slower solution improvement, and (ii) the inherent random nature of a genetic algorithm only guarantees the optimal solution will be reached in infinite time (discussed more in Sec. 3.4.4).



(a) Cascaded-Knapsack (b) Cascaded-MDF

Figure 3.4: The best ferrying trajectories determined by near-optimal policies after 2×10^5 generations.



(a) Averaged effective throughput versus generations (b) Averaged effective throughput versus time

Figure 3.5: Performance of Cascaded and Non-Cascaded ferry solutions, averaged over 8 runs each.

The solution at the end of Fig. 3.5 is simply the best solution found after letting the GA run for 200k generations. The high dimensionality in the ferrying problem means much more time may be required for the non-cascaded method to converge to the optimal ferrying solution. A ferry solution from one of the cascaded solvers is, of course, feasible for the non-cascaded solver, and seeding the non-cascaded solver with such a solution will close the gap; however, further evolution performance beyond such an intermediate solution will still grow more slowly for the high-dimensional non-cascaded method.

3.4.4 On the Use of a Genetic Algorithm

In a genetic algorithm, solutions are able to evolve over generations without the use of gradient information; however, the inherent randomness in a GA means convergence to the global optimal solution is only guaranteed as the number of generations goes to infinity [77]. Faster convergence may be achieved with gradient-based optimization techniques, such as Sequential Quadratic Programming (SQP) as used in a simplified data relay positioning scenario [24]. SQP forms sequential relaxations of the optimization problem to approximate gradients and provide iterative search directions in a process similar to Newton and quasi-Netwon methods; however this approach may suffer from getting stuck in local optima. The main advantage of the GA for ferrying is its ability to improve solutions in the face of non-smooth, non-convex objectives present in the data ferrying problem.

Simultaneous Perturbation Stochastic Approximation (SPSA) and Simulated Annealing (SAN) both use gradient approximation techniques to achieve faster convergence than a GA, with fewer function evaluations than SQP; they also incorporate randomness to improve their ability to avoid local optima in non-convex problems [67]. SPSA has been proven to converge to global optima over differentiable non-convex cost functions, and its gradient approximation can handle non-smooth convex functions [47]. Further, extensions have been demonstrated to discrete control variables over convex cost functions [26]. However, theoretical performance of SPSA to converge toward a global optimum has not been characterized for discrete variables over non-smooth, non-convex problems.

If successfully applied to data ferrying, the approximate gradient information could result in faster solution improvement for SPSA over GA. The significance of the cascaded method, however, is that the number of independent variables is reduced. The near-optimal link scheduling policies effectively reduce the problem search space to a manifold where, for each independent motion control, the link schedule control variables are fixed to a near-optimal set values determined by the near-optimal link scheduling policy. Further exploration of cost function's smoothness and convexity along this manifold is needed to determine which optimization technique, e.g. GA or SPSA, would converge fastest. If either sufficiently smooth or convex,

and SPSA can be effective, the gap between Cascaded and Non-Cascaded solutions under SPSA should show similar trends to the gap in Fig. 3.5.

3.5 Summary

The cascaded approach significantly simplifies the ferry planning process. At the high level, it hides the modal buffer dynamics. Additionally, the ferry problem's dimension is reduced with only the motion control as an independent input. This results in determining good ferry solutions orders of magnitude faster than the Non-Cascaded approach. Using a genetic algorithm, the solution obtained by the traditional approach after 20 hours was surpassed by cascaded approaches after less than 5 seconds; and cascaded approaches continued to improve to a solution more than 3 times better after 2 minutes of computation.

However, encapsulating the link control obfuscates the gradient information $\partial J/\partial \boldsymbol{u^m}$. This means any outer layer trajectory optimization tool will need to make many function evaluations (for example using direct methods or estimating gradients [29]). Further, evaluating the objective function J is not a trivial calculation, as it requires the inner optimization (3.4). In the interest of keeping the planning process fast, Chapter 4 develops and analyzes algorithms to determine near-optimal solutions of the inner optimization quickly. Incorporating such algorithms into the cascaded solution shows the significant savings computation time to determine good ferrying solutions.

Chapter 4

LINK SCHEDULING POLICIES

Data ferrying is interested in far more than the stationary single-source, single-destination case. With much more complicated problems in mind (i.e. many sources, many destinations, varying data priorities, multiple ferries [65, 71, 80, 84]), computation time to solve such problems will grow with the problem's complexity. Though the cascaded methodology reduces computation time, further reductions will be necessary to solve these more complicated problems and field their solutions. As the cascaded solution method requires many function evaluations of the communication policy, this chapter examines the link-scheduling subproblem (Prob. 4).

In a networking context, link scheduling is closely related to medium access control. Generally, medium access control is achieved through link scheduling that is designed from an end-node perspective. In most cases, decentralized policies are developed to minimize the likelihood of collisions with other end users [34]. Various protocols such as Time-Division Multiple Access (TDMA) [52] or Carrier Sense Multiple Access (CDMA)[75] attempt to balance energy consumption with network overhead to minimize probabilities of collisions, but not optimizing total network throughput. In contrast, the data ferry concept described here has a centralized single-access scheme; therefore there cannot be collisions and throughput can be freely optimized.

Other literature for centralized schemes only handle networks where the RF environment is constant between communicating nodes [21, 73], which is not the case for the mobile data ferry. This chapter first formalizes the link scheduling problem, and describes several heuristic approaches commonly used in data ferrying literature. These heuristics are based on restricted scenarios with simple RF environments. The

key contribution of this chapter is the development and analysis of two novel fast, near-optimal heuristics to the ferry link-scheduling problem: the Greedy Knapsack Heuristic, and the Most-Data-First algorithm. These are detailed in the context of the single-source, single-destination case; extensions to multiple-source scenarios are described as well. These, along with the commonly used heuristics for data-gathering and ferrying problems, are assessed in terms of achievable throughput and computation time. Finally, the scalability of each heuristic with respect to number of source nodes is evaluated in simulation.

4.1 Link Scheduling as Combinatorial Optimization

Consider optimizing link scheduling for a single data ferrying aircraft that follows a closed (periodic) path $\boldsymbol{p}$, transferring data from a single stationary source node A to a stationary destination node B over a single wireless radio channel. The necessary condition of Theorem 1 constrains the link scheduling policy, forming a Binary Integer Linear Program (BIP). The policy is given a discretized set of points to represent the closed path $\Gamma = \{p_k\}$ of time $\boldsymbol{t} = \{t_k\}$, $k \in [1, n]$, where each segment k goes from point p_{k-1} to p_k. The spatio-temporal dependent sequence of SNIR along each segment $S_a(\boldsymbol{t})$ and $S_b(\boldsymbol{t})$ is given as well, determining potential communication rates $R_a(\boldsymbol{t})$ and $R_b(\boldsymbol{t})$ through Eq. (2.8).

Define $d_{a,k} = R_a(t_k)\Delta t_k$ as the amount of data that the ferry can collect from A during segment k, and $d_{b,k} = R_b(t_k)\Delta t_k$ as the amount the ferry could deliver to B during the same segment. Ideally, segments would be defined sufficiently small to assume the radio channel is stationary over each segment. However if resolution of the SNIR environment is higher than the resolution of the path, and if SNIR is not stationary along each segment, then $d_{a,k}$ can be calculated by integrating integrating Eq. (2.8) over segment k. Figure 4.1 shows such an example scenario with communication rates between ferry and ground nodes plotted against the ferry's time along a 48-second path, with segments defined by 6-second intervals, and data potentials $\boldsymbol{d_a}, \boldsymbol{d_b}$ calculated across the segments. Knowing this path and the RF environment in advance enables the allocations of each segment to be determined off line (meaning segment allocation decisions do not have to be made in time order).

Let the column vectors $\boldsymbol{d_a} = [d_{a,1}, ..., d_{a,n}]^T$ and $\boldsymbol{d_b} = [d_{b,1}, ..., d_{b,n}]^T$. The objective is to allocate links among the two radio channels to maximize the amount of data delivered to B. Let the decision variable

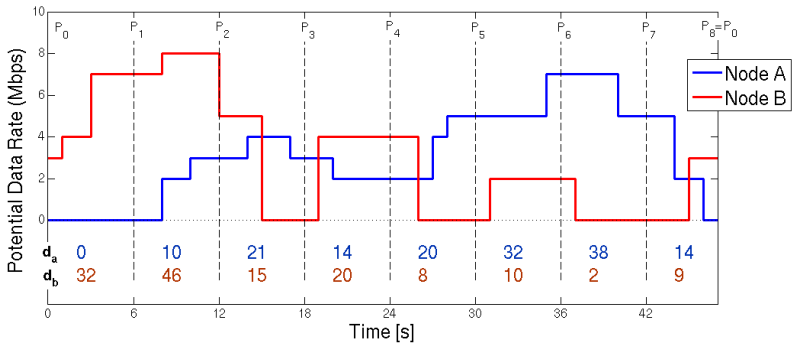


Figure 4.1: Example Link Scheduling scenario with data potentials to each node varying over the ferry's path.

$\boldsymbol{z} = [z_1, ..., z_n]^T$ where $z_k = 0$ if the ferry should deliver data to B along segment i, and $z_i = 1$ if the ferry should listen to A along segment i. The link scheduling problem (LSP) can be formulated the following binary integer linear program:

Problem 5 (Link Scheduling ILP)**.**

$$\boldsymbol{z}^* = \arg\max_{\boldsymbol{z}} \boldsymbol{d_a}^T \boldsymbol{z} \tag{4.1a}$$

$$s.t. \quad \boldsymbol{d_a}^T \boldsymbol{z} = \boldsymbol{d_b}^T (\mathbf{1} - \boldsymbol{z}) \tag{4.1b}$$

$$\boldsymbol{z} \in \{0, 1\}^n \tag{4.1c}$$

where $\mathbf{1}$ is a column vector of ones of length n. The equality constraint (4.1b) results from Theorem 1. This equality constraint is where the ferrying link scheduling problem varies from standard link scheduling problems. The standard LSP needs to divide a fixed amount of time among competing nodes (segments in our case) – and so the right side of (4.1b) would be a fixed value. For the ferrying problem, both sides of the equality are functions of the allocation $\boldsymbol{x}$ – the competing segments need to be allocated such that the amount of data delivered to B has been collected from A. However, we next show how the problem formulation can be manipulated to still take advantage of known link scheduling methods.

The LSP is a form of combinatorial optimization problem that is known to be NP-Hard [23, 56, 70]. As a result, this chapter develops and analyzes heuristic algorithms that have both great ferrying performance and quick computation time.

4.2 Common Heuristics

In data ferrying literature, several common heuristics are applied to the Link Scheduling Problem. These heuristics are developed from restricted communication environments that Traveling Salesman paradigms place on the ferrying problem. In scenarios where significant assumptions are appropriate, e.g. symmetric, smooth decaying radio signals or disk model communication protocols, these heuristics tend to work well. However, they lead to poor performance when such assumptions are not met by the environment. These common heuristics form the basis for comparisons to the algorithms developed later in this chapter, which are shown to work well regardless of such restrictions.

4.2.1 Closest Node

A standard heuristic is simply to talk to whichever node is physically closest to the ferrying aircraft. The Closest Node scheme is a quick partitioning algorithm that only depends on knowledge of ferry and node locations, and not based on knowledge of the data potentials (Fig. 4.1). At each segment along the given path, allocate to whichever node is physically closest. A calculation of distances to each node is all that is required of each segment. As the number of nodes increases, so do the required comparisons. For n segments and $s + d$ source and destination nodes, this algorithm runs in $\mathcal{O}((s + d)n)$ time.

4.2.2 Best RF

The Closest Node heuristic stems from the assumption that radio signal strength decays smoothly with distance, and further that both nodes have equal radio power parameters. To account for more complicated communication environments, such as with directional antennas and interferers, the Best RF heuristic is commonly used: at each segment, allocate to whomever has the strongest signal , which is solved in $\mathcal{O}((s + d)n)$.

This works well for the Data-Gathering Problem (DGP) [42, 58] , where the goal is to collect as much data from nodes as possible. The gatherer does not need to worry about fairness, where the data specifically comes from, or if data collected is balanced with data delivered. With no consideration for Eq. (4.1b), this scheme leads to potentially very unbalanced allocations in the ferrying problem, where the ferry may

attempt to deliver much more data than it collected, or spend too much time collecting data that will never get delivered. For the example scenario of Fig. 4.1, the resulting Best-RF allocation (Fig. 4.2) collects 125 Mb from node A but is only able to deliver 98 Mb to node B, [1] leaving a large gap from satisfying the equality in Eq. 4.1b.

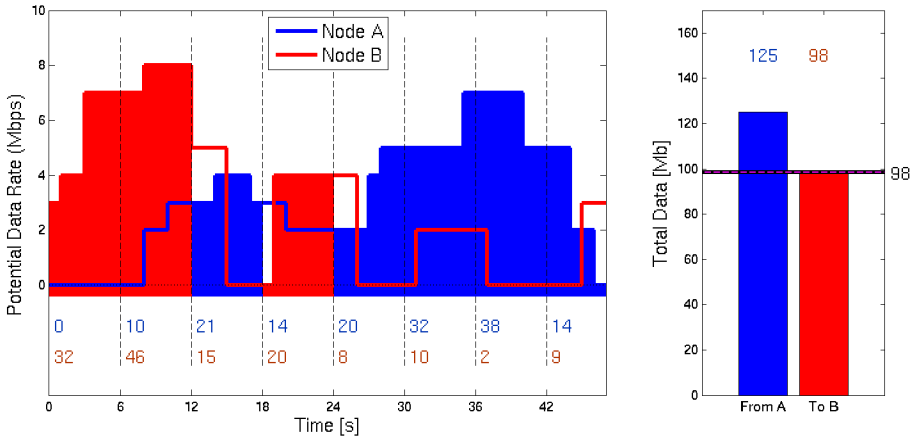


Figure 4.2: The Best RF allocation for the example scenario, showing unbalanced data transfer between the ferry and nodes.

4.2.3 Switching Point

More elaborate than Best RF, the Switching Point (SP) scheme [6, 18, 37] partitions the segments into two contiguous sections, explicitly trying to meet Eq. (4.1b). Segments k_1 to k_2 define the two allocation switching points; as the ferry follows the trajectory, it switches from collecting data from A to delivering to B at segment k_1, and switches back to collecting more data after segment k_2. The Switching Point scheme is optimal in the following case. Let the switching points be defined as segments k_1 and k_2, where segments $k \in [k_1, k_2]$ are allocated to B, and $k \notin [k_1, k_2]$ allocated to A. Then the switching point allocation is optimal

[1] Recall that this solution represents a single period to the persistent ferrying solution, so the long-term behavior of this allocation does not represent attempting to deliver any data to node B before collecting any from node A.

for a given trajectory if and only if

$$d_{b,i} \geq d_{b,j} \tag{4.2a}$$

$$d_{a,i} \leq d_{a,j} \tag{4.2b}$$

$$\forall\, k \in [k_1, k_2], \forall\, k \notin [k_1, k_2] \tag{4.2c}$$

This is the case when data potentials decay monotonically with distance. In this case, any pair of segments (i, j) as defined that swapped allocations would result in worse data throughput for both nodes.

Figure 4.3 shows the resulting Switching Point allocation for the example scenario. The allocations are partitioned with switching points defined by $k_1 = 0s$ and $k_2 = 24s$. Though the data potentials do not follow convex trends (i.e. monotonically decreasing in the event that the ferry is moving at constant speed, and closest to node A at time t=36 and to node B at t=11), this policy does well. The ferry collects 104 Mb from node A and allocates 113 Mb to delivering to node B, yielding a total data transfer of 104 Mb.

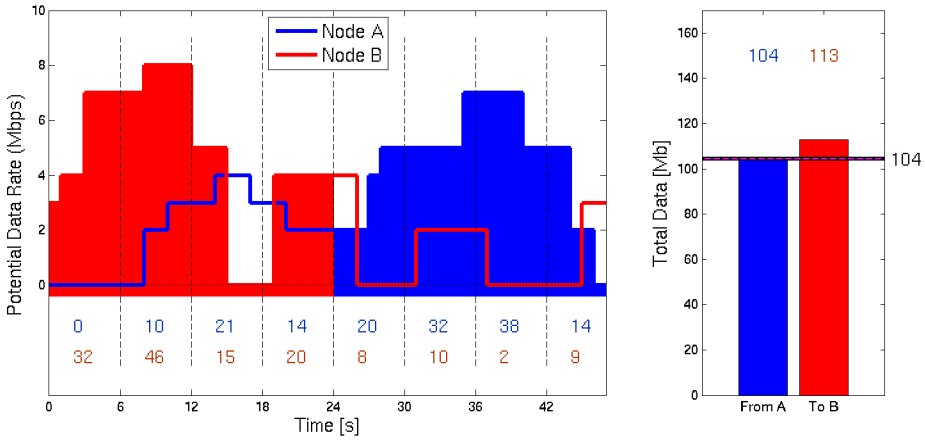


Figure 4.3: The Best RF allocation for the example scenario, showing unbalanced data transfer between the ferry and nodes.

Note that there are n^2 possibilities of switching locations for this scheme, but only n combinations meet the balanced allocation requirement in Eq. (4.1b). Defining the first switching point k_1 then determines the segment k_2 to switch back. So for the single-source, single-destination scenario, this algorithm runs in linear time, $\mathcal{O}(n)$. However, this run time is unique to the case of one source node and one destination. For s source nodes and one destination, there are s free switching locations. The equality constraint reduces the

destination's switching point to at most one feasible location between each pair of source switching points, for a total of s destination switching point options. Thus, the general run time of the Switching Point heuristic for s sources and one destination is $mathcalO\,(s\,n^s)$

4.2.4 Upper Bound Solution

An exact optimization program for the Link-Scheduling Problem should provide a bound on solution performance. Typical binary integer programming uses a combination relaxation and heuristic techniques in an attempt to reduce the problem as well as guide branch-and-bound and plane-cutting techniques to converge to an optimal solution [23, 56]. However, efficient deterministic algorithms that guarantee optimality do not exist for this class of problem. Instead, a relaxed form of the optimization problem is solved to provide an upper bound on solution performance.

By relaxing the integer constraint on the link allocation variable $\boldsymbol{x}$, allowing it to take any value in $[0, 1]$, this relaxed LSP is solved with a linear program (LP). LPs can take advantage of active-set and simplex methods to determine optimal solutions, achieving run times of $\mathcal{O}(n^3)$ [43]; MATLAB's *linprog* solver is used here [83]. The LP relaxation allows for fractional allocations of each segment, enabling the equality constraint 4.1b to be satisfied exactly, rather than the lower performance of the inequality 4.4.

Though the solutions to this relaxation are not generally attainable for the ferrying system, it provides a fast and generally close upper bound on performance. The LP relaxation exhibits the *greedy choice property* [41], meaning the the LP solution will only contain one non-integer element of $\boldsymbol{x}$. The LP upper bound solution for the example scenario is shown in Fig. 4.4. This solution exactly satisfies Eq. 4.1b by sharing segment 3 among both nodes, with 25% going to A (collecting 5.25 Mb in that segment), and 75% going to node B (delivering 11.25 Mb). [2] This results in a total data transfer of 109.25 Mb.

[2] This fractional split in bandwidth is effectively uniformly across the 6-second segment, rather than splitting in time, as the data potentials fed to the LP do not data rate changes within each segment.

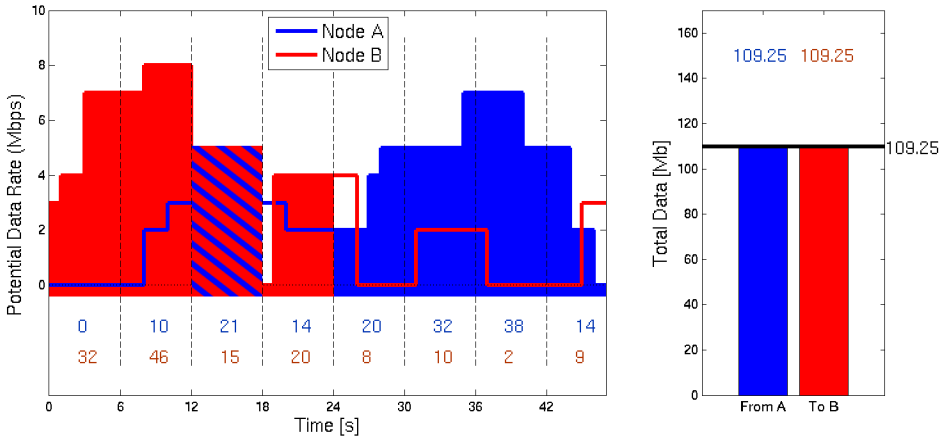


Figure 4.4: The Linear Programming solution for the example scenario, sharing segment 3 among both nodes to deliver exactly as much data as is collected.

4.3 The Knapsack Heuristic

4.3.1 The Knapsack Problem

The Knapsack Problem is a well-studied problem in combinatorial optimization [41]. The standard instance has n items of value $\boldsymbol{v} = \{v_k\}$ and weight $\boldsymbol{w} = \{w_k\}$ for $k = 1$ to n. The objective is to choose the best subset of items to pack in the knapsack without exceeding the knapsack's capacity c:

Problem 6 (Knapsack Packing Problem)**.**

$$\boldsymbol{z}^* = \arg\max_{\boldsymbol{z}} \boldsymbol{v}^T \boldsymbol{z} \tag{4.3a}$$

$$s.t. \quad \boldsymbol{w}^T \boldsymbol{z} \leq c \tag{4.3b}$$

$$\boldsymbol{z} \in \{0,1\}^n \tag{4.3c}$$

As this optimization problem is NP-Hard, several efficient approximation algorithms have been developed for this NP-Hard optimization problem. This section adapts the knapsack problem to link link scheduling, and applies a fast greedy knapsack heuristic to the LSP. The greedy heuristic from the knapsack problem can be shown to be arbitrarily bad in terms of performance, but characteristics of the ferrying problem enable the greedy heuristic to avoid this worst case scenario and in fact approach the performance of the optimal linear relaxation solution.

4.3.2 Link Scheduling as a Knapsack Problem

The Link Scheduling Problem can be manipulated to become a knapsack problem, allowing a link scheduling policy to take advantage of knapsack solutions. The difference between (6) and (5) lies in the constraints. First, Eq. (4.1b) is an equality constraint. However, in a discretized space, it may be infeasible to obtain an equality. Relaxing the quality to an inequality, with practicality dictating that the ferry should not deliver more than it collects, the constraint becomes

$$\boldsymbol{d_a}^T \boldsymbol{x} \leq \boldsymbol{d_b}^T (1 - \boldsymbol{x}) \tag{4.4}$$

More significantly, both sides of Eq. (4.1b) depend on the binary control variable $\boldsymbol{x}$, whereas c in Eq. (4.3b) is fixed. Since dot products distribute over addition, Eq. (4.4) can be rearranged:

$$\boldsymbol{d_a}^T \boldsymbol{x} \leq \boldsymbol{d_b}^T \cdot \mathbf{1} - \boldsymbol{d_b}^T \boldsymbol{x} \tag{4.5a}$$

$$\boldsymbol{d_a}^T \boldsymbol{x} + \boldsymbol{d_b} \boldsymbol{x} \leq \boldsymbol{d_b}^T \cdot \mathbf{1} \tag{4.5b}$$

$$(\boldsymbol{d_a} + \boldsymbol{d_b})\, \boldsymbol{x} \leq \boldsymbol{d_b}^T \cdot \mathbf{1} \tag{4.5c}$$

It is now of the same form as the knapsack problem, with $\boldsymbol{v} = \boldsymbol{d_a}$, $\boldsymbol{w} = \boldsymbol{d_a} + \boldsymbol{d_b}$, and $c = \boldsymbol{B} \cdot \mathbf{1}$.

4.3.3 Greedy Knapsack Heuristic

In the standard knapsack problem, an item's efficiency is defined as the ratio of its value to its weight, $e_k = v_k / w_k$. The greedy algorithm then selects items in order of efficiency until the knapsack's capacity is filled. For the LSP, efficient segments are

$$e_k = \frac{d_{a,k}}{d_{a,k} + d_{b,k}} \tag{4.6}$$

Higher efficiency segments are those where the ferry can deliver a lot of data to A, but could only receive a very small amount of data from B. The greedy knapsack algorithm for ferrying begins by allocating every segment to B. Then, in order of efficiency, segments are switched from B to A. Figure 4.5a shows the first allocation in the example scenario, and Fig. 4.5b shows the fourth. Switching allocations from B to A continues until the capacity condition is no longer satisfied. The final segment, where switching to A

violates the capacity condition is the decision item, found at the fifth allocation (segment 3) in Fig. 4.5c. The algorithm then compares the ferrying performance of the case where the decision item is allocated to B (Fig. 4.5b at 104 Mb) with it being allocated to A (Fig. 4.5c at 98 Mb), and chooses the better of the two as the final allocation.

The running time of the greedy algorithm, once sorted by efficiency, is linear in the number of segments, $\mathcal{O}(n)$; for a large number of segments with close, small data potentials, the capacity condition will end the Greedy Knapsack allocation at roughly half the segments. The need to sort segments increases the running time to $\mathcal{O}(n \log n)$. In terms of throughput performance, the knapsack's greedy algorithm can be shown to be arbitrarily bad (see Appendix A.2); however, it is provably optimal in a linear program relaxation [41]. This represents the case where segments can be allocated fractionally between A and B rather than exclusively one or the other. This means the greedy heuristic will approach the optimal allocation as the ferry trajectory's segment size decreases. Segment size thus represents a trade-off between the greedy algorithm's running time and the optimality of its solution.

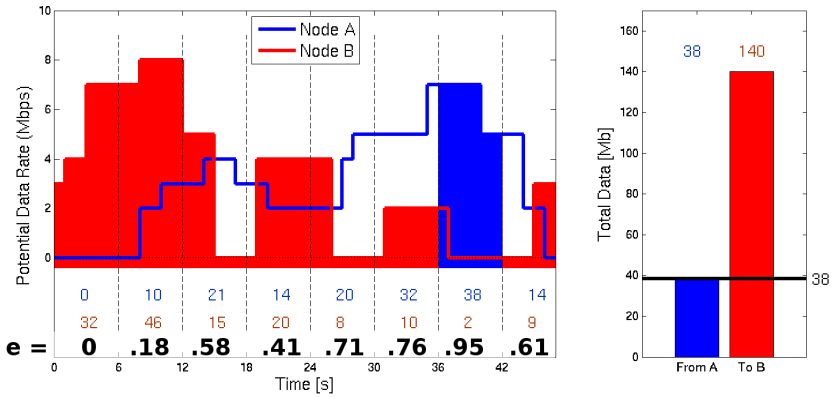
4.3.4 Greedy Knapsack for Multiple Sources

This same segment efficiency idea can be extended to the case of many source nodes delivering data to a single destination node. For destination node B and source nodes A_i, $i \in 1, 2, ..., s$, and segments $k \in 1, ..., n$, data potentials are now $\boldsymbol{d_b}$ and $\boldsymbol{d_{ai}}$. The decision variables are represented by a series of vectors $\boldsymbol{x_i} = \{x_{ik}\}$, made up of binary variables where a value of 1 represents allocating segment k to node A_i.

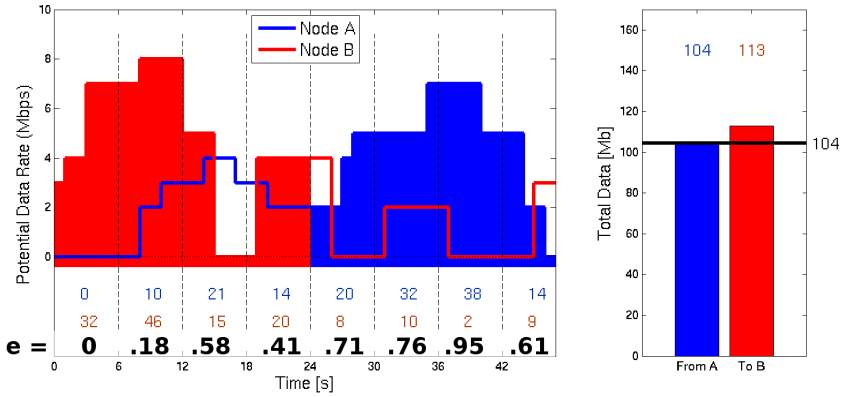
$x_{ik} \in 0, 1$ like before, but now there's an additional constraint that each segment can be allocated to at most one source node:

$$\sum_i x_{ik} \leq 1 \quad \forall k \in 1, ..., n \tag{4.7}$$

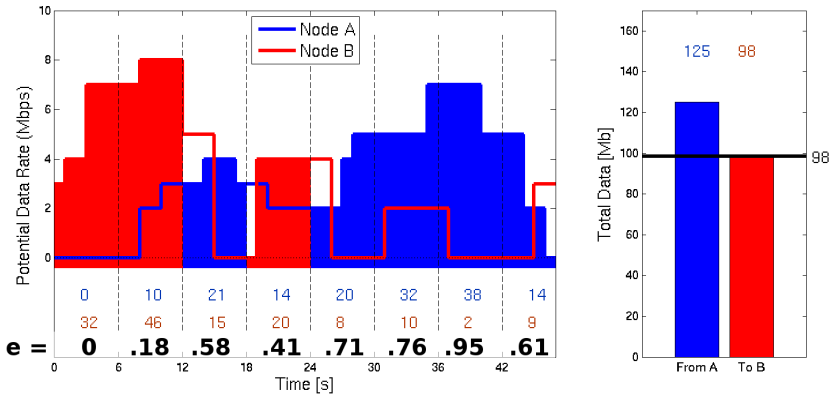
For segments k where $\sum_i x_{ik} = 0$, then those segments are allocated to delivering data to B. For s sources and one destination, the "s+1 node" link scheduling problem is then:



(a) The first steps of the Knapsack algorithm, beginning with the most efficient segment getting allocated to node A.



(b) The four most efficient segments allocated to node A, with the rest allocated to node B.



(c) The fifth most efficient segment marks the decision item, where node A has more allocated than node B.

Figure 4.5: Several steps of the Knapsack Policy on the example scenario.

Problem 7 (Multi-Source Link Scheduling Problem)**.**

$$[\boldsymbol{x_1}, ..., \boldsymbol{x_s}]^* = \underset{\boldsymbol{x_i}\ \forall i}{\arg\max} \sum_k \sum_i d_{ai,k}\ x_{ik} \tag{4.8a}$$

$$s.t. \quad \sum_k \sum_i d_{ai,k}\ x_{ik} \leq \sum_k d_{b,k} \left(1 - \sum_i x_{ik}\right) \tag{4.8b}$$

$$x_{ik} \in \{0, 1\} \quad \forall i, k \tag{4.8c}$$

$$\sum_i x_{ik} \leq 1 \quad \forall i, k \tag{4.8d}$$

The constraint Eq. (4.8b) can be manipulated to a knapsack form as follows:

$$\sum_k \sum_i d_{ai,k}\ x_{ik} \leq \sum_k d_{b,k} - \sum_k d_{b,k} \cdot \sum_i x_{ik} \tag{4.9a}$$

$$\sum_k \sum_i d_{ai,k}\ x_{ik} + \sum_k d_{b,k} \cdot \sum_i x_{ik} \leq \sum_k d_{b,k} \tag{4.9b}$$

$$\sum_k \sum_i d_{ai,k}\ x_{ik} + \sum_k \sum_i d_{b,k} x_{ik} \leq \sum_k d_{b,k} \tag{4.9c}$$

$$\sum_k \sum_i (d_{ai,k} + d_{b,k})\ x_{ik} \leq \sum_k d_{b,k} \tag{4.9d}$$

Now in the form of the knapsack weight constraint, we can define the efficiency of allocating segment k to node A_i as

$$e_{i,k} = \frac{d_{ai,k}}{\sum_j (d_{aj,k} + d_{b,k})} = \frac{d_{ai,k}}{s d_{b,k} + \sum_j d_{aj,k}} \tag{4.10}$$

The *multi-Knapsack* policy then uses this efficiency factor to define the order of allocating segments. An important insight from $e_{i,k}$ is that the denominator is constant for all source nodes. Therefore, for a given segment k, the source node with the highest efficiency will be the node with the highest value for $d_{ai,k}$ for that segment. Since the objective of the LSP is to maximize the data delivered (subject to at least as much being collected), it is intuitive that the ferry would not choose to listen to source A_i at segment k if it can collect more data from A_j at that segment.

This insight means the s+1 node problem can be formed into the single-source problem with an initial step that forms $d_{a,k} = d_{ai,k}$, and a final step converting allocations where $x_k = 1$ to x_{ik} where i maximizes $d_{ai,k}$). Let this solution method be the *max-Knapsack* policy. This consolidation step is an extra $\mathcal{O}(sn)$ operation, which is the same order needed to calculate Eq. (4.10) for all nodes. For both +1 node knapsack

algorithms, namely the multi-Knapsack and max-Knapsack policies, their run times are $\mathcal{O}(sn + n \log n)$, which is dominated by $n \log n$ for ferrying problems.

4.4 Most Data First Heuristic

4.4.1 MDF for 1-Source, 1-Destination

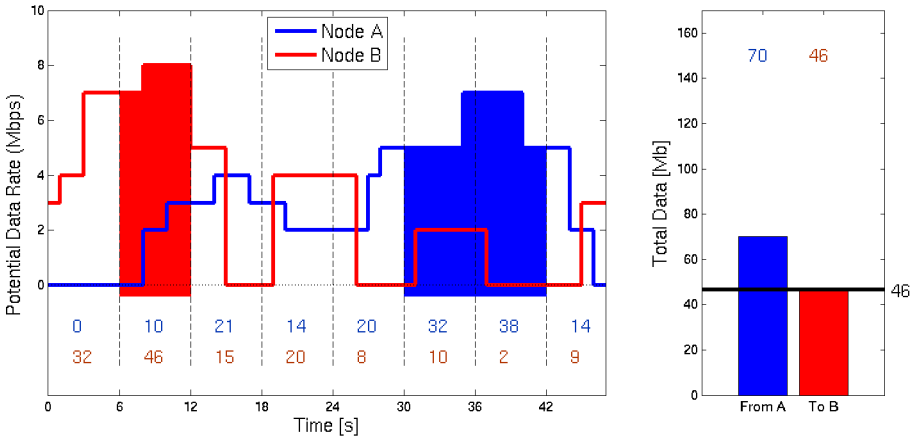
One approach to scheduling links while maintaining the balance of (4.1b) is to alternate allocating segments between A and B. This is the basis for the Most Data First (MDF) algorithm [12]. As segments get allocated, a running sum of data collected from A and delivered to B is maintained. The algorithm starts off allocating the segment with the most delivery potential to B. Then, while B's total is greater than A's, allocate available segments to A in order of best collection potential. Once A's allocation total is greater than B's total, available segments are allocated to B in order of delivery potential. This process repeats until all segments have been allocated.

Figures 4.6a and 4.6b show how the MDF algorithm allocates segments in the example scenario. It begins by allocating segment 2 to B, with a total data potential of 46 Mb. Then, with more allocated to delivery than collection, the Algorithm allocates segment 7 to A, with a data potential of 38 being the best for A. Since A's total is still less than B's total, the MDF algorithm then allocates segments to A, choosing segment 6 as the next best option. This tips the allocation in favor of A, seen in Fig. 4.6a, so the algorithm then allocates segments to B. This process continues until all segments have been allocated, resulting in the allocation in Fig. 4.6b with 107 Mb delivered to B.

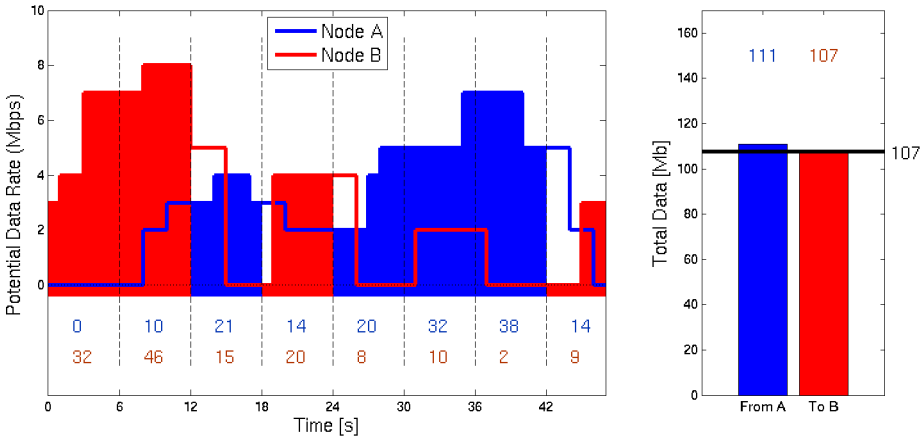
Also like the knapsack's greedy heuristic, the MDF can be shown to be arbitrarily bad in an extreme case (Appendix A.3); however, good ferry trajectories will be far from this extreme case. The MDF algorithm will achieve an optimal allocation in more cases than the switching point algorithm. Let the data potential for node A be sorted from high to low:

$$\overline{\boldsymbol{A}} = \{d_{a,1} \geq d_{a,2} \geq ... \geq d_{a,n}\} \tag{4.11}$$

where the index i of $d_{a,i}$ represents the segment with the i^{th} highest data potential. Suppose the MDF algorithm results in an allocation where segments 1 through k in the ordering from $\overline{\boldsymbol{A}}$ are allocated to A,



(a) MDF allocation after 3 steps.



(b) The final MDF allocation.

Figure 4.6: The MDF algorithm after 3 and 8 allocations.

and segments $k+1$ to n are allocated to B. This will happen if $d_b(i) \leq d_b(j)\ \forall i \in [1,k],\ j \in [k+1,n]$. This will also be an optimal allocation in the similar way that the switching point would be. The difference is that the ordering in the switching point allocation is based on time along the trajectory. The ordering in the MDF allocation allows for multiple switches between collecting from A and delivering to A. Thus it can achieve superior average throughput over the switching point policy in non-ideal communication environments.

The MDF algorithm also runs in linear time with the number of segments, once the segments are sorted. Note that the list of segments need to be sorted twice - one copy sorted for collecting data from A and another for delivering to B. Because of the need to sort the segments, it runs in $\mathcal{O}(n \log n)$ time.

4.4.2 Extensions to s-Source, 1-Destination

In the case of s source nodes and 1 destination, the MDF algorithm could also be reduced to the 1-source case in the same way the Greedy Knapsack algorithm can: reduce all the sources A_i into a single source A where each segment is represented by $d_{a,k} = \arg\max_i d_{ai,k}$. This *max-MDF* extension also runs in $\mathcal{O}(sn + n \log n)$.

However, the general Most Data First algorithm incorporates a natural fairness among nodes. Whereas the max-Knapsack and max-MDF systems could result in data only being collected from a subset of source nodes and completely ignore others, the general MDF system explicitly chooses to which node to allocate a next segment. In the single node case, it alternates between picking a segment for the source node and for the destination node based on which side of Eq. (4.1b) is lower. In the multi-source case, the *fair-MDF* makes the same choice between allocating to a source and allocating to a destination based on Eq. (4.8b); when the net potential data collected from sources is less than the potential data delivered to the destination, the fair-MDF picks the least-served source node for picking the next segment, i.e. the source node A_i for which of $\sum_j d_{ai,j}x_i$ over currently allocated segments j is least. In this way, the end result ensures a minimized ratio between least-served and most-served source nodes [3] The trade-off for this balanced allocation is that the overall amount of data delivered may be lower, which is examined in Section 4.6. Additionally, the sorting burden is increased, needing to sort data potentials for all $s+1$ nodes. The run time is then $\mathcal{O}\left((s+1)nlogn\right)$ (which is the general form for the run time of the MDF in the single-source case.

4.5 Simulation Results

This section shows the behaviors of the 5 link scheduling policies under simulated RF environments, which are chosen to highlight some of the extremes discussed previously. Though the RF environment is simulated, it reflects some of the patterns exhibited during flight experiments. A genetic algorithm is used to generate *good* trajectories through the environments, taking into account realistic aircraft dynamics with fixed altitude and heading control. The simulations were performed on a 64-bit Intel Xeon Quadcore CPU

[3] In the case of varying data priorities γ_{ai} between source nodes, the selection of a source node can be based on the weighted sum of potential collected data, $\sum_j \gamma_{ai} d_{ai,j} x_i$.

at 3.0 GHz each, with 8GB of RAM and running Gentoo Linux 3.8.13. The policies are then applied to these trajectories, with computation time and throughput performance compared to the relaxation LP solution of the link scheduling problem.

Note that for very sparse networks where there is no communication overlap, the ferry is only able to communicate to at most one node at a time. In this situation, deciding who to talk to is trivial – simply have the ferry talk with whomever it can, whenever it can. 4 of the 5 policies will result in this link schedule, while the 5th, Closest Node, will result in this case when there is a strong inverse relationship between distance and signal strength. These simulations consider the more interest case where node communication environments overlap.

4.5.1 Clean and Symmetric Environment

In an idealistic RF environment, communication decays with distance, with no multipath or interference effects. When both sensors have the same power, and the ferry's trajectory has symmetry – the trajectory around node A mirrors the trajectory around node B (Fig. 4.7) – then all policies result in roughly the same allocation. Furthermore, the conditions of (4.2) and (4.11) are met, and this allocation is optimal (Table 4.1). The linear program solver determines the solution [4] in 85 seconds. However, this is calculated over a single given trajectory on a powerful computer, rather than hundreds or thousands of candidate trajectories to evaluate on a small on-board computer. Relative to the scale of a 1-hour UAV flight typical for small unmanned aircraft, this solution would not work on line. The link scheduling policies achieve the same throughput in fractions of the computation time for this scenario. The time savings enables these policies to evaluate on the order of 1500 trajectories in the time it takes the BIP to solve one, enabling the potential to plan in the field.

[4] In this scenario, the Linear Program's relaxed solution matches the un-relaxed solution.

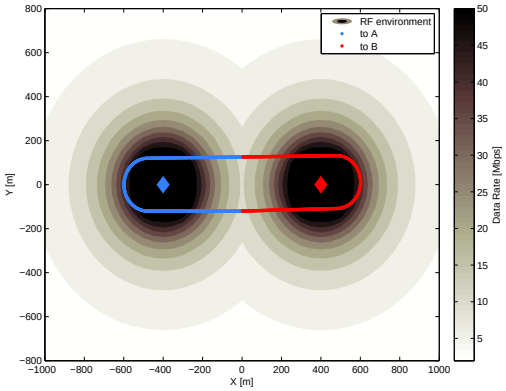


Figure 4.7: Ferry path and allocations using the knapsack heuristic, which is practically the same in this case for MDF, Switching Point and Best-RF.

Table 4.1: Results for a clean RF environment

Policy	**Effective Throughput**		**Computation Time**	
	Mbps	**%**	**s**	**%**
Linear Program	20.543	100	84.8	100
Closest	20.510	99.84	0.080	0.09
Best RF	20.510	99.84	0.080	0.09
Switching Point	20.529	99.94	0.543	0.64
Greedy Knapsack	20.529	99.94	0.138	0.16
Most Data First	20.529	99.94	9.783	11.5

4.5.2 Noisy Directional Environment

Wireless communication rarely behaves as cleanly as in the environment of Fig. 4.7. Instead, background RF noise, directional effects, and interferers result in much more complicated RF patterns. Figure 4.8 reflect this environment around nodes A and B. A good path does not fly straight between the nodes, but instead arcs to spend more time in regions of stronger communication (Fig. 4.9). As mentioned previously, oversimplifying vehicle dynamics for TSP-based solutions would not be able to account for such trajectories and will result in worse effective throughput.

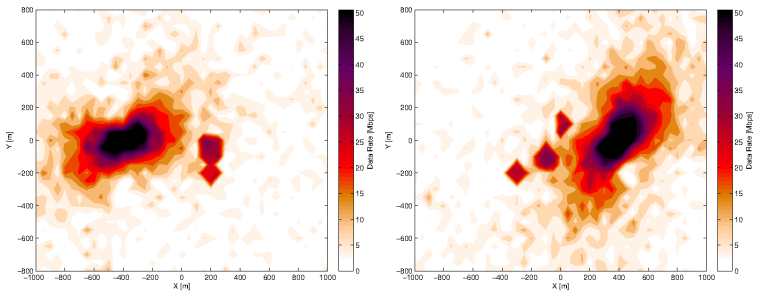


Figure 4.8: RF pattern for node A (left) and node B (right), with dipole antennas in a very noisy environment.

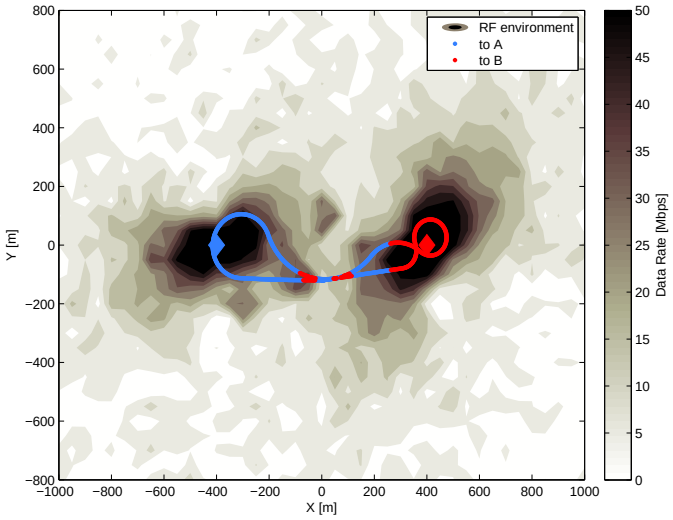


Figure 4.9: Ferry path and allocations using the knapsack heuristic, which highlights the need to for many allocation switches between nodes A and B.

Because of the high spatial variability of this environment, the ferry alternates its allocation much more frequently to take advantage of the various spikes in link quality for all policies except Switching Point, which inherently cannot alternate as frequently. Table 4.2 compares the performance of each policy. The Greedy Knapsack allocation achieves very near the optimal allocation in less than 1% of the computation time, with the Most-Data-First and Switching Point policies close behind. Though Best-RF can alternate frequently to take advantage of link quality spikes, it does not ensure the balanced communication of Eq. (4.1b). So although much more data can be collected from B, much less is delivered to A, resulting in a worse effective throughput.

Table 4.2: Results for the noisy RF environment

Policy	**Effective Throughput**		**Computation Time**	
	Mbps	**%**	**ms**	**%**
Linear Program	16.112	100	26.9	100
Closest	13.497	83.77	0.07	0.26
Best RF	13.981	86.77	0.08	0.31
Switching Point	15.556	96.55	0.38	1.36
Greedy Knapsack	16.104	99.95	0.11	0.42
Most Data First	16.054	99.64	7.69	28.6

4.5.3 Monte Carlo Simulations

The previous two environments highlight extreme examples of communication environments and trajectories. To more fully characterize the performance of the link scheduling policies, statistics were gathered over 150 different RF environments, where node communication power, antenna direction and additive noise levels were varied. Communication power in these simulations is bounded on the low end such that there still remained some overlap between the communication regions of the nodes. Any lower, or with nodes any further apart, results in a degenerate case where all policies yield the same allocation – specifically where each segment is allocated to the only node with non-zero communication. For each environment, 1000 trajectories were generated by the genetic algorithm, with aircraft flight speed varied between 20 and 50 meters per second (similar to the profile of the Tempest [20] aircraft platform). The naïve approach of the Closest heuristic, and the time-consuming Binary Integer Program solver are omitted from this study.

Tables 4.3 and 4.4 show the average and worst case performance of the policies relative to the Linear Programming relaxation solution. In general over the wide range of scenarios, the Greedy Knapsack policy consistently achieves near optimal throughput performance (over 98% of the Linear Programming solution in the worst case) with extremely quick calculation times (better than 2 milliseconds per trajectory). The Most-Data-First algorithm takes more time to evaluate, with slightly less throughput, averaging above 98 % of optimal, but 83% at the worst case. The Switching Point policy performs very well on average, but far from optimal in the worst case. It is also slightly slower than the Greedy Knapsack Policy. The commonly used Closest Node and Best RF policies are not surprisingly the fastest to compute. However they only achieve on average just 50% of the optimal throughput.

Table 4.3: Throughput Results for Monte Carlo Simulations

Policy	**Average %**	**Worst Case %**	**Std. Dev. %**
Linear Program	100	100	0
Closest	50.41	0.0	26.4
Best RF	48.39	0.0	32.5
Switching Point	98.51	73.35	2.06
Greedy Knapsack	99.86	98.31	0.13
Most Data First	98.20	83.36	1.56

Table 4.4: Timing Results for Monte Carlo Simulations

Policy	**Average**		**Worst Case**		**Standard Deviation**	
	ms	**%**	**ms**	**%**	**ms**	**%**
Linear Program	28.24	100	329.1	100	0	0
Closest	0.059	0.22	2.42	7.63	0.005	0.035
Best RF	0.046	0.17	0.62	2.19	0.004	0.025
Switching Point	0.088	0.32	0.74	2.76	0.018	0.069
Greedy Knapsack	0.059	0.22	1.58	0.83	0.006	0.033
Most Data First	0.609	2.24	12.40	39.2	0.094	0.347

4.6 Multiple-Source Simulation Results

In terms of algorithm scalability, Table 4.5 summarized the computation time for each heuristic. This section evaluates their performance with respect to data delivered as the problem size grows from one to 10 source nodes. This analysis excludes Switching Point, which has prohibitive exponential growth in computation time.

Additionally, the Monte Carlo simulations for multiple source nodes here cover more general environments. The previous simulations generated a broad variety of communication environments for each node and generate ferry trajectories through each environment. Instead, note that only the Closest heuristic is dependent on physical location of the nodes. We instead focus analysis on the remaining heuristics' performance with regards to Problem 7.

In these simulations, values for $d_{ai,k}$ and $d_{b,k}$ are randomly generated, where the number of segments k ranges from 100 to 400. 10,000 random data sets with a uniform distribution were generated for each scenario of source nodes i from 1 to 10. The Linear Program solution to each case is again the benchmark to which Best RF, multi-Knap, multi-MDF and max-MDF are compared. Data throughput performance is plotted in Fig. 4.10, showing average (solid lines) and worst-case (dashed lines) performance. Fig. 4.11 presents a close-up, showing that both Knapsack algorithms and both MDF algorithms are within 95% of the Linear Programming solution's performance. Multi-Knap and multi-MDF take approximately a 1% performance hit in the 10-source node case over the max-Knap and max-MDF.

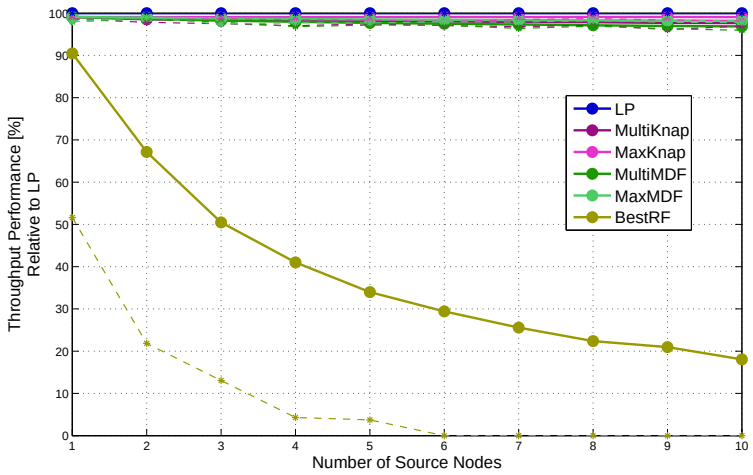


Figure 4.10: Average and worst-case throughput performance as the problem complexity grows with number of source nodes.

Figure 4.12 highlights the fairness metric of each algorithm. Fairness here is represented by a ratio of least served source nodes to most-served source nodes. 100% means all sources delivered the same amount of data; and 0% indicates that at least one node was completely ignored by the data ferry. For the max-Knap,

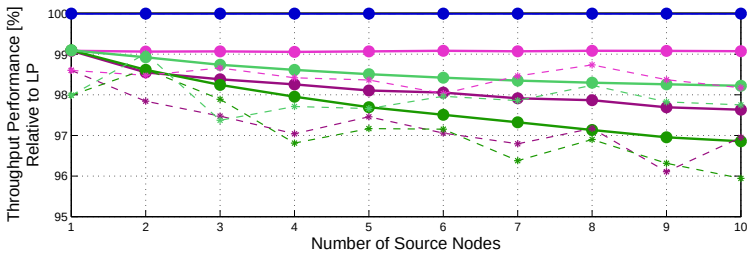


Figure 4.11: A close-up of multi-source throughput performance, showing both average (solid) and worst-case (dashed) performance.

max-MDF, LP, and Best-RF solutions, link schedules become much less balanced as the number of source nodes increases. The multi-Knap and multi-MDF algorithms, conversely, lose balance much more slowly, only degrading to a service ratio of 70% in the case of 10 source nodes.

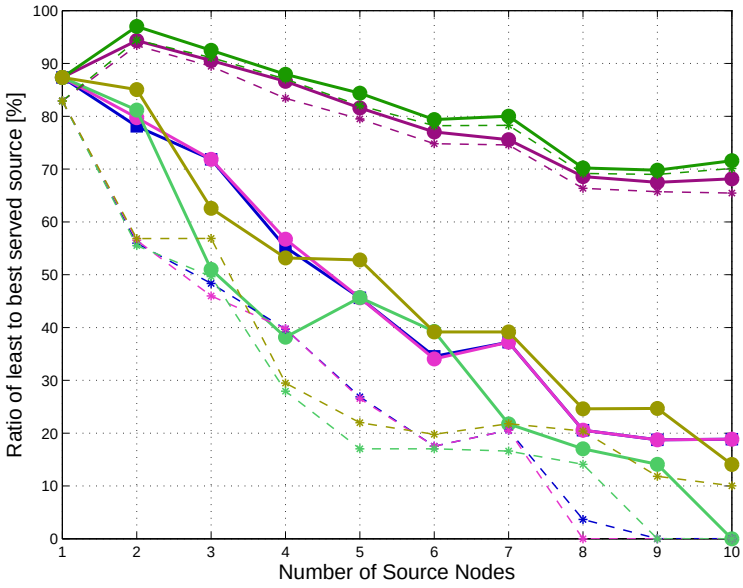


Figure 4.12: Service ratio between source nodes, representing fairness among link scheduling.

4.7 Summary

This chapter adapted the Link Scheduling Problem to the well-studied Knapsack packing problem. The Greedy Knapsack heuristic was developed for data ferry link scheduling, along with the Most Data First algorithm. These algorithms are compared to show several common heuristics, with theoretical run

times summarized in Table 4.5. The Greedy Knapsack and MDF algorithms show a significant improvement in overall performance in comparison to the other common approaches. In the single-source case, their computation time is similar to other fast methods, while their average performance is above 98% of the Linear Programming relaxation solution; worst case performance shows better robustness, with an effective throughput still above 80%. Further, these algorithms maintained their performance when extended to scenarios involving multiple source nodes.

Table 4.5: Theoretical run times of link policies based on n segments and s source nodes.

Policy	**Single-Source, Single-Destination**	**s-Source, Single-Destination**
BIP	More than Polynomial Time	More than Polynomial Time
Linear Program	$\mathcal{O}(n^3)$	$\mathcal{O}(sn + n^3)$
Closest	$\mathcal{O}(n)$	$\mathcal{O}(sn)$
Best RF	$\mathcal{O}(n)$	$\mathcal{O}(sn)$
Switching Point	$\mathcal{O}(n)$	$\mathcal{O}(s\ n^s)$
Greedy Knapsack	$\mathcal{O}(n \log n)$	$\mathcal{O}(sn + n \log n)$
Most Data First	$\mathcal{O}(n \log n)$	$\mathcal{O}\left((s+1)n \log n\right)$

Most significantly, using these policies within the cascaded ferry solution demonstrated their value to data ferry planning. With the motion control layer running these policies for each candidate trajectory, the fast and near-optimal performance of these policies resulted in magnitudes of improvement to data ferry control over the non-cascaded method. The next chapter will apply these techniques and validate their simulation results through field experiments using an unmanned aircraft as the data ferry.

Chapter 5

FIELD EXPERIMENTS

Evaluations thus far have been in simulated environments, where specific implementation details have been abstracted. Initially surveying communication environments between aircraft and ground nodes, true RF environment models will validate the priori Monte Carlo simulation results. Further, a ferrying system is developed that integrates link switch commands with a waypoint-based path follower. Implementing and fielding this ferrying system demonstrates several challenges to the assumptions mentioned in Ch. 2, namely stationary communication environment and perfect path tracking by the aircraft. Experimental results assess the performance of the link scheduling policies in the face of such challenges. Errors between planned and actual system performance are classified and evaluated, providing directions for future work.

5.1 Experimental System

Data ferry experiments involve two stationary Wi-Fi nodes and one unmanned aircraft to ferry between them. Ad-Hoc communication capabilities and a customized software suite combined autopilot, aircraft, planning algorithms and communication systems to both measure the RF environment and to carry out full ferrying experiments.

5.1.1 Aircraft System

The Skywalker X8 Unmanned Aircraft System shown in Figure 5.1 was utilized for RF data gathering and ferrying experiments. It is a small, light, rapidly deployable system with a delta wing configuration that spans 2.1m. It's high-density foam construction provides good durability and repairability. Combined

Figure 5.1: Skywalker WiFi-Ferrying UAS.

with the wide body to support a large payload volume, it is a robust airframe for research purposes. The airframe's specifications are summarized in Table 5.1

The Skywalker X8 carries multiple embedded systems for flight and mission control. The singleboard computer (SBC) pictured in Fig. 5.2 is a Ubiquiti Networks RouterStation [2], chosen for its common support of Linux and OpenWRT network control. The OpenWRT framework [1] is a well-supported open routing system with capabilities for sophisticated ad-hoc network routing protocols and network monitoring tools. The OpenWRT cross-compilation environment was further utilized to cross-compile the ferrying and flight-control software for the 32-bit MIPS architecture of the RouterStation. The WiFi connection is supported by an Atheros 802.11 mini PCI express card (Fig. 5.3a), connecting a hemispherical 2.4GHz WiFi Plus antenna [79] (Fig. 5.3b) to the RouterStation.

Figure 5.4a shows the Black Swift Technologies (BST) Swiftpilot, customizable and low-cost autopilot, weighing 34 grams. It handles low-level aircraft control integrating a 6axis inertial measurement unit, GPS, and dynamic and static pressure sensors. and provides a tablet-based interface to control altitude, turn rates,

Table 5.1: Skywalker X8 Airframe Specifications

Wingspan	2.1m
GTOW	3.5kg
Battery	5Ah 6S Lipo
Air speed	12-25 m/s
Endurance	30-45 minutes
Launch	Bungee

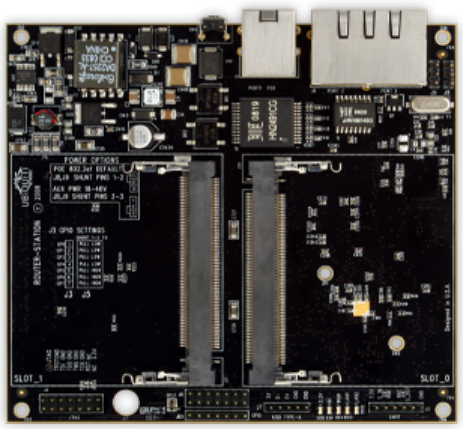


Figure 5.2: Ubiquiti RouterStation

and waypoint navigation. The tablet is connected to a groundstation (Fig. 5.4b) that provides long-range remote control to the aircraft via a 900 MHz link.

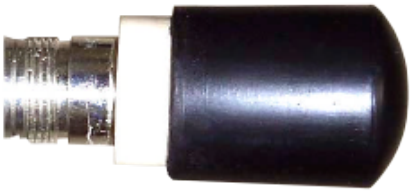
5.1.2 Ground Nodes

Two devices are primarily used as ground nodes within the data ferrying system. A Buffalo Networks AirStation router running a standard version of OpenWRT is used as the destination node. A mesh-networking radio (Fig. 5.5a) customized for ground-to-air communication is used as the source. It has a 32-bit AMD Geode LX processor from PC Engines, and like the Skywalker, it runs a customized version of OpenWRT Backfire, along with an Atheros network card and hemispherical antenna. For diversity in environments, some data gathering experiments included the use of an Odroid U3 Microcomputer with Alpha Networks dipole antenna and GPS receiver (Fig. 5.5b) as a source; and an Apple Macbook Pro with a patch antenna as a destination.

5.1.3 Software

To measure throughput capacity, the open-source networking tool Iperf is used [72]. Iperf measures maximum bandwidth of a channel from the Transport Layer (Layer 4 of the OSI model), meaning packet overhead to manage routing information is not captured in channel measurement. Here, it is used to measure Transport Layer capacity with the TCP protocol. All nodes are running a customized version of Iperf for

(a) Atheros 802.11 card



(b) 2.4GHz hemispherical antenna

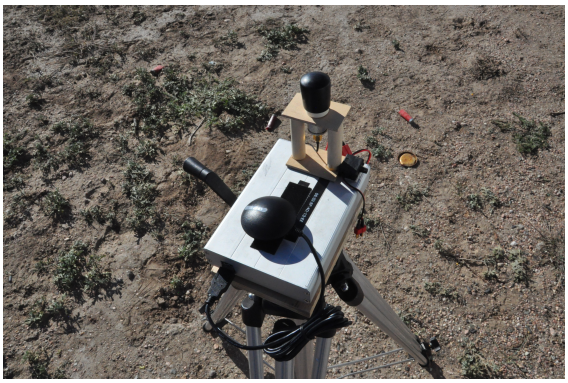
Figure 5.3: The network card and antenna for ferry communications.

(a) The BST SwiftPilot

(b) The BST GroundStation

Figure 5.4: The Black Swift Technologies Autopilot System.

the sake of data ferrying. The Iperf server mode is a daemon waiting and listening for incoming connections, while the client mode is actively seeking to connect with a server based on IP address. Whereas the client ends if a connection is either dropped or simply not established in a specific amount of time, the client program quits; because ferrying involves switching links, the MNR as the client will lose the connection with



(a) The Mesh Networking Radio (MNR)

(b) The Odroid microcomputer

Figure 5.5: Multiple types of WiFi transmitters were used as ground nodes.

the plane when it switches to communicate with the router as the destination / server. The customized version adds a wrapper to Iperf, creating new threads to attempt a re-establish a connection after drop-out. This allows the source and destination nodes to be set up and left alone for the entirety of the flight experiments.

A control interface between the RouterStation SBC and the Swiftpilot autopilot was developed for flight control. This interface recorded telemetry data from the autopilot in order to coordinate aircraft position with Iperf's link measurements. Additionally, the planner could send designed flight plans to the autopilot for the plane to fly for ferrying missions. The two-way interface for monitoring and controlling aircraft motion enabled the planner's link scheduler to coordinate link switches with ferry position. This system is detailed in Section 5.3.1.

5.1.4 Permissions

Flight experiments were performed under the Certificate of Authorization (CoA) 2013-WSA-34-COA, with a medically certified pilot and observer present for all flights. Authorized from the U.S. Federal Aviation Administration (FAA) this covers the Skywalker aircraft system at the Table Mountain test site, a designated radio quiet zone located 12 miles north of Boulder, Colorado. This area supports fundamental research of several governmental and public organizations, primarily for radio and electromagnetic experiments. The

rural area covers 4 square miles of mostly flat terrain; the clear visibility on this uncluttered mesa ensures autonomous operations are well-supervised by the ground crew. Operations include a defined orbit path close to the ground station, acting as a "lost-comm" plan: if the autopilot detects a loss of communication, it will return to this path to regain communications or for the manual remote control pilot to take control.

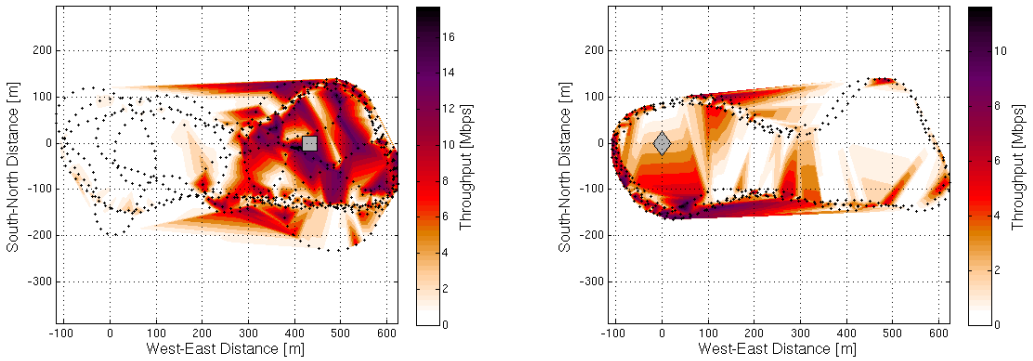
5.2 Field Throughput Data

Five flight experiments between November 2014 and March 2015 surveyed ferry-based RF environments, where throughput between the aircraft and stationary ground nodes was measured. In these tests, a ground node would operate Iperf in client mode (with the customized persistent-client wrapper), to connect with the aircraft running Iperf in server mode. Throughput rate measurements were taken at 1Hz as the plane flew along a standard surveying waypoint path. An example path from December 12 is shown in Figure 5.6, where the source node (MNR) is at the left, and the destination (laptop) is at the right.

Figure 5.6: Example "Zamboni" Waypoint Path for Throughput Environment Surveys, as viewed through the Black Swift Tech Swiftpilot.

Several cycles of this pattern were flown for the link from the MNR to the aircraft, ensuring sufficient and repeated coverage of the environment. The aircraft would then switch into the persistent client mode within Iperf, surveying the throughput from it to the destination node on the ground, where the destination node (Macbook Pro Laptop for Dec. 12) would listen with Iperf in server mode. The resulting measurements

are correlated with aircraft position through the environment; throughput is assumed linear and continuous between measurements, and interpolation based on Delaunay triangulation predicts values for such positions. Figure 5.7 shows the resulting throughput environments, where the diamond is the location of the source (MNR), and the square represents the location of destination (laptop); the black dotted line represents the aircraft's actual flight path where measurements were directly taken. These, along with other measured fields (included in Appendix A.4), exhibit nonlinear spatial variation ignored by typical communication assumptions.



(a) Throughput environment for source node to plane (b) Throughput environment for plane to destination node

Figure 5.7: Measured throughput environments between ground nodes and plane.

5.2.1 Validating Link Policy Performance

The first goal of these data gathering flights is to apply link scheduling policies to flight paths through true RF fields. For the five distinct RF environments, the genetic algorithm described in Section 3.4.1 is used to generate trajectories. For each scenario, the genetic algorithm runs for 2000 generations, each with a population of 40 motion command chromosomes. The evolution of chromosomes aims to determine trajectories that maximize the effective throughput of the system subject to a time constraint T, where the constraint is limited by a penalty function. The resulting objective is

$$J = \frac{b_b(t_n)}{t_n} - (T - t_n) \tag{5.1}$$

Table 5.2: Link Policy performance through the Dec. 12 Communication Environment

Policy	**Relative Effective Throughput**		**Computation Time**	
	Average %	**Worst Case %**	**Average [ms]**	**Worst Case [ms]**
Linear Program	100	100	21.8	38.8
Closest	52.20	0.00	0.04	0.59
Best RF	61.40	0.00	0.05	3.77
Switching Point	93.76	0.69	0.06	1.11
Greedy Knapsack	99.62	90.40	0.05	0.24
Most Data First	95.39	0.00	0.73	6.20

Of the total 80,000 trajectories, only trajectories that meet the time constraint (with period time $t_n < T$) are evaluated. In the following case study using the data from Figures 5.7a and 5.7b, the ferry is simulated to fly through the measured environment at a constant speed of 16 m/s, and periods are limited by $T = 90$ seconds.

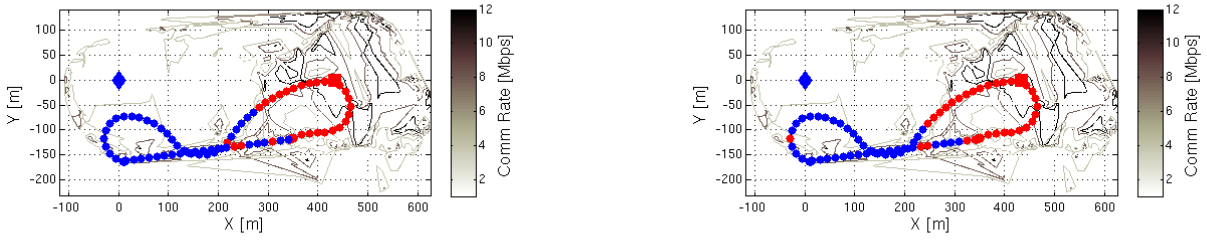
Table 5.2 shows the performance of each link scheduling policy relative to the Linear Programming solution, across the resulting 8404 admissible trajectories. As expected, the throughput trends from Monte Carlo simulations are maintained through these field environments: the Knapsack, Most Data First and Switching Point algorithms achieve a high average of effective throughput relative to the Linear Programming upper bound, and they arrive at solutions between 30 and 400 times faster.

The worst case performance for these policies occurs when the entire ferry path is out of communication range with one of the two nodes. In such a degenerate case, all policies (including LP) will result in a value of zero. Over the paths in these results, the worst case relative to the LP relaxed solution occurred when the ferry was only able to communicate with the source node along a small portion of the path, which corresponded to a great signal to the destination. In this case, the Most Data First began by allocating this region to the destination before starting to allocate to the source, which then had no useful segments left. This segment had the highest efficiency for the source node, though, and so the knapsack heuristic allocated it to the source first. In practice, this particular class of paths results in poor effective throughput regardless of policy; hence the need for the genetic algorithm to generate better trajectories.

5.2.2 Optimal Planning in Field Environments

The last example showed policy performance over a random set of admissible trajectories (with a portion of the population progressively getting better with each generation of the GA). The question still arises as to how the performance of overall ferry planning depends on policies within field environments. Over the same Dec. 12 environment, the full cascaded solution is implemented for each policy where the ferry is tasked with flying constant speed periodic trajectories to maximize average throughput, with $v = 16$ m/s and $T = 90$ seconds. The GA planner is run over 5000 generations for each policy, with a population of 40 chromosomes, where each chromosome is 10 pairs of turn rate and hold-time commands.

Over 5000 generations, the Cascaded-LP, Cascaded-Knapsack and Cascaded-MDF resulted in the same motion control, with differences only in link control. The path and allocations are shown in Fig. 5.8. The blue diamond is the MNR (source node) and the red square is the laptop (destination node). The blue dots along the ferry's path represent segments allocated to the ferry receiving data from the MNR, and red dots represent the ferry sending data to the laptop. The difference between their allocations are only in the final 4 segment allocations, located primarily around the center of the environment, as well as at one particularly strong spot for the laptop at the left end of the trajectory. (This left spot also has good signal strength to the MNR, and so the Knapsack efficiency is not as high as other segments, resulting in the different allocations between Knapsack and MDF.)

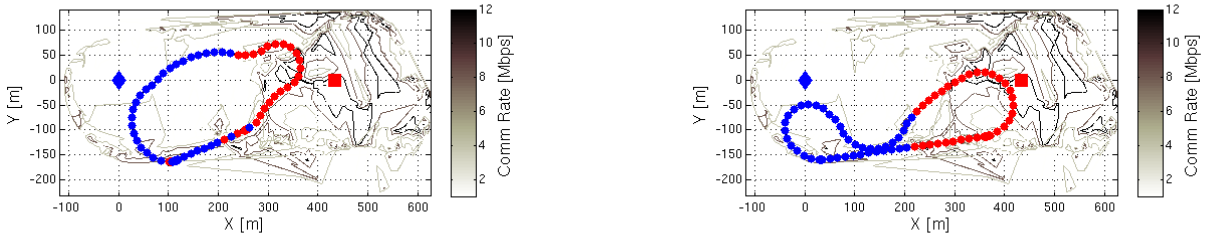


(a) Cascaded-Knapsack solution

(b) Cascaded-MDF solution

Figure 5.8: Paths and link schedules planned after 5000 generations for the Dec. 12 environments.

In contrast, the BestRF solution was unable to improve as quickly through generations as the others. Its path (Fig. 5.9a) after 5000 generations remained in a higher central region, rather than remaining low and extending further to the right as the other paths. The top section of this environment does not show as much of a correlation in throughput rates as the bottom half; consequently, the Closest-Node heuristic was able to determine a trajectory (Fig. 5.9b) closer to that of the Knapsack and MDF solutions, outperforming BestRF. Table 5.3 summarizes the results of planning within flown RF field, demonstrating that the plans obtained with better heuristics have better performance over a fixed number of generations.



(a) Cascaded-Best-RF solution

(b) Cascaded-Closest-Node solution

Figure 5.9: Paths and link schedules planned after 5000 generations for the Dec. 12 environments.

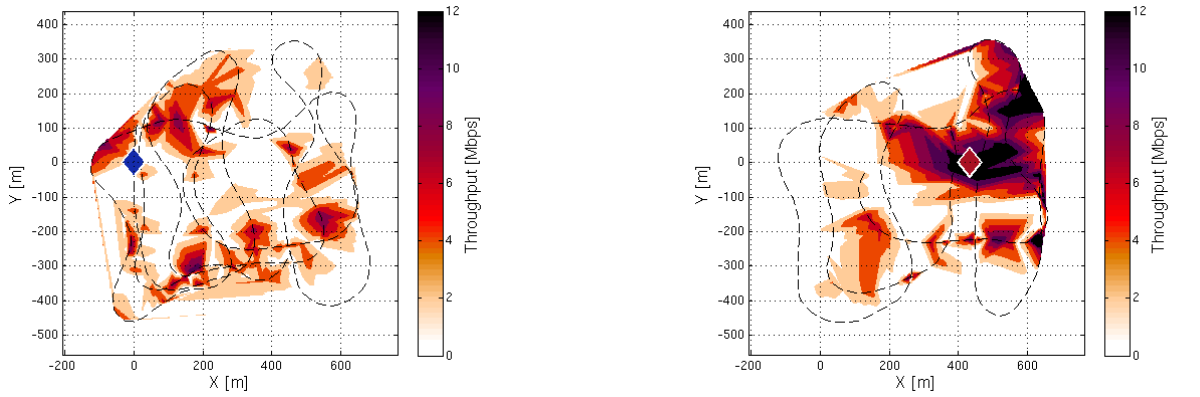
Table 5.3: Performance of Planned Routes versus Link Policy

Policy	**Effective Throughtput [Mbps]**	**Performance relative to LP [%]**
Linear Program	3.72	100
Closest	3.30	88.8
Best RF	3.12	83.8
Switching Point	3.17	85.3
Greedy Knapsack	3.71	99.7
Most Data First	3.68	99.1

5.3 Ferrying in the Field

The previous section evaluated simulated ferry paths through field environments. This section goes beyond simulation and implements a ferry on the unmanned aircraft in the field. First, the process to

integrate ferry algorithms on the aircraft is discussed, followed by experimental results. The discussion focuses on flight experiments involving the MNR as a source, and the router as the destination, also located at Table Mountain. An initial flight surveyed the throughput fields to provide the basis for planning subsequent ferrying flights. The survey showed similarly non-smooth characteristics in throughput versus position in the environments (Fig. 5.10).



(a) Throughput environment from MNR to aircraft

(b) Throughput environment from aircraft to router

Figure 5.10: March 20 throughput environments between aircraft and ground nodes.

5.3.1 Adapting Ferry plans to the Unmanned Aircraft System

The cascaded ferry planner generates trajectories at the higher level by determining a sequence $\boldsymbol{u}\{\omega_k, \tau_k\}$ to control the dynamics in Eqs. 2.3-2.4. These commands determine the flight trajectory $\Gamma = [\boldsymbol{p}, \boldsymbol{t}]$. The surveyed throughput environments (e.g. Fig. 5.10) then determine data potentials $\boldsymbol{d_a}, \boldsymbol{d_b}$ along the resulting trajectory. These data potentials are fed to a link policy to then determine the link schedule control variables for that trajectory. This trajectory and planned link schedules are based on the assumptions that the aircraft will follow the trajectory (being at a specified point at the right time), and that the throughput fields while later ferrying matches the surveyed fields used to generate the link policy. That is, for flown

states $\hat{\boldsymbol{X}}$, the following are assumed true for all segments k:

$$\hat{p}_k = p_k, \tag{5.2a}$$

$$\hat{t}_k = t_k, \tag{5.2b}$$

$$\hat{R}_a(\hat{t_k}) = R_a(t_k), \tag{5.2c}$$

$$\hat{R}_b(\hat{t_k}) = R_b(t_k). \tag{5.2d}$$

Assumptions (5.2a) and (5.2b) are challenged in flight by disturbances such as wind gusts or potential timing error in a turn-rate hold command. To address such disturbances, the SwiftPilot waypoint-following navigation is used. This waypoint-based navigation ensures that the aircraft will closely follow the general path in a manner robust to disturbances. Further, the times at which the autopilot switches waypoints are used to coordinate aircraft position with switches along the link schedule. Though this method does not completely mitigate the effects of flight disturbances, it provides a robust way for the UAV to continue following the ferrying plan.

Addressing discrepancies in assumptions (5.2c) and (5.2d) requires a feedback control system that dynamically adjusts the ferry's plan. This is discussed in more detail in Ch. 6, which develops an iterative learning-and-replanning method to improve the ferry's performance in the face of dynamic RF environments.

The planned trajectory Γ and link schedule $\boldsymbol{c}$ is then mapped to path-defining waypoints:

$$\boldsymbol{WP} = f(\Gamma, \boldsymbol{c}) \tag{5.3}$$

In practice, the planner determines trajectories at 1-second intervals, matching the sampling rate of the throughput environments. A limit in the autopilot's waypoint-following system results in some issues with such a dense plan; specifically, the autopilot chooses the next guiding waypoint based on the closest waypoint on the path at least some minimum distance ahead of the plane. Instead of allocating a guiding waypoint at every segment, Eq. (5.3) aggregates the 1-second segments into larger waypoint segments, where endpoints of the waypoint segments are based on either a minimum separation distance from the previous waypoint, or a switch in link control.

A one-to-one mapping $k = k(m)$ corresponds each waypoint WP_m to a position and link command $\{p_k, c_a(t_k), c_b(t_k)\}$. Define the first waypoint with $m = 0$ and corresponding $k = 0$, giving $WP_0 = \{p_0, c_a(t_0), c_b(t_0)\}$. Then for $j = k(m-1)$, the next waypoint WP_m corresponds to the minimum $k > j$ where at least one of the following conditions is met:

$$\begin{cases} ||p_k - p_j|| > D, \\ c_a(t_k) \neq c_a(t_j), \text{ or} \\ c_b(t_k) \neq c_b(t_j). \end{cases} \tag{5.4}$$

In this way, the waypoint sequence can be used as a proxy for link scheduling commands, as the link schedule commands are constant for each aggregated waypoint segment. The autopilot shares telemetry information with the flight computer, particularly the current waypoint to which the aircraft is tracking. Associated with a link schedule command, the flight computer can then control the ferry's communication links appropriately.

5.3.2 Policy Performance on a Standard Path

The Figure 5.11 shows the BlackSwift Tech tablet interface to the autopilot control. A basic path through the environment is used as a baseline path to compare policy performance in flight, with a period of 80 seconds at 16 m/s. The waypoints defining this path are based on the Knapsack-based link schedule; the overall path remains the same, but location of waypoints vary based on the policy used to define the link schedule. Additionally pictured is the lost-comm waypoint, a circling orbit around Waypoint 10.

Multiple laps of this path are flown for each policy and waypoint plan. The resulting path and allocations for the Most Data First policy is shown in Fig. 5.12. The thick solid line represents the planned flight path, and the circles represent the segments along the path actually flown. For the ferry's link schedule, blue indicates listening to the MNR and red indicates sending data to the destination router.

The significant issue present in this flight is the error in path tracking. Additionally, there is error present in the time along the path. Figure 5.13 show the position states over time, along with the aircraft flight speed across all experiments. The horizontal distance (X-position) tracks fairly well, with the main error resulting from a difference in speed. Though the aircraft maintains indicated air-speed at 16 m/s well,

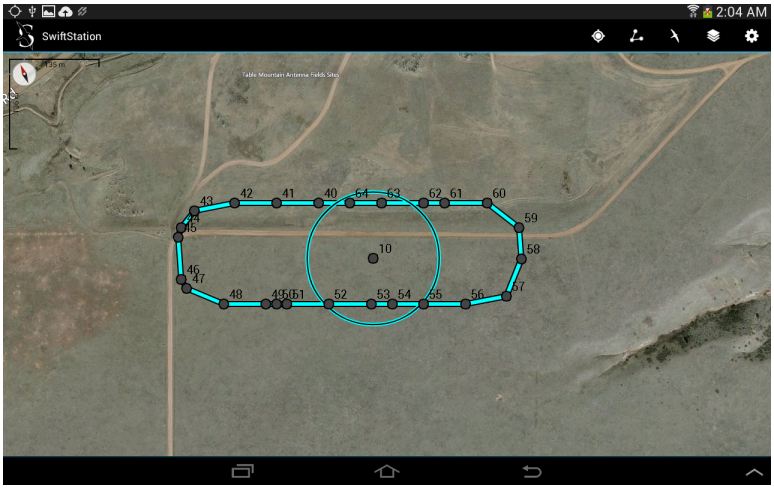


Figure 5.11: Standard ferry "race-track" plan as viewed through the BST tablet interface, with waypoints defined for Knapsack-policy switching.

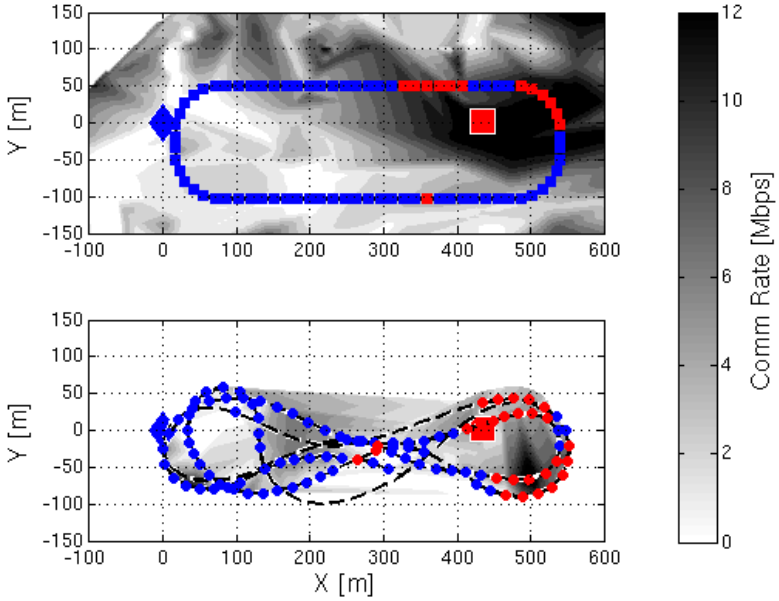
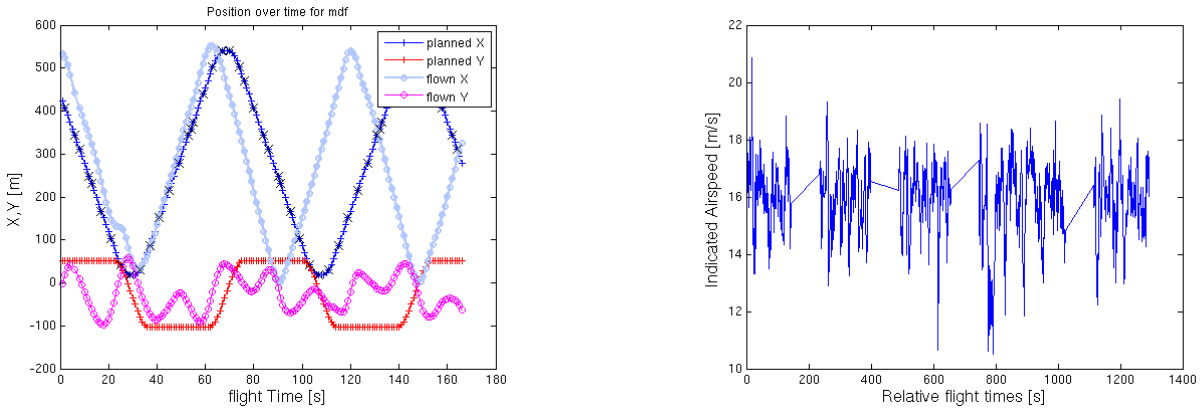


Figure 5.12: Flown path and allocations for the Most Data First policy.

the ground speed was above 18 m/s. This resulted in period times of 60 seconds instead of 80. The vertical (Y) position has difficulty tracking around the tight turns on the ends of the path, with a cyclic response after overshooting the turns. The RMS error in time-based position is between 80 and 90 meters over a single period, which is similar among the flights for each remaining link scheduling policy.



(a) Position states versus time for the MDF flight

(b) Ferry speed over time across all policy flights

Figure 5.13: Aircraft states versus time, position and speed.

Figure 5.12 illustrates the ferry's link switching throughout the environment. Though the UAV's speed changed the timing of the ferry along the path, using the autopilot's waypoint switching as the coordinator for link switching provided a degree of robustness to the speed issue. Note that at the northeast portion of the planned path (above the router, displayed as a red square), the link schedule for Most Data First includes a brief section of collecting data from the MNR (blue diamond at the left) in between delivering data to the router. This link switch is consistently replicated at the west side of the flown path despite the increased speed of the UAV (and hence shorter period of the route). This is able to occur because of using the position-based waypoints that the autopilot is following, and that it is these guiding waypoints that governs link switching. However, there is still a small timing issue with this coordination. The autopilot uses waypoints as guides, and so it aims toward a waypoint that is close, but beyond some minimal distance in front of it. This means that shortly before the plane reaches Waypoint WP_m, it will switch guiding waypoints to WP_{m+1}. This early switching leads to the small timing error in link switching.

With the sensitivity of communication rates on position, Fig. 5.14 shows how the communication rates vary with time. The solid lines represent the planned throughput rates, while the dashed lines with circles represent the actual throughput rates at each segment in flight; blue represents the source node (MNR), and red represents the destination (router). With the challenge in path tracking and timing, it is not surprising that the actual rates for both nodes vary significantly from the planned communication.

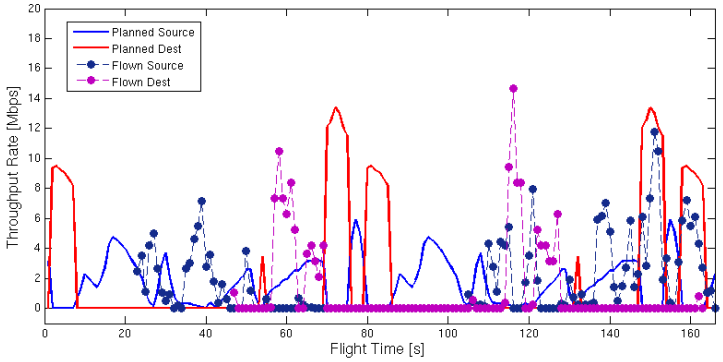


Figure 5.14: Planned communication link schedule and rates versus actually flown for the Most Data First policy.

Since the aircraft struggled to follow the path, it is expected that actual communication rates would vary from what was planned. Using the communication environment from the original planner, the solid lines in Fig. 5.15 represent what rates would have been expected along the path actually flown, based on the environments surveyed prior to ferrying missions (Fig. 5.10). Though much closer to what was actually seen, there are some significant remaining variations, most notably weaker communication to the router at 65 to 70 seconds and 120 to 125 seconds.

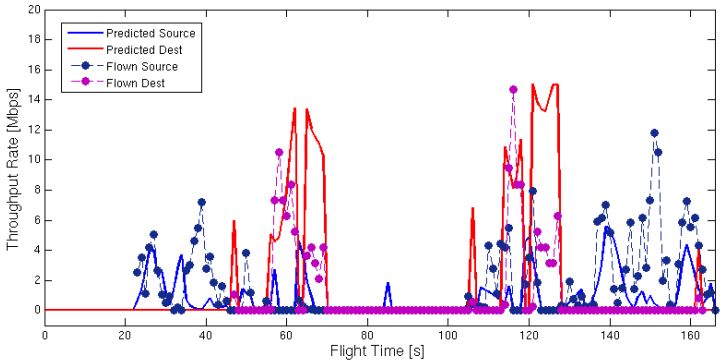


Figure 5.15: Communication link schedule and rates from the Most Data First flight versus what the planner's model would have predicted along the flown path.

The path tracking errors are common and similar across the flights for all policies, as are the variations in communication. These errors impact the effective throughput achieved by the ferry. Fig. 5.16 compares what the planned ferry performance was expected, and what was actually achieved. In most cases, ferry

performance suffered significantly; Knapsack achieved 54% of the planned performance, Most Data First achieved 71%, and Switching Point had 19%. BestRF actually performed 1% better than planned; however, as a generally poor policy, it only achieved 0.75 Mbps, while Knapsack and Most Data First achieved 0.92 and 1.18 Mbps respectively.

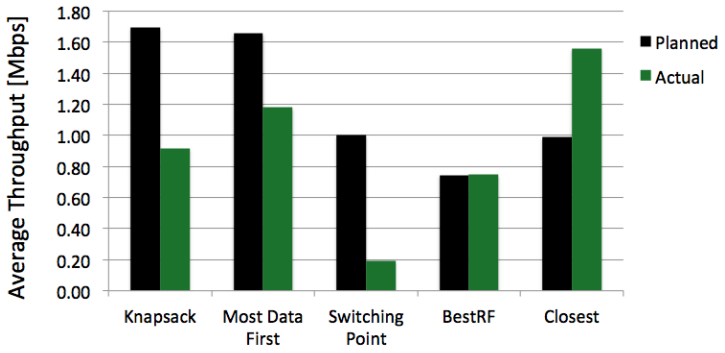


Figure 5.16: Average ferry throughput for each policy, showing planned performance versus performance actually achieved in flight..

Surprisingly, the Closest Node policy actually performed the best, with 1.56 Mbps instead of the 0.99 Mbps planned. The main reason for this is that the RF environment is not static in time, but instead changes. Figure 5.15 showed how the router's RF environment changed for the worse during the MDF flight. Peculiarly, the environment for both the MNR and the Router were much stronger later in the day for the Closest Node flight. Figure 5.17 shows an increase in actual throughput rates for both nodes across the entire path, leading to large improvements in achievable ferrying rates. The variation of RF environments with time is an important factor to which each link scheduling policy is susceptible. Addressing this challenge is the goal of the following chapter.

5.4 Summary

The developed throughput sampling system enabled persistent, direct measurements of Transport-Layer throughput capacity between ground nodes and aircraft. The resulting fields gathered through flight-based surveys validated the simulation results in Ch. 4 by conducting similar evaluations on policy performance through true fields. Most significantly, this chapter developed a system to integrate path and link

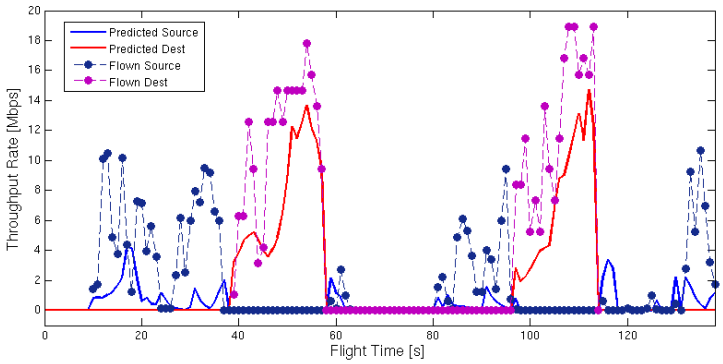


Figure 5.17: Communication link schedule and rates from the Closest Node flight versus what the planner's model would have predicted along the flown path.

control with the Swiftpilot's control; this attempted to ensure that the planner could assume the aircraft would remain close to the path to execute the ferry plan well.

Accurate execution of the ferry plan was challenged by several factors. Though still generally following the planned path, tracking error was beyond the range of the radio frequency environment's variations. The process of coordinating link schedule with the autopilot's waypoint switching ensured continued ferrying despite the poor path tracking; however, the autopilot's guiding algorithm would switch waypoints earlier than desired. Further tuning could improve both tracking and the choice of waypoints to improve link switch timing. Beyond path tracking, the time-varying communication environments resulted in significant differences between planned and actual ferry performance. Despite such a difference, the higher potential performance from plans based on the Greedy Knapsack and Most Data First policies resulted in decent ferry performance in comparison to the other plans. More experiments are needed to assess this trend further, as well as to isolate and eliminate the three categories of errors present in field experiments.

The assumption of trajectory following, requiring accurate timing and positioning of the aircraft to ferry data according to plan, proved to be a challenging aspect to fielding a ferrying system. Integrating the planner and link scheduler with the autopilot by mapping commands to a waypoint plan demonstrated functionality. However, improvements to the autopilot's tracking system, as well as communication between autopilot and mission computer, are necessary to reduce the errors seen in flight experiments. Despite errors

in path-following, link switch timing, and time-varying RF environments, the experiments demonstrated the benefits of smarter link-scheduling methods.

Chapter 6

LEARNING THE RF ENVIRONMENT

An essential requirement to optimize a ferry system is that the RF environments between each node and the ferry are known. This chapter integrates methods of learning the RF environment with ferrying, specifically using a Gaussian Process to model the spatial variation of RF signals. Learning here is done *opportunistically*: as the unmanned aircraft ferries data between the nodes along a planned trajectory, knowledge of the RF environment learned during the trajectory is used to further refine the models. This allows the ferry to replan trajectories based on the most up-to-date information of the environment, enabling it to adapt to any previous errors or changes within the environment.

6.1 Background

6.1.1 Radio Frequency Environment Modeling

RF environments can be modeled theoretically based on signal propagation and decay [61]. Probabilistic shadowing and fast fading [50], as well as terrain features [46] add more detail. Though these models can provide a reasonable starting point for planning, additional variations in the RF environment where the sensors and ferry are deployed will yield modeling errors [53], leading to significant differences between planned and actual data ferry performance [9]). Thus for accurate ferry planning, some form of on-line learning and model refinement is necessary.

Reinforcement learning has been examined for data ferrying [59], avoiding RF models altogether. Here, the ferry can fly many paths with varied communication schedules to determine the optimal solution, in a process similar to stochastic optimization. A designed policy defines how the exploration of an environment is

balanced with exploration, using only the flight trajectory's reward as a guide, eliminating the need to model the environment. However, this suffers from the amount of candidate trajectories that the ferry must fly, requiring exponentially more time and energy to reach the optimal solution [32]. Further, RF environments typically exhibit spatial correlations that can be learned and modeled [61, 76, 51, 31]. Instead of the slow model-free approach of Reinforcement Learning, a model-based approach has the advantage of learning and refining an RF model, enabling previous measurements to inform and predict values in regions not measured. This, in turn, allows the ferry to achieve better ferry paths with far fewer passes.

6.1.2 Modeling Approaches

RF models are commonly represented by sets of parameters to represent the environment; the goal is then to learn the parameters that best fit the data that has been measured. This idea has been tested with an unscented Kalman filter [68] to fit a physics-based model, where signal strength is assumed to decay with distance from the node. Including additional parameters to capture additional propagation effects, such as fading [31] and directionality [11] can improve the ultimate resolution of a learned model. However, the parametric nature of physics-based models limits the accuracy in modeling stochastic environments.

Gaussian processes (GP) [62] are a non-parametric alternative for measurement prediction in stochastic environments. With only a relatively small set of hyperparameters [76, 22], GPs capture the inherent characteristics of the environment by learning the correlations between the training samples. While the predictions of the GP improve as more training data from the environment is available, its data requirements are modest compared to the extensive in-flight data collection required for model-free reinforcement learning [59]. This chapter utilizes a Gaussian process (GP) to capture RF variations, i.e. deviations from an *a priori* physics-based model.

6.1.3 Objectives

Most motion control work for learning these RF variations has focused on environment learning as the sole objective, eg. random motion [76], signal extremum seeking [22, 68], or full environment characterization [22, 38]. Here our objective is data ferrying. Rather than taking time away from ferrying to explore the

environment, this chapter learns the RF environment only *opportunistically* while ferrying. The unmanned aircraft ferries data between the nodes, along a path planned using some initial RF model. While ferrying, the aircraft is able to sense the received signal strength from each node, and feed these measurements into the GP. The GP uses these measurements to refine the RF model, enabling the ferry to replan a smarter ferrying route.

This chapter details the integration of the ferry planner with the GP's RF estimation (Section 6.3). Important considerations are described to smooth this iterative ferry-and-learn process, and the general behavior of the integrated system is discussed. The system is analyzed through simulation in Section 6.4, showing significant ferry performance improvement with opportunistic GP learning. The ferry-and-learn process is then implemented with alternative estimation approaches in Section 6.5. The Gaussian process is evaluated against learning systems using model-based parameters through least-squares fitting, similar to [68] and [11]. This shows how the geospatial modeling and non-parametric adaptability of the GP allows it to capture environmental artifacts, and thus perform well regardless of the characteristics of the RF environment.

6.2 Gaussian Processes for RF Modeling

[76]

6.2.1 RF Propagation Model

Several factors, including environmental effects like fast fading, multipath transmission, slow fading, and interference, influence the signal-to-noise ratio (SNR) and throughput measurements at a location. Traditionally, SNR is derived from empirical RF propagation models with additive white Gaussian noise (AWGN) ν

$$z_{j,i} = \Xi_j(p_i) + \nu \tag{6.1}$$

where $z_{j,i}$ is the measured signal and $\Xi_j(p_i)$ is the *a priori* predicted model for node j at location p_i. However, there is a component of noise ν which is dependent on the measurement's location [51]. This dependence

allows the signal to be written as

$$z_{j,i} = \Xi_j(p_i) + v_i \tag{6.2}$$

where $v_i \sim \mathcal{N}(\mu(p_i), \sigma^2(p_i))$ is the location-dependent noise, and represents a distribution over functions. This distribution is learned from spatial correlations that exist in local regions of the operating field. These spatial correlations can be harnessed to extend the initial prediction model by learning the model variations.

Variation $e_{j,i}$ in node j's model at a measurement location p_i is the difference between the signal measurement $z_{j,i}$ and the predicted model estimate Ξ_j. Thus, the measured RF variation is the residual or the error of the empirical model

$$e_{j,i} = z_{j,i} - \Xi_j(p_i) = v_i \tag{6.3}$$

The construction of the RF variation model requires location-tagged datasets of signal measurements. N measurements are collected at known locations, and their RF variations are calculated using (6.3). These N samples of RF variation serve as the training set $\{\mathbf{X}, \mathbf{Y}\} = \{(\boldsymbol{\rho}_i, e_i)\}_{i=1:N}$, where $\boldsymbol{\rho}$ represents the state of the receiver at the time of the measurement. For the spatial model used here, $\boldsymbol{\rho}$ represents the measurement location.

6.2.2 Guassian Process Method

A Gaussian process is used to learn the RF variations. This is a nonparametric modeling tool that learns the spatial correlations from a training set of measurements. Once trained, the GP can use the learned information to predict the probability density function of the value of e' at any previously unseen state $\boldsymbol{\rho}'$, by computing its correlation with the training set samples. The prediction is made in the form of a Gaussian probability density function (PDF) with mean and variance

$$\begin{aligned} \mu_e(\boldsymbol{\rho}') &= k(\boldsymbol{\rho}, \boldsymbol{\rho}')^T (K + \sigma_n^2 I)^{-1} \mathbf{e} \\ \sigma_e^2(\boldsymbol{\rho}') &= k(\boldsymbol{\rho}', \boldsymbol{\rho}') - k(\boldsymbol{\rho}, \boldsymbol{\rho}')^T (K + \sigma_n^2 I)^{-1} k(\boldsymbol{\rho}, \boldsymbol{\rho}'). \end{aligned} \tag{6.4}$$

Here $k(\boldsymbol{\rho}, \boldsymbol{\rho}')$ is the $N \times 1$ vector of correlations of the test point with all the training points, $\mathbf{e}$ is the $N \times 1$ vector of measured variations at the training points, K is an $N \times N$ kernel matrix with entries $k_{ij} = k(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j)$ $(i, j = 1, \ldots, N)$, and σ_n^2 is the variance of the Gaussian noise in the original process.

One of the main advantages of using a GP is that it provides the prediction in the form of a PDF, matching the probabilistic nature of RF environments. In addition, GPs are non-parametric, enabling learning and adaptation in a wide range of environments with diverse physical properties.

A key design decision when using a GP is the choice of the correlation function $k(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j)$. Prior work [76, 51] has established that the following squared exponential kernel function is appropriate for modeling RF variations:

$$k(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j) = \sigma_f^2 \exp\left(-\frac{1}{2}(\boldsymbol{\rho}_i - \boldsymbol{\rho}_j)^T L^{-1}(\boldsymbol{\rho}_i - \boldsymbol{\rho}_j)\right) + \sigma_n^2 \delta_{ij} \tag{6.5}$$

where σ_f is the signal variance, σ_n^2 is the noise variance, and $L = l_s I_2$ is a diagonal matrix of lengthscales which determines the distance over which samples are correlated. Together, these comprise the hyperparameters $\theta = [\sigma_n, \sigma_f, l_s]$ of the GP, and capture inherent properties of the environment being modeled. The hyperparameters are learned by maximizing the log marginal likelihood using conjugate gradient methods [62].

$$\begin{aligned} \theta^* &= \arg\max_{\theta} \ln p(\mathbf{e}|\theta) \\ &= \arg\max_{\theta} \left[-\frac{1}{2}\mathbf{e}^T C_N^{-1}\mathbf{e} - \frac{1}{2}\ln|C_N| - \frac{N}{2}\ln 2\pi\right] \end{aligned} \tag{6.6}$$

where $C_N(\boldsymbol{\rho}; \theta) = K(\boldsymbol{\rho}; \sigma_f, L) + \sigma_n^2 I$. Further details of GPs, and their use for modeling RF variations can be found in [62, 7] and [76], respectively.

6.3 RF CHARACTERIZATION FOR DATA FERRYING

This section describes how the GP-based approach is combined with the cascaded ferrying process presented in the previous chapters. Learning is done in an *opportunistic* manner, using data collected while ferrying to improve the RF model. The general combined process flow is first described, followed by several key integration details to help ensure smooth operation.

6.3.1 Integrated System Overview

In the first iteration, the ferry plans its trajectory based on *a priori* RF models that cover the full environment for both nodes, setting $\mathcal{M}_{A,0} = \Xi_A$ and $\mathcal{M}_{B,0} = \Xi_B$. These models can be generated using

basic radio propagation theory or modeling tools such as SPLAT! [46]. The initial models do not need to be very accurate, though a bound on *a priori* error is still to be determined. The simulation studies in Sections 6.5 and 6.4 will show how over-estimating the RF environment generally leads to better performance.

The genetic algorithm described in previous chapters optimizes the ferry's first path with these initial models. The unmanned aircraft then flies along this path in the real environment, collecting data from A and delivering it to B. During this process, it simultaneously measures and logs the strength of the signal $z_{i,j}$ it is receiving from each of the two nodes. Each of these datasets can be compared to the *a priori* models' estimates for signal strengths at each location along the path. These differences, given by

$$e_{A,i} = z_{A,i} - \Xi_{A,i} \quad i = 1, \ldots, N_A \tag{6.7}$$

are node A's RF variations, as in (6.3). Node B's variations, $e_{B,i}$ for $i = 1, \ldots, N_B$, can be similarly computed from the flight measurements.

For each of the nodes, a GP is trained on these location-tagged datasets of RF variations. These variations are spatial in nature, and are used by the GP to learn hyperparameters $\theta_{\cdot,n}$ for each radio for every interval, which include the lengthscale or the distance of correlation l_s, signal variance σ_f^2, and noise variances σ_n^2. These hyperparameters and the training set can then predict, through (6.4), the RF variation for each node at any location in the entire environment

$$\begin{aligned} (\mu_A(p'), \sigma_A^2(p')) &= \mathcal{GP}(p' | \{\mathbf{p}_{1:n}, \mathbf{e}_{A,1:n}, \theta_{A,n}\}) \\ (\mu_B(p'), \sigma_B^2(p')) &= \mathcal{GP}(p' | \{\mathbf{p}_{1:n}, \mathbf{e}_{B,1:n}, \theta_{B,n}\}). \end{aligned} \tag{6.8}$$

Thus, once the training is complete, the learned model can be used as a black box to predict the RF variation at any given location p'. In the rest of the chapter, the short forms $\mathcal{GP}_{A,n}$ and $\mathcal{GP}_{B,n}$ represent these learned stochastic models, which predict the signal correction values for some or all of the locations in an environment.

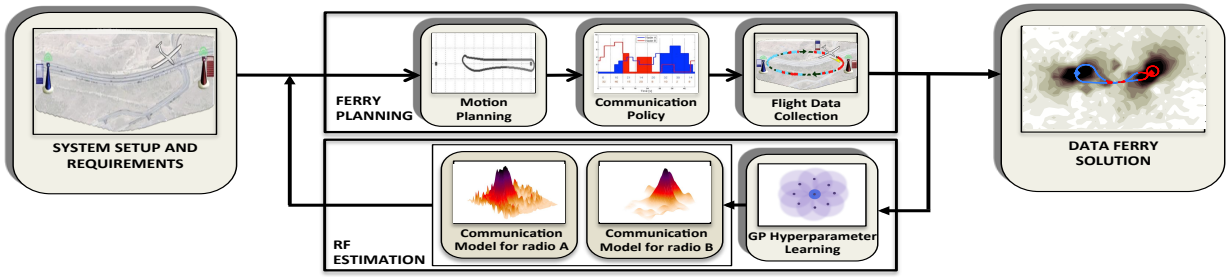


Figure 6.1: Integrating RF Characterization with Data Ferrying involves iterating through planning ferry paths based on a predicted environment, and improving the predictions based on signal strength samples gathered while ferrying.

The GP predictions are then combined with the *a priori* model to improve the RF estimates. At the end of iteration n, the updated models of the RF environments are

$$\begin{aligned} \mathcal{M}_{A,n} &= \Xi_A + \mathcal{GP}_{A,n} \\ \mathcal{M}_{B,n} &= \Xi_B + \mathcal{GP}_{B,n} \end{aligned} \tag{6.9}$$

and are used by the ferry planner in the next iteration to select path $\boldsymbol{p}_{n+1}$. The entire process is summarized in the block diagram shown in Fig. 6.1.

In this manner, improvements learned from the previous iteration's data are incorporated in ferry planning and decision making during the subsequent iteration. The subsequent improvements are a result of accumulating more variation data, which increases the size of the training set, and consequently improves the GP predictions. With better accuracy in the RF models, the ferry can plan a better trajectory.

It is possible that the paths generated between sequential models $\boldsymbol{p_n}$ and $\boldsymbol{p_{n+1}}$ have no intersections if the predicted environments drastically change between $\mathcal{M}_{A,n-1}$ and $\mathcal{M}_{A,n}$. For this work, the transition between paths $\boldsymbol{p_n}$ and $\boldsymbol{p_{n+1}}$ is ignored.

6.3.2 Detailed Integration Considerations

6.3.2.1 Initializing Subsequent Iterations

The genetic algorithm to optimize the ferry path uses a population of 20 chromosomes over 200 generations. The initial 20 chromosomes are initially generated randomly, and evolve according to various

crossover and mutation rates that are tuned to work reasonably well for data ferrying (similar to those in [12]). After 200 generations, the best path is then flown to sample the environments. With these measurements known, this trajectory is then used to seed one chromosome of the next iteration, with random chromosomes for the remaining 19. If the RF environment is perfectly known and unchanging between iterations, then seeding the genetic algorithm in this way ensures the ferry's performance will not decrease through iterations. (Note this is not guaranteed through changing environments, i.e. as the RF model changes through learning.)

Optimizing the GP hyperparameters has a similar initialization process. The first iteration initializes the hyperparameters to a set of values expected to perform decently for most Signal-Strength applications. In subsequent iterations, the GP is optimized over all accumulated data, rather than just the most recent sampling flight data. Since the dataset accumulates, the hyperparameters for each optimization begin with the previous iteration's values. This tends to result in faster GP optimization time.

6.3.2.2 Limited measurement

The data for each iteration's estimates are sampled from optimized ferry paths, which will tend to be flight patterns near and between both nodes. Over the full environments, this leads to a heavy sampling from a limited region. For a ferry that wants to fly mostly between the nodes, such a sampling region is useful. However, this means the predicted values far from this sampling region will have high variances (low confidence), and high potential for error. The result is that the ferry may miss out on trajectories through unexplored pockets of strong communication. Future work will examine integrating an exploration method in the ferry's trajectory optimization to alleviate this issue.

6.3.2.3 Preventing GP Minimization Failure

Another issue from heavily sampling a small region is the risk of the GP over-fitting data. In extreme cases, the optimization of the GP may fail, resulting in divergent signal strength estimates. This type of failure can be detected by examining the log marginal likelihood values returned by the conjugate gradient optimization function: if the negative log likelihood values are higher than a preset threshold, the learning is considered to have failed. Empirically, successful maximum likelihood estimations return values of negative

log likelihood below 10^4; negative log likelihood values above 10^5 indicate that the optimization failed after getting stuck in a local minima. When this happens, the resulting GP is thrown out, and the ferry returns to the previous model (e.g. set $\mathcal{M}_{A,n+1} = \mathcal{M}_{A,n}$ instead of (6.9)). This allows the ferry to plan based on the last-known good environment estimate.

6.4 Case Study: Results and Discussion

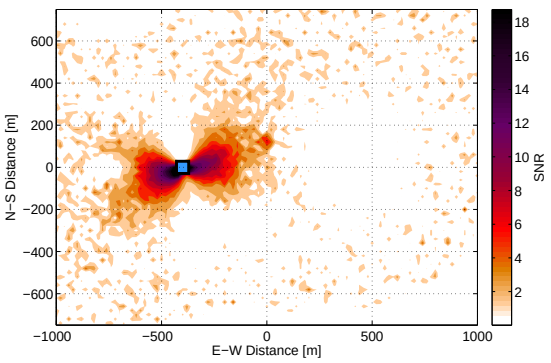
This section presents a simulation case study to illustrate the process of integrating Data Ferrying with Gaussian process RF Estimation, and then to further analyze the system's behavior and performance.

6.4.1 Initial Configuration

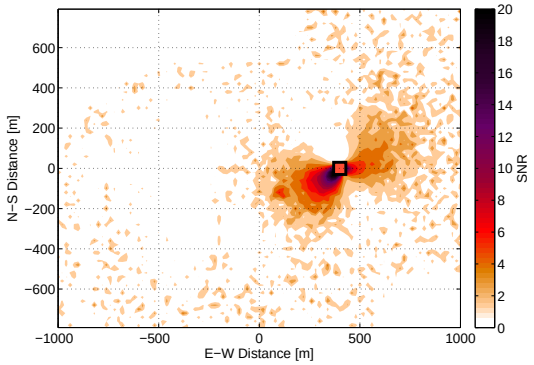
Two nodes are set in an obstacle-free environment, 800 meters apart. The aircraft flies at a constant speed of 25 m/s as it ferries data between the nodes. Its goal is to maximize the amount of data transferred over a closed trajectory with maximum duration of 2 minutes. The true RF environments for the nodes are based on dipole antennas, several interferers and additional RF noise. Though simulated, this type of environment is similar to those commonly seen during flight experiments (as in the previous chapter and [68]). Figure 6.2 shows the mean fields of the RF propagation models for each node. The inherent stochastic nature of the RF environments is reflected by adding a zero-mean normally-distributed variation term $\nu = \mathcal{N}(0, 0.5)$ to this mean field as the ferry flies through these truth environments. Here, the variance is chosen to reflect the stochasticity of RF signal sampling, based on experiments between ground nodes and unmanned aircraft across the UHF communication band; a value of 0.5 balances the low variance seen at 433MHz [69] and with the higher variance seen at 2.4GHz [68].

Note that though the figures are separated, the two environments do overlap; they are displayed separately for clarity of the environments. Significantly, the signal propagation within these RF environments is such that the spacing of the nodes is far enough to make direct communication a challenge; thus the nodes need the aircraft to ferry data for them.

The *a priori* environment estimates $\Xi_{(.)}$ shown in Fig. 6.3 have the correct node locations, but higher power levels, less interferers, and incorrect antenna angles than the true environment.

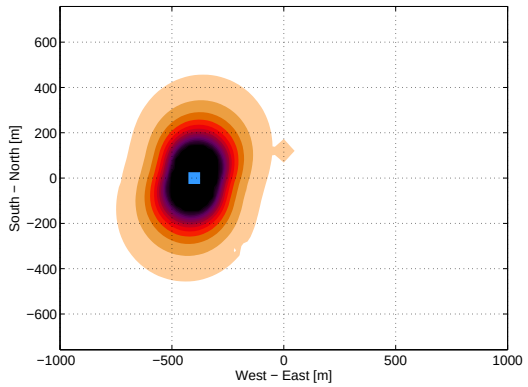


(a) True RF environment for Node A

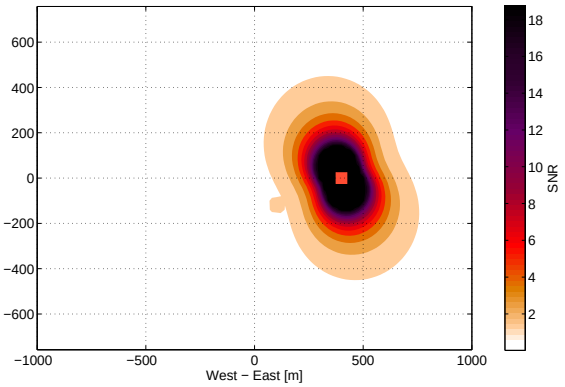


(b) True RF environment for Node B

Figure 6.2: True RF environments.



(a) Node A RF estimate



(b) Node B RF estimate

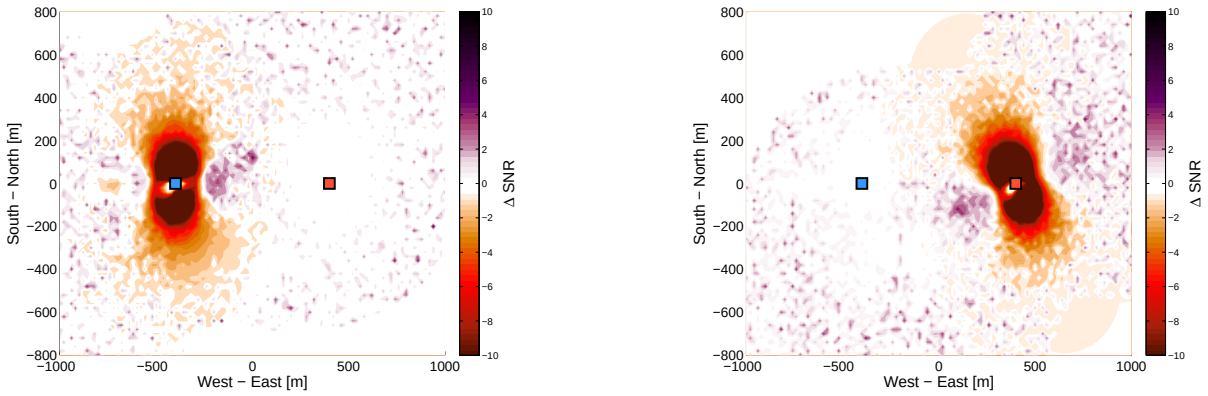
Figure 6.3: Initial estimates of RF environments.

In practice, the combined ferry-and-learn system would continue through the duration of the ferrying aircraft's flight. For this evaluation, the system is run for 20 iterations. The performance of our combined ferry-and-learning system is bounded by what the ferry would achieve if optimized with the true RF environment known; for the environments in Figs. 6.2, the optimal path achieves an effective throughput of 22.4 Mbps.

6.4.2 GP Performance

6.4.2.1 General Behavior

Figure 6.4 shows the RF variations $\mathbf{e}_{(\cdot)}$ from the true environment that are not captured in the *a priori* model, as explained in Eqs. (6.3) and (6.7). These variations are large and negative around the nodes as the *a priori* model's incorrect antenna angles and higher power levels overestimate signal strength. Away from the nodes, the *a priori* model assumes a much cleaner decay to 0 than the true environment, resulting in small positive variations. These RF variations are not Gaussian in nature, and thus cannot be modeled by a single additive white Gaussian noise (AWGN) term as in Eq. (6.1). They do however exhibit spatial similarity in the case of both nodes; it is this location-dependent noise (Eq. (6.2)) that can be learned.

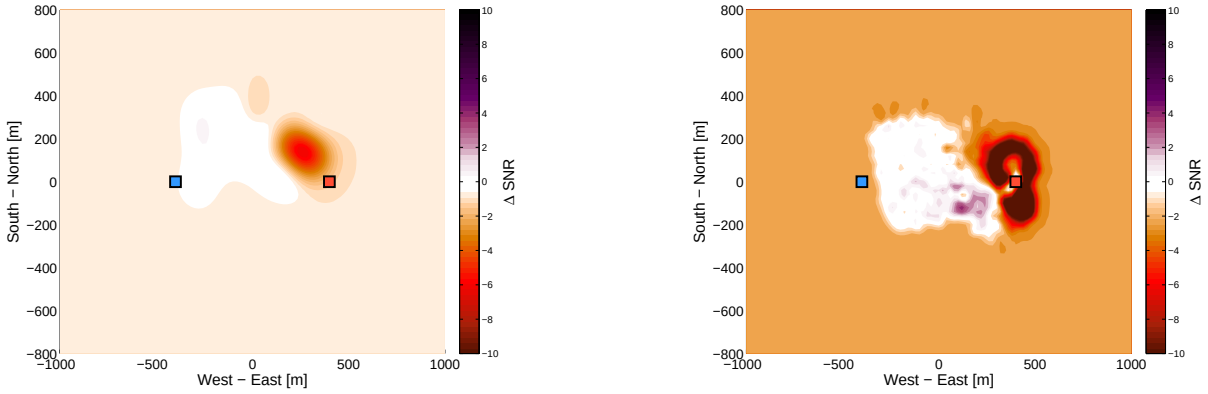


(a) RF variations for node A

(b) RF variations for node B

Figure 6.4: RF variations for the *a priori* models of (a) Node A and (b) Node B.

The goal of the GP is to learn these RF variations from the *a priori* model by capturing the spatial correlations via its hyperparameters. Figures 6.5a and 6.5b show the mean field of the $\mathcal{GP}_{B,1}$ and $\mathcal{GP}_{B,20}$ predictions, respectively. As the training dataset grows with each iteration, the GP learns the RF variations for node B in greater detail by further tuning its hyperparameters. For example, after iteration 1, the GP mean prediction (Fig. 6.5a) has very little detail and has not captured any of the key features of Fig. 6.4b. However, by iteration 20 the mean prediction of GP (Fig. 6.5b) has learned most of the dark red region

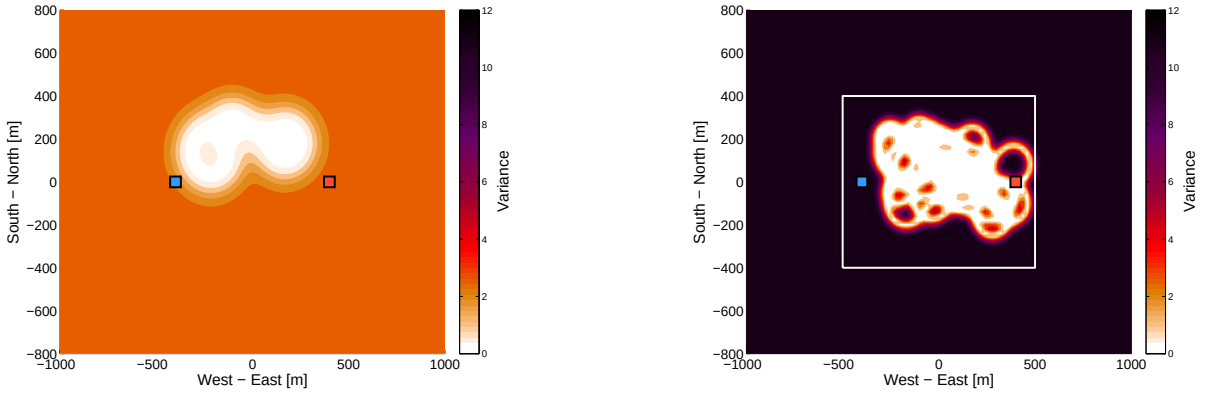


(a) After Iteration 1 (b) After Iteration 20

Figure 6.5: Mean prediction for GP learned on RF Variations for Node B

surrounding node B, and closely matches the initial error (i.e the RF variations of the *a priori* model, Fig. 6.4b).

Besides the mean field, GP predictions also provide a variance, which represents the uncertainty in the mean predictions. Figures 6.6a and 6.6b show the GP variance throughout the environment, at iterations 1 and 20, respectively. The white color marks the regions of very low variance, i.e. high certainty, which



(a) After Iteration 1 (b) After Iteration 20

Figure 6.6: Variance for GP learned on RF variations for Node B

correspond to the sampled region where the ferry has flown. Because the GP's ability to accurately predict decreases further away from the sampled regions, the corresponding variance increases, represented by darker

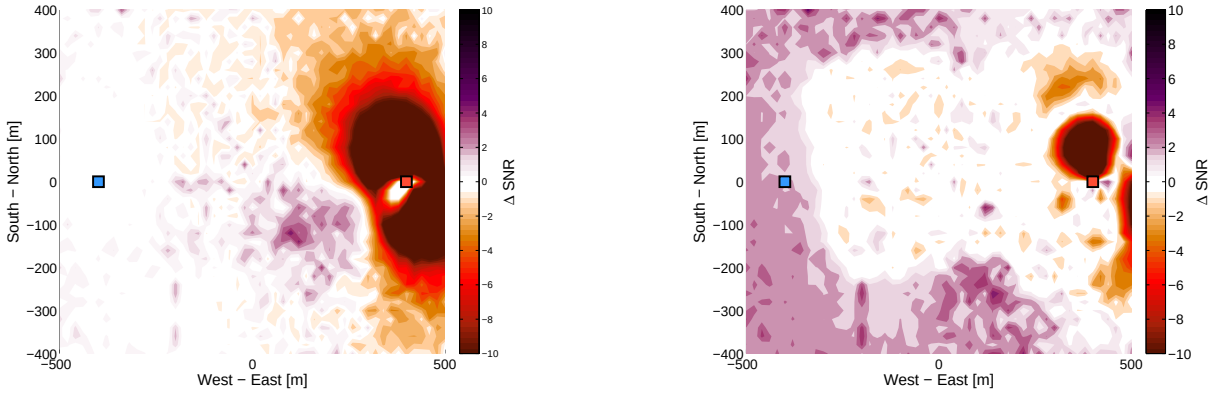
colors. As the ferry samples more of the environment with each iteration, the regions with low variance grow from Fig. 6.6a to Fig. 6.6b. Similar behavior is seen in the GP progression for node A. The white box in Fig. 6.6b represents a symmetric area around the nodes that represents a region intuitively of most interest to a data ferrying unmanned aircraft; and importantly, the variance within this region is significantly reduced.

The largest values of variance in Figs. 6.6a and 6.6b, however, differ greatly. This discrepancy stems from the different hyperparameters learned from different size training sets at these two iterations. At iteration 1, when the training set and sampled region are small, the training samples don't exhibit much variability. Hence, the GP learns hyperparameters with small values for σ_f^2 and σ_n^2, and calculates the maximum variance of 2.81, assuming the rest of the environment has small variance as well. As the training set grows, the GP learns that the signal deviations are in fact highly variable, and it increases the signal variance σ_f^2. Now the maximum variance at iteration 20 is at 11.72 in Fig. 6.6b, which in fact reflects the increase in the uncertainty of the unexplored far away regions, which may also prove to be highly variable.

The value of the GP's learning can be seen by comparing the model error for the *a priori* model $\Xi_{(.)}$ (Fig. 6.7a) with the model error for $\mathcal{M}_{(.),20}$, close to the region of ferrying interest (shown by the white box in Fig. 6.6b). Figure 6.7 zooms into this box, and shows that the high error regions of Fig. 6.7a (represented by the darker colors) have shrunk considerably after 20 iterations. The exception to this is around the left side of Fig. 6.7b with a non-zero error, an area not flown and sampled by the ferry during the 20 iterations.

6.4.2.2 Root Mean Squared Error (RMSE)

The GP and the overall model's prediction performance at any iteration i can be measured by calculating the mean prediction's error on unseen measurement data, i.e. a validation dataset. The errors over the entire validation dataset can be aggregated using root mean squared (RMS) error (Fig. 6.8). The bold lines represent the RMS error for the path-based validation set, which at iteration i comprises of points sampled by the path flown during iteration $i+1$, and are used to evaluate $\mathcal{GP}_i$. This path is optimized given the predictions of $\mathcal{GP}_i$, and being a new path means the new sampled points are not part of $\mathcal{GP}_i$'s training set. The path-based RMS error starts out higher than 2 dB, but converges to 0.5 dB, approaching the variance



(a) Apriori Ξ_B model error

(b) Learned $\mathcal{M}_{B,20}$ model error

Figure 6.7: Model Error Comparison for Node B

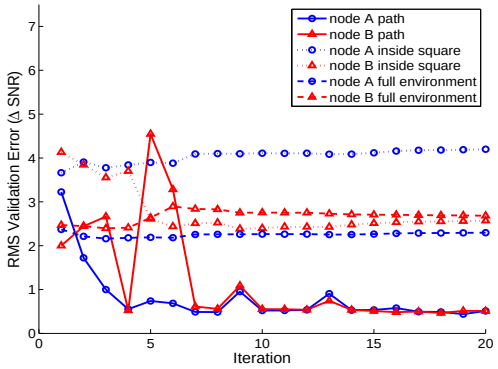


Figure 6.8: GP validation error over 20 Iterations.

seen in the true environment. In fact, by the 7^{th} iteration, the RMSE of A and B reduce by 84% and 69%, respectively. The thick dashed lines in Fig. 6.8 represent the RMS error of mean predictions at locations on a uniform grid over the entire environment, which includes boundary points far outside the sampled regions. Since the ferry paths naturally remain within a small region, the unsampled points dominate this error and prevent it from reducing significantly.

The thin dotted lines in Fig. 6.8 represent the RMSE over the focused ferrying region (defined by the white box in Fig. 6.6b). Note that from Fig. 6.3a, the most significant *a priori* error occurs near the nodes, which bias the RMSE values within this focused region. As the ferry flies through this region, the RMSE for node B in this region reduces below the full environment error, as expected. However, after 20

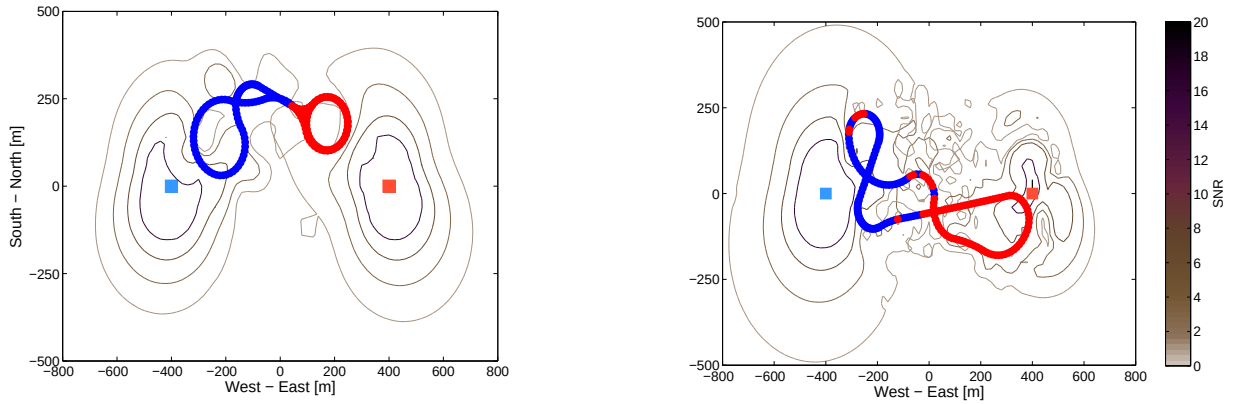
iterations, the model's remaining error is biased by the left side of the region (Fig. 6.7b), which has been left unsampled; hence the region error of 2.57 is only marginally better than the full environment error at 2.68. This is even more drastic for node A, where the unsampled left side of the region includes those high *a priori* errors, which mostly remain through the 20 iterations. This biases the regional RMSE of 4.2 higher than the full environment error of 2.3. Significantly though, the ferrying system is able to perform well despite these errors. Because the true environment for node A includes a strong region to the right of the node (Fig. 6.2a), the ferry planner's erroneously high expectation of throughput around and left of the node was not enough to draw the ferry to that area, leaving that area unsampled. The different behaviors of these regional RMS errors, and the full environment errors, make a compelling case that wider exploration throughout the environment can produce significant reduction in model error, which may help further close the gap between good ferry performance after 20 iterations, and the upper bound in Fig. 6.10.

Further analysis of the GP's performance in the context of data ferrying is presented in [13].

6.4.3 Ferrying Performance

The previous section shows that the GP is able to effectively model the environment over successive iterations. The improved modeling directly correlates to improved ferry performance. Figure 6.9 shows how the optimized paths vary from the first iteration to the 20th. The blue sections of the paths indicate the ferry listening to A, and the red for delivering to B. The first iteration generates a trajectory based on $\Xi_{(.)}$, with an expected throughput of 18.2 Mbps, but actually achieving 6.9 Mbps because of the drastic model error. The contour lines behind the ferry paths represent the predicted RF environments after the GP is trained. Over 20 iterations, the predictions are very different in Fig. 6.9b in comparison to Fig. 6.9a, with much more detail in the contour lines between the two nodes where the ferry has sampled the most data. With a much better prediction here, the ferry can plan to take advantage of rapid variations in signal strength by switching links frequently between A and B. As a result, the ferry's actual performance converges to its predicted performance, ultimately improving to an effective throughput of 21 Mbps.

The ferry's actual performance improves rapidly while also converging to its predicted performance as the GP predictions are refined (Fig. 6.10). The first 4 iterations show quick convergence between the



(a) After Iteration 1

(b) After Iteration 20

Figure 6.9: The ferry path evolves as the RF model predictions are improved from (a) Iteration 1 to (b) Iteration 20.

ferry's expected performance based on $\mathcal{M}_{(.),i}$ and what is actually obtained while flying through the true environment; the trend also shows rapid improvement for both planned and actual performance. The same trend is evident from iterations 5 to 16. After 16 iterations, the solutions have converged to within the signal strength sensing noise.

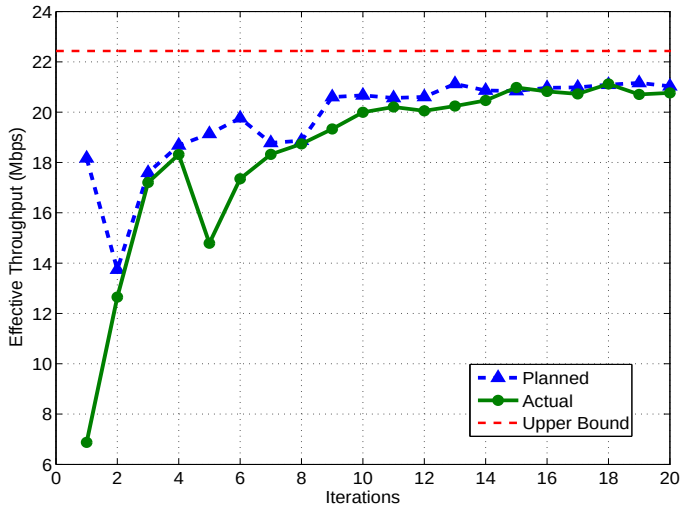
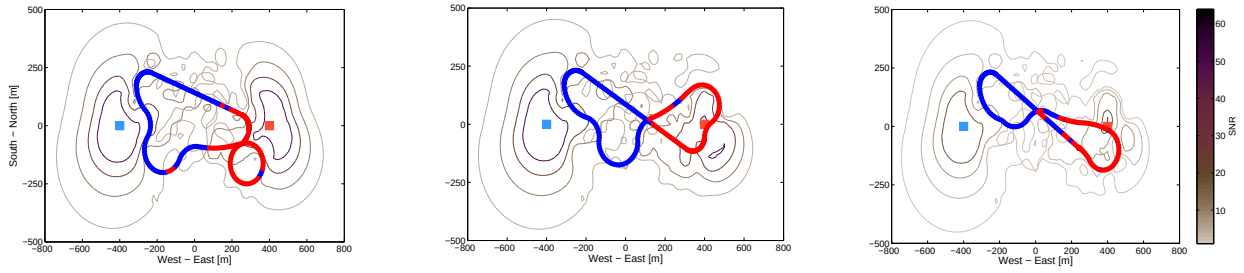


Figure 6.10: Planned and actual ferry throughput performance, ferrying through the estimated RF models and the true RF environment.

The dip in performance at iteration 5 results from a significant change in planned path (Fig 6.11). The ferry path at iteration 4 passes through a region just southwest (down and left) of node B. In sampling this region, the GP determined the signal strength was generally weaker in that region than the *a priori* estimate had predicted. An unsampled region above and right of node B was predicted to have good signal strength (shown by the contour lines in Fig. 6.11a), which would improve the ferry's performance. When flying that path, performance drops as it learns after Iteration 5 that the region has poor signal strength as well. Then with the improved GP predictions of this region, following iterations are able to avoid that region, shown immediately with Iteration 6 (Fig. 6.11c).



(a) Iteration 4 (b) Iteration 5 (c) Iteration 6

Figure 6.11: Evolution of ferry paths and predicted environments from (a) a previous good trajectory through (b) the region north of node B with previously over-estimated signal strength, then learning to avoid that region in (c).

This demonstrates that despite poor *a priori* knowledge of the RF environment, integrating opportunistic learning enables the ferrying system to be extremely effective.

6.5 Learning Comparisons

The ferry-and-learn methodology can use any method for learning RF environments. This section analyzes the choice of Gaussian process learning in comparison to two other common forms of modeling, highlighting the benefits of the GP over the complex RF environments.

6.5.1 Radio Model Fidelity

Improving the RF model with a Gaussian process of RF variations takes a data-driven and nonparametric approach to communication modeling. Because a GP defines a distributions over functions, this single methodology can learn the RF variations stemming from diverse RF behaviors and environmental artifacts. This does not require prior knowledge of the hardware of the nodes, or the presence of interferers. Instead, the GP implicitly captures these factors from the training data via the hyperparameters and the correlations within the measurements. While this approach can be computationally more intensive, it provides a single, flexible method for modeling all kinds of radio propagation, as well as for correcting any inaccurate *a priori* model.

In contrast to a GP, physics-based models have parameterized functional forms for $\Xi_j(p_i)$ in Eq. (6.1), generally assuming specific antenna patterns and how RF transmission would propagate in the environment. These models, then, capture the RF behavior from the training data by parameter estimation. The following are the two examples of physics models, which differ in the type of antenna they model, and the number of parameters they use

6.5.1.1 Omnidirectional Antenna Model

The omnidirectional antenna model [68] assumes that the signal power received at all locations at a range r away from the transmitter can be modeled as

$$P_{Rx,dBm} = 10\log_{10}\left(\frac{k_0}{r^{\gamma}}\right) + \nu. \tag{6.10}$$

This model is an example of the traditional empirical models represented in Equation (6.1), and thus, ν represents the additive white Gaussian noise in the measurements. The term k_0 represents the transmitter's power density, and γ represents the path-loss exponent, and is dependent on environmental factors like atmospheric conditions, and obstacles in the environment.

6.5.1.2 Dipole Antenna Model

The dipole antenna [11] model represents the signal power received at a range r from the transmitter as

$$P_{Rx,dBm} = 10\log_{10}\left(\frac{k_1\sin^2(\xi) + k_2\cos^2(\xi)}{r^\gamma}\right) + \nu \tag{6.11}$$

where, once again, ν and γ are the AWGN and the path-loss exponents. In contrast to the omnidirectional antenna model, the dipole antenna model combines transmit power, gain, and fading effects via directional power density terms k_1 and k_2. ξ is the angle of the aircraft in the frame of the node's antenna pattern:

$$\xi = \psi - \phi \tag{6.12}$$

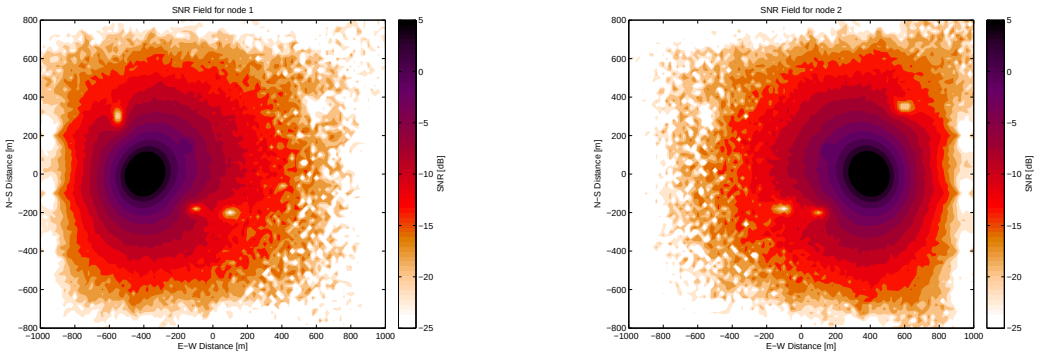
where ψ is the nadir angle on the ground node's antenna, and ϕ is the relative angle (from the axis pointing East) between the aircraft and the ground node. While this model is designed to capture directional effects, it can also model omnidirectional antennas by setting $k_1 = k_2$.

6.5.2 Evaluation Setup

To evaluate how the nonparametric approach of a GP overcomes the limitation of physics-based radio modeling approaches, we compare it to estimation systems based on both the omnidirectional and dipole antenna models. Specifically, the ferry-and-learning system is implemented with an estimator that uses either the omnidirectional or the dipole antenna model in place of the GP. In the case of the physics-based models, at each iteration, a least squares solver uses the accumulated ferry measurements for each node, and calculates the radio parameters that best fit the ferry measurements. For the omnidirectional model, $[k_0, \gamma]$ are estimated with a linear least squares fit, while the dipole model estimates $[k_1, k_2, \psi, \gamma]$ with a nonlinear least squares solver. These parameters are then used for predicting RF signal strengths within the ferry planner in the following iteration. This process is continued for 15 iterations.

The comparison was performed on RF measurements from two specific truth environments: 1. nodes with near-omnidirectional antennas, and little interference and thermal noise (Fig. 6.12), and 2. nodes with highly directional dipole antennas, in an environment with interferers and significant noise (Fig. 6.13). These

cases validate the GP's benefits over the two extremes of RF environments. For a fair comparison between the solvers, and limited by the few parameters of the omnidirectional model, the estimators begin with an *a priori* model (Fig. 6.14) that reflects an omnidirectional antenna with erroneously high power densities, resulting in RMS errors of 13 dB for the near-omni case, and 18 dB for the dipole case.



(a) Mean field of truth SNR for A

(b) Mean field of truth SNR for B

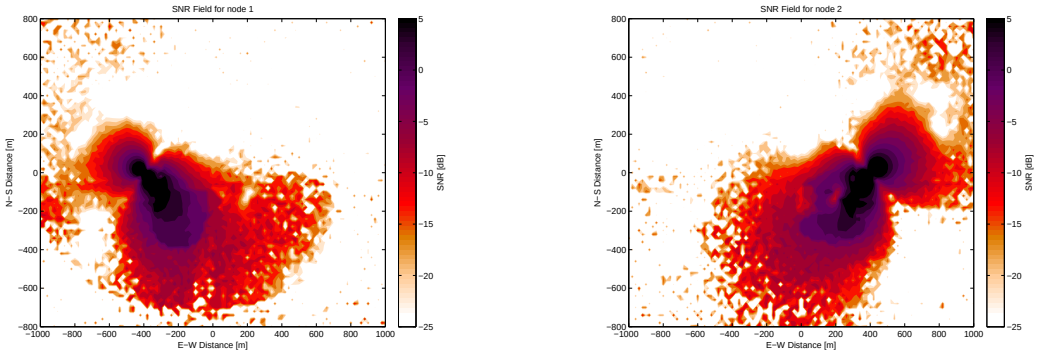
Figure 6.12: Case 1: a relatively clean omnidirectional truth environment.

6.5.3 Comparison Results

6.5.3.1 Estimator Analysis

The estimation errors on the path and full environment validation datasets loosely represent the learners' prediction performances in the region between the nodes and throughput the environment, respectively. Consequently, the root mean squared error (RMSE) for the path validation datasets indicates how well the RF environment is known while the ferry is being planned. On the other hand, the RMSE for the full environment validation dataset indicates a learner's ability to extrapolate to unseen regions of the environment.

When the environment is fairly clean and near-omnidirectional, all three learners perform similarly, reducing the RMSE from the *a priori* 13 dB to 2 or below on the path validation set, as shown in Fig. 6.15. In fact, with all three learners, the error reduces immediately and converges 3 iterations. This is consistent with the ferry performance of all learners exceeding 12 Mbps only after 4 iterations, as shown in Fig. 6.17a.



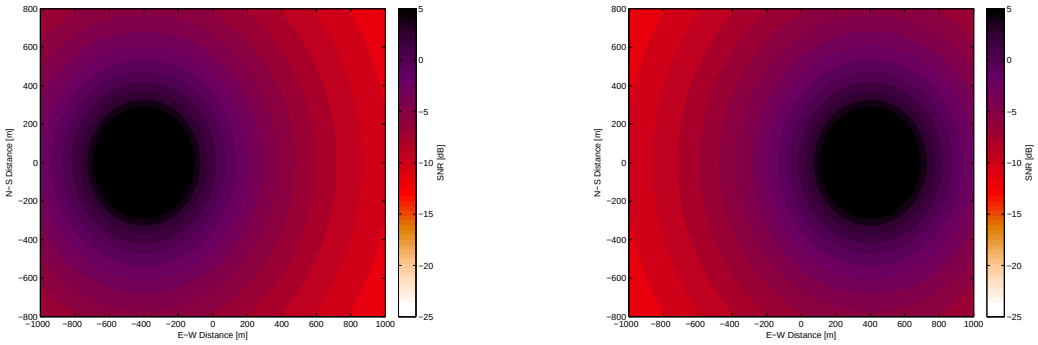
(a) Mean field of truth SNR for A

(b) Mean field of truth SNR for B

Figure 6.13: Case 2: a much more complicated truth environment, with dipole antennas, noise, and many interferers.

In contrast, the path validation RMSE of the three learners after 15 iterations differ greatly in the case of the complicated dipole environment, shown in Fig. 6.16. For both nodes, the RMSE of the linear least squares estimator is above 5.5 dB, and performs the worst of all 3 learners. While the nonlinear least squares estimator performs better than its linear counterpart, with an RMSE higher than 2.6 dB. Only the Gaussian process learner achieves performance comparable to case 1 with a final RMSE below 2 dB for both nodes. However, faced with a more challenging learning task, the convergence in this case is slower, with the errors falling below 2 dB only after 7 iterations once the training set grows sufficiently large.

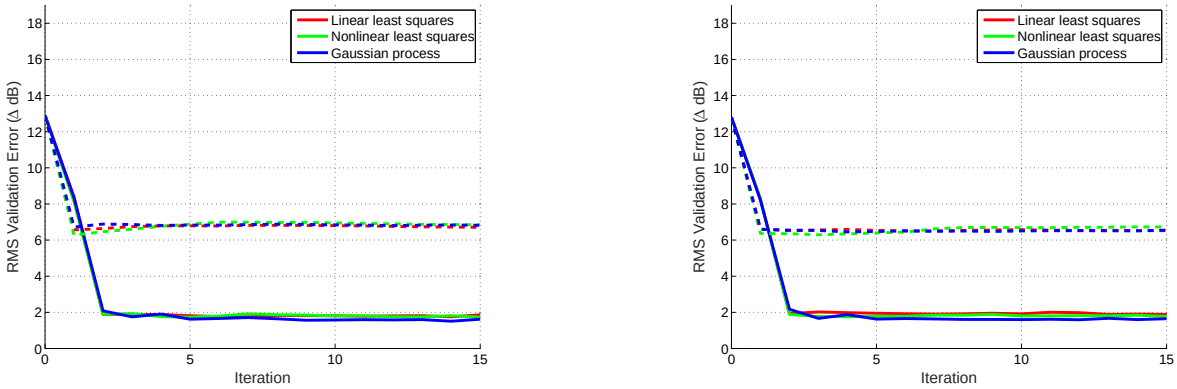
When the true environment closely matches a clean omnidirectional antenna, all learned models extrapolate well to most of the unexplored parts of the environment, as seen by the dashed lines in Fig. 6.15. This is because in case 1 the signal largely decays with distance and the correlations are stationary; and the parameter-based models can capture this trend. This clean trend and stationarity is not present in case 2 with the presence of interferers, making it hard to extrapolate to unexplored regions, consequently increasing the full environment validation RMSE, as shown in Fig. 6.16. In addition, because the GP predicts based on correlations with training data, it especially fails on the boundaries. While the GP does warn against the predicted means in those regions with high variance there, the RMSE cannot capture this indication of uncertainty.



(a) Mean field of *a priori* model for A

(b) Mean field of *a priori* model for B

Figure 6.14: Initial estimates, i.e. *a priori* models of the RF environment, used in both cases.



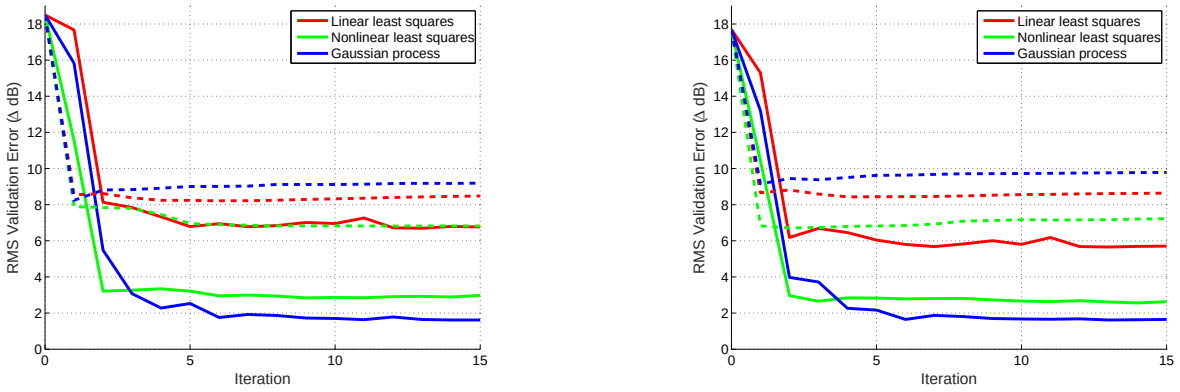
(a) For node A

(b) For node B

Figure 6.15: Path (solid) and full environment (dashed) validation RMS error comparison of the three learners, averaged over 8 runs of the near-omnidirectional environment.

6.5.3.2 Ferrying Performance

The ferry's performance is bounded by what could actually be achieved if the true environment was known accurately. In both cases, optimizing the data ferry with perfect knowledge results in an effective throughput of 16 Mbps, plus or minus 0.5 Mbps depending on the stochastic fluctuations of the RF environments. Because of this randomness, the following results are averaged from each simulation run 8 times.



(a) For node A

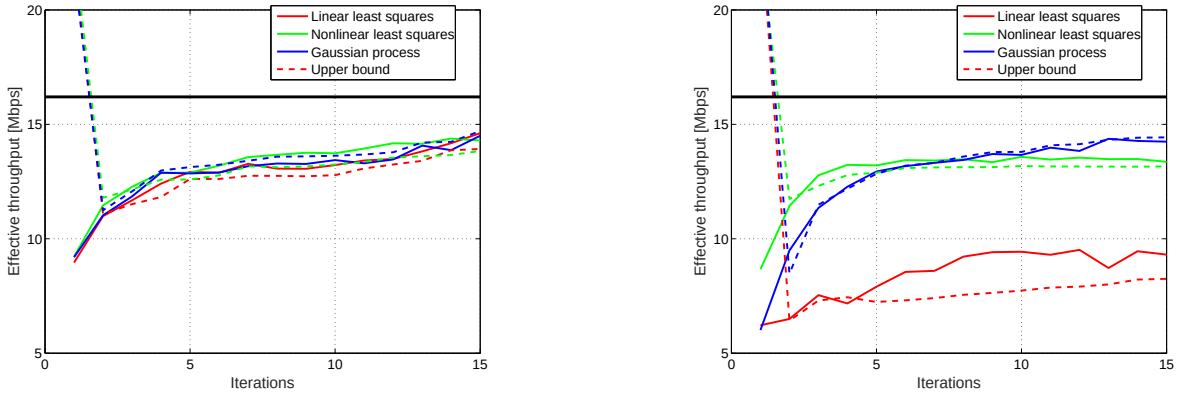
(b) For node B

Figure 6.16: Path (solid) and full environment (dashed) validation RMS error comparison of the three learners, averaged over 8 runs of the complicated dipole environment.

Figure 6.17 shows how the effective throughput that the ferry achieves grows over the iterations, as the learned models of the environment improve. In Case 1, the ferry's actual throughput is 14.62 Mbps for the linear least squares estimator, 14.30 Mbps for nonlinear least squares, and 14.50 Mbps for the Gaussian process. Because the true environment varies only slightly from what Eq. 6.10 can capture, the actual performance for each learning system is fairly close over the 15 iterations.

In contrast, the ferry's throughput for Case 2 shows a much larger variation as the least squares models struggle to capture the nuances of the true RF environments. Not surprisingly, the linear least squares fit has the lowest throughput after 15 iterations at 9.31 Mbps; the nonlinear least squares fit captures the true environments a little better to yield a throughput of 13.36 Mbps; the GP captures the true environment best, enabling a ferry performance of 14.61 Mbps.

The differences between the expected performance (dashed lines) and the actual performance (solid lines) in Fig. 6.17 further compare the learner performances. In the first case, although the actual performances are close among the three learners, the model error remaining after 15 iterations using the linear least squares model results in a performance error of 0.69 Mbps; the nonlinear least squares model is a little better at 0.46 Mbps; the GP captures the truth model the best, with a performance error of only 0.21 Mbps. In the more complex environments of Case 2, the errors have a larger range, where the performance error



(a) Case 1: near-omnidirectional environment

(b) Case 2: dipole environment

Figure 6.17: Ferry's expected (dashed) and actual (solid) throughput performance compared between the three learners, averaged over 8 runs.

for the linear least squares model is 1.1 Mbps; the nonlinear least squares has an error of only 0.28 Mbps; and the GP has a ferry performance error of only 0.19 Mbps.

Though the least squares errors are larger than those of the GP, an error of 1.1 Mbps is not enormous; that is, even the linear least squares model is able to somewhat predict the actual performances of its resulting ferry path; and the nonlinear least squares model predicts performance extremely well. This initially indicates that the least squares models are estimating the RF world reasonably well, despite achieving a lower ferrying throughput. However, this is actually because of model over-fitting. In the iterative ferry-and-learn system, a path is determined based on a limited model; the models are then fit to the signal strength data sampled along these paths. The paths and model predictions eventually stabilize, with the path changing very little, and the models then being fit to very similar data.

This behavior is seen after Iteration 10 in Fig. 6.17b for the two Least Squares models, where the ferry's performance flattens, and the predicted performance converges to it. The same condition can occur for the ferry-and-learn setup with the Gaussian process. However, because of the limitations of the least squares models, the stabilized models and resulting paths are stuck at lower local maxima than that of the GP. To reduce the probability of any models and paths getting stuck in local maxima, future work will investigate including an exploration component into path planning.

Table 6.1 summarizes the effectiveness of each learner, and its impact on ferrying performance. The Gaussian process does not predict the RF environment over the full area as well as the nonlinear least squares parameter-based model. However, learning how complicated the RF environment is from the limited data it has seen, the GP better captures the stochastic characteristics of the environment. Comparing the full probabilistic predictions of the three learners shows how the GP outperforms the least squares models. Thus, the GP is able to adapt to especially complex environments over observed areas; and with the best RMSE in the general ferrying region, the GP enables the best and most accurate ferrying performance.

Table 6.1: Radio Model Comparison Summary.

Metric	Node	Near-omnidirectional environment			Complicated dipole environment		
		Linear	Nonlinear	GP	Linear	Nonlinear	GP
Ferry Throughput [Mbps]	-	14.62	14.30	14.50	9.31	13.36	14.61
Ferry Throughput Error [Mbps]	-	0.69	0.46	0.21	1.1	0.28	0.19
Path Validation RMSE [dB]	A	1.86	1.74	1.62	6.77	2.98	1.62
	B	1.89	1.77	1.65	5.70	2.63	1.62
Environment Validation RMSE [dB]	A	6.70	6.85	6.82	8.48	6.82	9.19
	B	6.52	6.73	6.55	8.63	7.22	9.78

6.6 Summary

This chapter addresses the issue of uncertain RF communication environments when planning ferry routes. Errors between an *a priori* model and the true environment are sampled as the aircraft ferries data. These errors are fed to a Gaussian process to better predict the RF environment elsewhere, leading to more accurate plans for the ferry's path.

The simulation studies showed the adaptability of the GP in comparison to parameter-based estimators. Through relatively clean RF environments where the parameter-based methods perform well, the GP performs just as well and converges just as quickly. In more complicated environments, the non-parametric nature of the GP allows it to more accurately capture complex RF behaviors in the ferrying region between the sensor nodes. Further, the quick convergence of the ferry-and-learn process to improve RF models enables the ferry to plan better paths and increase effective throughout with the nodes.

Chapter 7

CONCLUSION

This dissertation presented a cascaded method for optimizing a persistent data ferrying system. Specifically, by taking advantage of the natural cascaded structure between motion planning from communication link scheduling, the link scheduling optimization can be implemented by a policy that depends on a given trajectory. Associated link scheduling policies were developed and assessed, and scale smoothly to scenarios of many source nodes and one destination. The combined system was shown to significantly reduce computation time for ferry planning, while achieving solutions with high ferry system performance.

7.1 Cascaded Formulation for Data Ferrying

The fundamental basis of this dissertation is that the communication and motion dynamics in the data ferry problem are naturally cascaded. By exploiting this fact, the data ferry problem was reformulated where the independent control was reduced to motion control, while link scheduling could be solved by an optimal policy based on the given motion control. Insights about the system dynamics further resulted in derived necessary conditions for optimality within periodic ferry paths, specifically that all data that is collected within the period must also be delivered. Simulations demonstrated that this formulation has the effect of reducing computational effort required, and was proved that this computational reduction maintains the quality of the ferrying solution. Significantly, this formulation applies to ferrying systems covering a vast range of motion dynamics and complicated RF environments.

7.2 Fast Link Scheduling Policies

The cascaded formulation encapsulates link scheduling within the motion control's objective function, which obfuscates gradient information from the control. Thus, optimization tools must make many function evaluations, solving the NP-Hard link scheduling policy for every candidate motion control. However, the link scheduling policy was adapted to the well-studied Knapsack problem, and two heuristic algorithms were developed to approximate the solution to this combinatorial optimization problem. The Greedy Knapsack Heuristic was adapted from the Knapsack literature to be applied to ferrying, and the Most Data First algorithm applies a fairness technique to meet the derived necessary condition for periodic data ferrying paths.

Theoretical algorithm analysis and Monte Carlo Simulations demonstrated that the Greedy Knapsack and Most Data First algorithms are on the same order of magnitude as common heuristics based on simplified communication environments; they are significantly faster than both binary integer programming and the linear programming relaxation methods. Able to account for complex environments, the resulting ferry performance is much higher with these new algorithms. Further, they are easily extendible to ferrying systems involving a single destination node requesting data from many source nodes.

7.3 Experimental Assessment

A throughput sampling system was integrated with the unmanned aircraft platform, enabling persistent connections for nodes to continue testing despite connection dropouts, as will happen with data ferry applications. Link policies and the cascaded planning framework showed similar behaviors when planning with experimental data as with the simulated environments.

The assumption of trajectory following, requiring accurate timing and positioning of the aircraft to ferry data according to plan, proved to be a challenging aspect to fielding a ferrying system. Integrating the planner and link scheduler with the autopilot by mapping commands to a waypoint plan demonstrated functionality. However, improvements to the autopilot's tracking system, as well as communication between autopilot and mission computer, are necessary to reduce the errors seen in flight experiments. Despite errors

in path-following, link switch timing, and time-varying RF environments, the experiments demonstrated the benefits of smarter link-scheduling methods.

7.4 Opportunistic Learning of the RF Environment

In light of the significant challenges presented by dynamic communication environments, even with stationary ground nodes, a Gaussian process learner was incorporated with the ferry planning process to opportunistically learn and improve environment models. With these improvements, the ferry's performance was shown in simulation to reduce error between expected behavior and actual behavior, ultimately improving the full system's performance as well. Further, modeling the environment in a non-parametric way enabled the learned model to capture even the complex characteristics of the environment, and do so much better than parameter-based models. By being able to learn and improve models, and adapt to changes within this framework, the fast cascaded ferry solution method demonstrates potential to handle even more challenging scenarios.

7.5 Future Work

A fundamental aspect of this work is that the performance of the ferry planning system depends on how accurately the communication throughput can be predicted, and how well the plane can follow the planned trajectory. If the path isn't followed exactly as planned, such as if a gust of wind blows the aircraft off course, discrepancies arise between the planned and actual data transferred. This aspect was demonstrated in Ch. 5, and this motivates two main directions for future studies to follow.

First, flight experiments in Chapter 5, as well as in other literature ([53, 68, 76], show how stochastic the 2.4 GHz spectrum can be. Chapter 6 demonstrated the ability to improve model predictions by learning in an opportunistic fashion while ferrying. Within the explore-vs-exploit paradigm, this ferry-and-learn process relates to the extreme of pure exploitation, where the aircraft's path is based on fully exploiting current knowledge. Alternative approaches could explore running a pure exploration phase for an initial amount of time to build the *a priori* models, sacrificing ferrying time; but the knowledge and confidence gained may prove beneficial over a long horizon. Additionally, the ferry's path planning can incorporate the covariance

estimates provided by the Gaussian Process, motivating either an aggressive exploration component, or a conservative reliability component to the ferrying system.

Second, path tracking should be improved, either with enhanced flight system, or a controller more explicit than waypoint-based navigation. The field of path following is extensive, and several techniques may be more robust to motion control noise, such as wind gusts. Further, ferrying in a feed-forward fashion is inherently not robust to errors or disturbances. Because of this, future work should investigate the use of a receding-horizon or reactive planner. If at any given point the data transferred is less than what was planned, the ferry may react by continuing that link connection longer to transfer the necessary amount. The link scheduling policies in Chapter 4 are easily extended to handle offsets to handle replanning in the event that more data was collected than expected. Receding horizon planners could also adjust for necessary course corrections, adapting link schedules as appropriate as well.

A receding horizon controller could also potentially handle changes in communication environments based on node movement. This dissertation focused on stationary nodes with a single data ferry. However, the cascaded system dynamics still apply to a system including mobile sources and destinations, as well as multiple ferries. Exploring the performance of cascaded ferrying and link scheduling policies through these more complex scenarios will yield great insights to the area of motion control for communication services.

Bibliography

[1] Openwrt linux. `http://openwrt.org/`.

[2] Ubiquiti. `http://www.ubnt.com/`.

[3] T Abatzoglou and B O'Donnell. Minimization by coordinate descent. Journal of Optimization Theory and Applications, 36(2):163–174, 1982.

[4] Christian Becker and Gregor Schiele. New mechanisms for routing in ad hoc networks through world models. In Proceedings of the 4th CaberNet Plenary Workshop, volume 2, pages 1–4, Pisa, Italy, 2001. IST/ICT.

[5] Richard Ernest Bellman. Dynamic Programming. Princeton University Press, 1957.

[6] Deepak Bhadauria, Onur Tekdas, and Volkan Isler. Robotic data mules for collecting data over sparse sensor fields. Journal of Field Robotics, 28(3):388–404, 2011.

[7] Christopher M. Bishop. Pattern recognition and machine learning. Springer New York, 2009.

[8] Timothy X Brown, Brian M Argrow, Eric W Frew, Cory Dixon, Daniel Henkel, Jack Elston, and Harvey Gates. Experiments using small unmanned aircraft to augment a mobile ad hoc network. Emerging Technologies in Wireless LANs: Theory, Design, and Deployment, pages 123–145, 2007.

[9] Anthony Carfang, Eric W. Frew, and Timothy X Brown. Improved delay-tolerant communication by considering radio propagation in planning data ferry navigation. In AIAA Guidance, Navigation and Control Conference, pages 1–14, Toronto, Canada, August 2010.

[10] Anthony Carfang, Mikhail Zaturenskiy, Joseph Lloyd, and Dennis Roberson. Performance of wi-fi and bluetooth in mobile ad hoc networks. In Mobile Vehicular Networks, 2008. MoVeNet 2008. 2nd IEEE International Workshop on, pages 701–706. IEEE, 2008.

[11] Anthony J. Carfang and Eric W. Frew. Real-Time estimation of wireless Ground-to-Air communication parameters. In International Conference on Computing, Networking and Communications, Wireless Ad Hoc and Sensor Networks Symposium (ICNC'12 - WAHS), pages 975–979, Maui, Hawaii, USA, January 2012. IEEE.

[12] Anthony J Carfang, Eric W Frew, and Derek B Kingston. Cascaded optimization of aircraft trajectories for persistent data ferrying. Journal of Aerospace Information Systems, pages 1–14, 2014.

[13] Anthony J Carfang, Neeti Wagle, and Eric W Frew. Integrating nonparametric learning with path planning for data ferry communications. Journal of Aerospace Information Systems, pages 1–18, To Appear, 2015.

[14] Guner D. Celik and Eytan Modiano. Dynamic vehicle routing for data gathering in wireless networks. In Decision and Control (CDC), 2010 49th IEEE Conference on, pages 2372–2377. IEEE, 2010.

[15] Zong Da Chen, HT Kung, and Dario Vlah. Ad hoc relay wireless networks over moving vehicles on highways. In Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing, pages 247–250. ACM, 2001.

[16] Cory Dixon. Controlled Mobility of Unmanned Aircraft Chains to Optimize Network Capacity in Realistic Communication Environments. PhD thesis, University of Colorado, 2010.

[17] Cory Dixon and Eric W. Frew. Maintaining optimal communication chains in robotic sensor networks using mobility control. ACM/Springer Mobile Networks and Applications, Special Issues on Mobility of Systems, Users, Data and Computing(3):281–291, June 2009.

[18] Cory Dixon, Dan Henkel, Eric W. Frew, and Timothy X. Brown. Phase transitions for controlled mobility in wireless ad hoc networks. In AIAA's Guidance, Navigation and Control (GNC) Conference and Exhibit, number AIA-2006-6464, pages 1–11, Keystone, CO, 21-24 Aug. 2006.

[19] Lester E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. American Journal of mathematics, 79(3):497–516, 1957.

[20] Jack S. Elston, Jason Roadman, Maciej Stachura, Brian Argrow, Adam Houston, and Eric W. Frew. The tempest unmanned aircraft system for in situ observations of tornadic supercells: Design and vortex2 flight results. Journal of Field Robotics, 28(4):461–483, July 2011.

[21] Atilla Eryilmaz, Rayadurgam Srikant, and James R Perkins. Throughput-optimal scheduling for broadcast channels. In ITCom 2001: International Symposium on the Convergence of IT and Communications, pages 70–78. International Society for Optics and Photonics, 2001.

[22] Jonathan Fink and Vijay Kumar. Online methods for radio signal mapping with mobile robots. 2010 IEEE International Conference on Robotics and Automation, pages 1940–1945, May 2010.

[23] Christodoulos A Floudas. Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications: Fundamentals and Applications. Oxford University Press, USA, 1995.

[24] C Francolin, A Rao, Christiane Duarte, and Gerald Martel. Optimal control of an autonomous surface vehicle to improve connectivity in an underwater vehicle network. Journal of Aerospace Computing, Information, and Communication, 9(1):1–13, 2012.

[25] Camila C. Françolin, Anil V. Rao, Christiane Duarte, and Gerald Martel. Optimization of the motion of a mobile gateway to improve connectivity in a network of autonomous underwater vehicles. In AIAA Guidance, Navigation and Controls Conference, pages 1–20, Toronto, Canada, August 2010.

[26] László Gerencsér, Stacy D Hill, Zsuzsanna Vago, and Zoltan Vincze. Discrete optimization, spsa and markov chain monte carlo methods. In American Control Conference, 2004. Proceedings of the 2004, volume 4, pages 3814–3819. IEEE, 2004.

[27] Alireza Ghaffarkhah and Yasamin Mostofi. Optimal motion and communication for persistent information collection using a mobile robot. In Globecom Workshops (GC Wkshps), pages 1532–1537. IEEE, 2012.

[28] Alireza Ghaffarkhah, Yuan Yan, and Yasamin Mostofi. Dynamic coverage of time-varying environments using a mobile robot–a communication-aware perspective. In IEEE Globecom International Workshop on Wireless Networking for Unmanned Autonomous Vehicles, pages 1–6. IEEE, 2011.

[29] Philip E Gill, Walter Murray, and Margaret H Wright. Practical optimization. Academic press, 1981.

[30] John C Gittins. Bandit processes and dynamic allocation indices. Journal of the Royal Statistical Society. Series B (Methodological), pages 148–177, 1979.

[31] Alejandro Gonzalez-Ruiz, Alireza Ghaffarkhah, and Yasamin Mostofi. A comprehensive overview and characterization of wireless channels for networked robotic and control systems. Journal of Robotics, 2011, 2012.

[32] Robert Grande, Thomas Walsh, and Jonathan How. Sample efficient reinforcement learning with gaussian processes. In Proceedings of the 31st International Conference on Machine Learning (ICML-14), pages 1332–1340, 2014.

[33] Matthias Grossglauser and David N.C. Tse. Mobility increases the capacity of ad hoc wireless networks. Networking, IEEE/ACM Transactions On, 10(4):477–486, 2002.

[34] Ajay Chandra V Gummalla and John O Limb. Wireless medium access control protocols. Communications Surveys & Tutorials, IEEE, 3(2):2–15, 2000.

[35] Dan Henkel and Timothy X. Brown. On controlled node mobility in delay-tolerant networks of unmanned aerial vehicles. In International Symposium on Advanced Radio Technologies, pages 1–10, Boulder, CO, March 2006.

[36] Dan Henkel and Timothy X. Brown. Towards autonomous data ferry route design through reinforcement learning. In World of Wireless, Mobile and Multimedia Networks, pages 1–6, June 2008.

[37] Geoffrey A. Hollinger, Sunav Choudhary, Parastoo Qarabaqi, Christopher Murphy, Urbashi Mitra, Gaurav S. Sukhatme, Miica Stojanovic, Hanumant Singh, and Franz Hover. Underwater data collection using robotic sensor networks. IEEE Journal on Selected Areas in Communications, 30(5):899–911, 2012.

[38] Geoffrey A. Hollinger and Gaurav S. Sukhatme. Sampling-based robotic information gathering algorithms. International Journal of Robotics Research, 33(9):1271–1287, 2014.

[39] IEEE Computer Society. IEEE Std 802.11-2007. 3 Park Avenue, New York, NY, June 2007.

[40] Sushant Jain, Rahul C. Shah, Waylon Brunette, Gaetano Borriello, and Sumit Roy. Exploiting mobility for energy efficient data collection in wireless sensor networks. ACM/Springer Mobile Networks and Applications Journal, pages 327–339, 2006.

[41] Hans Kellerer, Ulrich Pferschy, and David Pisinger. Knapsack Problems. Springer Verlag, 2004.

[42] Andrew T. Klesh, Pierre T. Kabamba, and Anouck R. Girard. Path planning for cooperative time-optimal information collection. In American Control Conference, pages 1991–1996, Seattle, Washington, USA, June 2008. IEEE.

[43] J.C. Lagarias, J.A. Reeds, M.H. Wright, and P.E. Wright. Convergence properties of the Nelder-Mead simplex method in low dimensions. SIAM Journal on Optimization, 9(1):112–147, 1999.

[44] Eugene L Lawler, Jan Karel Lenstra, AHG Rinnooy Kan, and David B Shmoys. The traveling salesman problem: a guided tour of combinatorial optimization, volume 3. Wiley New York, 1985.

[45] Jerome Le Ny. Performance Optimization for Unmanned Vehicle Systems. PhD thesis, Massachusetts Institute of Technology, 2008.

[46] J.A. Maglicane. Splat! an rf signal propagation, loss and terrain analysis tool. `http://www.qsl.net/kd2bd/splat.html`, 2013.

[47] John L Maryak and Daniel C Chin. Global random optimization by simultaneous perturbation stochastic approximation. Automatic Control, IEEE Transactions on, 53(3):780–783, 2008.

[48] Daniel Medina, Felix Hoffmann, Serkan Ayaz, and Carl-Herbert Rokitansky. Feasibility of an aeronautical mobile ad hoc network over the north atlantic corridor. In Sensor, Mesh and Ad Hoc Communications and Networks (SECON'08). 5th Annual IEEE Communications Society Conference on, pages 109–116. IEEE, June 2008.

[49] R. Moazzez-Estanjini and I.C. Paschalidis. On delay-minimized data harvesting with mobile elements in wireless sensor networks. Ad Hoc Networks, 2012.

[50] Yasamin Mostofi. Decentralized communication-aware motion planning in mobile networks: An information-gain approach. Journal of Intelligent and Robotic Systems, 56(1-2):233–256, 2009.

[51] Yasamin Mostofi, Mehrzad Malmirchegini, and Alireza Ghaffarkhah. Estimation of communication signal strength in robotic networks. In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pages 1946–1951. IEEE, 2010.

[52] Randolph Nelson and Leonard Kleinrock. Spatial tdma: A collision-free multihop channel access protocol. Communications, IEEE Transactions on, 33(9):934–944, 1985.

[53] Calvin Newport, David Kotz, Robert S. Gray, Jason Liu, Yougu Yuan, and Chip Elliott. Experimental evaluation of wireless simulation assumptions. Simulation, (9):643–661, 2007.

[54] Hoa G Nguyen, Hobart R Everett, Narek Manouk, and Ambrish Verma. Autonomous mobile communication relays. In AeroSense 2002, pages 50–57. International Society for Optics and Photonics, 2002.

[55] Karl J. Obermeyer. Path planning for a uav performing reconnaissance of static ground targets in terrain. In AIAA Guidance, Navigation, and Control Conference, 10-13 Aug. American Institute of Aeronautics and Astronautics, 1801 Alexander Bell Dr., Suite 500 Reston VA 20191-4344 USA,, 2009.

[56] Christos H Papadimitriou and Kenneth Steiglitz. Combinatorial optimization: algorithms and complexity. Courier Corporation, 1998.

[57] Andrei Patrascu and Ion Necoara. Efficient random coordinate descent algorithms for large-scale structured nonconvex optimization. Journal of Global Optimization, pages 1–28, Feb 2014.

[58] B. Pearre and T.X. Brown. Model-free trajectory optimization for wireless data ferries among multiple sources. In GLOBECOM Workshops (GC Wkshps), 2010 IEEE, pages 1793–1798. IEEE, 2010.

[59] Ben Pearre and Timothy X Brown. Model-free trajectory optimisation for unmanned aircraft serving as data ferries for widespread sensors. Remote Sensing, 4(10):2971–3005, 2012.

[60] Luciana Pelusi, Andrea Passarella, and Marco Conti. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. Communications Magazine, IEEE, 44(11):134–141, 2006.

[61] T.S. Rappaport. Wireless communications: principles and practice, volume 207. Prentice Hall PTR New Jersey, 1996.

[62] C.E. Rasmussen. Gaussian Processes for Machine Learning. Advanced Lectures on Machine Learning, 3176:63–71, 2004.

[63] Izhak Rubin and Runhe Zhang. Placement of uavs as communication relays aiding mobile ad hoc wireless networks. In Military Communications Conference, 2007. MILCOM 2007. IEEE, pages 1–7. IEEE, 2007.

[64] J. Ryu, L. Ying, and S. Shakkottai. On efficient data transport with mobile carriers. IEEE Journal on Selected Areas in Communications, 29(10):1991–2001, 2011.

[65] Rahul C Shah, Sumit Roy, Sushant Jain, and Waylon Brunette. Data mules: Modeling a three-tier architecture for sparse sensor networks. Ad Hoc Networks, 1(2):215–233, 2003.

[66] Arun Agrahara Somasundara, Aditya Ramamoorthy, and Mani B. Srivastava. Mobile element scheduling with dynamic deadlines. IEEE Transactions on Mobile Computing, 6(4):395–410, April 2007.

[67] James C Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. Automatic Control, IEEE Transactions on, 37(3):332–341, 1992.

[68] Maciej Stachura and Eric W. Frew. Cooperative target localization with a communication aware unmanned aircraft system. AIAA Journal of Guidance, Control, and Dynamics, 2011.

[69] Maciej Stachura, Neeti Wagle, and Eric W Frew. A Comparison of Filters for UAS-Based Localization of Stationary RF Sources. In AIAA Aviation Forum, Dallas, Texas, June 2015.

[70] Ryo Sugihara and Rajesh K. Gupta. Improving the data delivery latency in sensor networks with controlled mobility. Distributed Computing in Sensor Systems, 5087:386–399, 2008.

[71] M. M. B. Tariq, M. Ammar, and E. Zegura. Message ferry route design for sparse ad hoc networks with mobile nodes. In Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing, pages 48–59, Florence, IT, May 2006. Association for Computing Machinery.

[72] Ajay Tirumala, Feng Qin, Jon Dugan, Jim Ferguson, and Kevin Gibbs. Iperf: The tcp/udp bandwidth measurement tool. `http://iperf.fr/`, 2005.

[73] Volker Turau, Christoph Weyer, and Christian Renner. Efficient slot assignment for the many-to-one routing pattern in sensor networks. In Proc. Intl. Workshop on Sensor Network Engineering. Citeseer, 2008.

[74] Amin Vahdat, David Becker, et al. Epidemic routing for partially connected ad hoc networks. Technical report, Technical Report CS-200006, Duke University, 2000.

[75] Andrew J Viterbi et al. CDMA: principles of spread spectrum communication, volume 129. Addison-Wesley Reading, 1995.

[76] Neeti Wagle and Eric Frew. Spatio-temporal characterization of airborne radio frequency environments. In GLOBECOM Workshops (GC Wkshps), 2011 IEEE, pages 1269–1273. IEEE, 2011.

[77] Matthew Wall. Galib: A c++ library of genetic algorithm components. Mechanical Engineering Department, Massachusetts Institute of Technology, 87:54, 1996.

[78] Peter Whittle. Multi-armed bandits and the gittins index. Journal of the Royal Statistical Society. Series B (Methodological), pages 143–149, 1980.

[79] WiFi-Plus. Wifi 2.4ghz antennas & spec sheets. `http://www.wifi-plus.com/antennas/24ghzantennasspecs.html`, 2011.

[80] M. Ye, X. Tang, and D. L. Lee. Fair delay tolerant mobile data ferrying. In Tenth International Conference on Mobile Data Management: Systems, Services and Middleware, pages 182–191, 2009.

[81] Pengcheng Zhan, Kai Yu, and ALee Swindlehurst. Wireless relay communications with unmanned aerial vehicles: Performance and optimization. Aerospace and Electronic Systems, IEEE Transactions on, 47(3):2068–2085, 2011.

[82] Xiaolan Zhang, Jim Kurose, Brian Neil Levine, Don Towsley, and Honggang Zhang. Study of a bus-based disruption-tolerant network: mobility modeling and impact on routing. In Proceedings of the 13th annual ACM international conference on Mobile computing and networking, pages 195–206. ACM, 2007.

[83] Yin Zhang. Solving large-scale linear programs by interior-point methods under the matlab environment. Optimization Methods and Software, 10(1):1–31, 1998.

[84] Z. Zhang and Z. Fei. Route design for multiple ferries in delay tolerant networks. In IEEE Communications Society's Wireless Communications Networking Conference, pages 3460–3465. IEEE, June 2007.

[85] Zhensheng Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: Overview and challanges. Communications Surveys Tutorials, 8(1):24–37, 2006.

[86] Wenrui Zhao, Mostafa Ammar, and Ellen Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing, pages 187–198. Association for Computing Machinery, 2004.

[87] Wenrui Zhao, Mostafa Ammar, and Ellen Zegura. Controlling the mobility of multiple data transport ferries in a delay-tolerant network. In INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, pages 1407–1418. IEEE, 2005.

Appendix A

A.1 Pareto-Optimal Bounds

This section derives the performance bounds at the endpoints of the Pareto-Optimal curve that represents the tradeoff of throughput and delay in a data ferrying system. Let R_A^Γ represent the average communication rate obtained between the ferry and the source node A over a trajectory Γ, and R_B^Γ represent the average communication rate between ferry and destination node B. The bounds of net effective throughput from source to destination occur in two places: (i) with minimal allowable delay; (ii) with an infinite allowable delay.

A.1.1 Minimum Delay

Under zero- or minimum-delay, data received from A is minimally stored on the ferry before forwarded on to B; this reflects the flow-pipe model of [17]. With instantaneous store-and-forward communication, throughput is limited by the minimum of the two communication rates. The maximum throughput in this case requires finding the tightest orbit Γ_0 of the ferrying aircraft that maximizes the minimum of the two rates:

$$\Gamma_0^{max} = \arg\max_{\Gamma} \ \min(R_A^\Gamma, R_B^\Gamma) \tag{A.1}$$

When there is no overlap in communication regions (in the event A and B are far apart), this limit will approach 0 as the allowable delay goes to zero.

A.1.2 Infinite Allowable Delay

Define the following trajectories that maximize communication between the ferry and A, and the ferry and B, respectively:

$$\Gamma_A^{max} = \arg\max_{\Gamma} R_A^{\Gamma} \tag{A.2a}$$

$$\Gamma_B^{max} = \arg\max_{\Gamma} R_B^{\Gamma} \tag{A.2b}$$

and R_A^{max} and R_B^{max} being the resulting effecitve communication rates. Under a long trajectory that takes a long time T to complete, the system is best served by having the ferry spend a fraction of time t_A along Γ_A^{max}, and another fraction t_B along Γ_B^{max}, with minimal transition time between the two [35, 18]. The equality constraint of the Link Scheduling Problem (Eq. (4.1)) reduces to

$$t_A R_A^{max} = t_B R_B^{max} \tag{A.3}$$

and as T grows,

$$t_A + t_B \to 1 \text{ as } T \to \inf . \tag{A.4}$$

Combined with the equality equation, allocation times can be solved as a ratio of the total time:

$$t_A R_A^{max} = (1 - t_A) R_B^{max} \tag{A.5a}$$

$$t_A (R_A^{max} + R_B^{max}) = R_B^{max} \tag{A.5b}$$

$$t_A = \frac{R_B^{max}}{(R_A^{max} + R_B^{max})} \tag{A.5c}$$

The resulting effective throughput from A to B is

$$t_A R_A^{max} = \frac{R_A^{max} R_B^{max}}{(R_A^{max} + R_B^{max})} \tag{A.6}$$

A.2 The Greedy Knapsack Heuristic is Arbitrarily Bad

This section explains how the greedy knapsack heuristic can be arbitrarily bad, as explained in [41], and then discusses how the setup of a typical ferry optimization problem significantly exceeds this bound. The Knapsack problem is stated as follows: we are given a set of n items j with profit p_j and weight w_j;

the objective is to select the best subset of items that maximizes profit subject to a total knapsack weight capacity c.

$$\text{maximize} \sum_{j=1}^{n} p_j x_j \tag{A.7a}$$

$$\text{subject to} \sum_{j=1}^{n} w_j x_j \leq c, \tag{A.7b}$$

$$x_j \in \{0,1\}, \quad j = 1, ..., n \tag{A.7c}$$

The greedy heuristic defines an item's efficiency as $e_j = p_j/w_j$, and then chooses items in order of most efficient to least until the capacity limit is reached. To show that this approach is arbitrarily bad, consider the instance where $n = 2$ and $c = M$ for some large value M. Item 1 has $w_1 = 1$ and $p_1 = 2$, while item 2 has $w_2 = p_2 = M$. In this instance, the efficiencies are $e_1 = 2$ and $e_2 = 1$. The greedy approach starts by packing item 1, but since item 2 would then exceed the capacity, the algorithm ends here with total profit 2. The optimal solution would be to pack item 2 for a profit of M. Thus for arbitrarily large M, the performance of the greedy algorithm can be forced arbitrarily close to 0.

Despite this arbitrarily bad performance bound, Chapter 4 showed extremely good performance. This happens for two reasons. First, the greedy approach can be shown to be optimal for the linear relaxation of the knapsack problem, where fractions of items can be selected; $x_j \in [0,1]$. Here, the optimal solution takes all of the most efficient items until the capacity limit is reached. The next item that would be packed is taken at a fraction to exactly reach the capacity. In the previous (worst-case) example, item 1 would be packed, and

$$\frac{w_2 - w_1}{w_2} = \frac{M-1)}{M}$$

would be chosen, for a profit of $2+(M-1)/M$ at the full capacity $c = M$. Second, the ferrying problem generally involves many segments, or items; and communication rates for good ferry trajectories tend to have a balanced set of achievable data rates, rather than having any single segment completely dominate the remaining subset of segments. This results in the trend seen in the segmentation issue - that as segments decrease in size and increase in number, the Greedy Knapsack solution approaches the linear relaxation's performance.

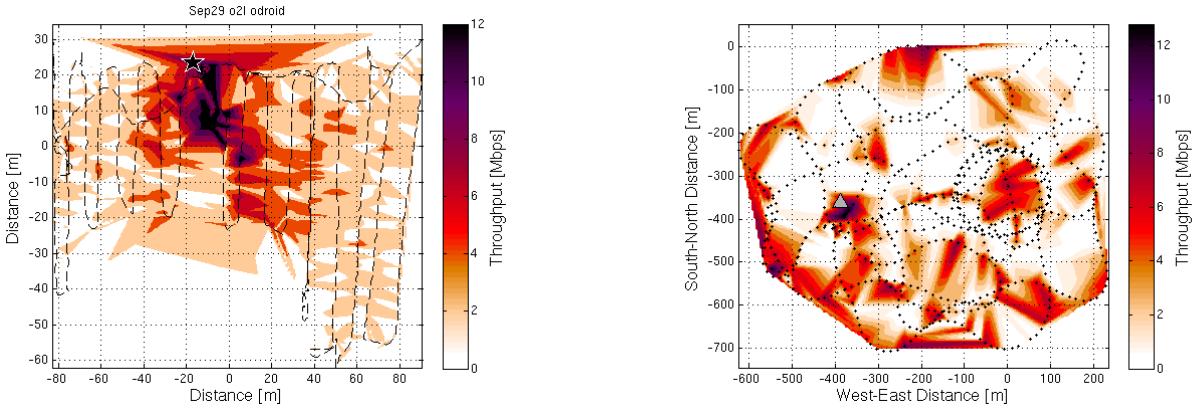
A.3 The Most Data First Heuristic is Arbitrarily Bad

Similar to the extreme knapsack case, the MDF algorithm is arbitrarily bad. Consider the instance of two segments where the data potentials from source node A are $d_a(1) = M$, $d_a(2) = 0$; and the potentials from destination B are $d_b(1) = N > M$ and $d_b(2) = M$. The optimal allocation in this case is to allocate segment 1 to A, 2 to B, resulting in M data collected and delivered.

The MDF algorithm begins by allocating a segment to B, choosing the one with the highest data potential: segment 1, potentially delivering N. Since the algorithm sees more potential data has been allocated to B than to A, it next picks a segment to allocate to A. Segment 2 is the only one remaining, which has a value of 0. This results in no data collected, and hence, no data delivered (despite the allocation to B).

A.4 Throughput Environments between Ground Node and Aircraft

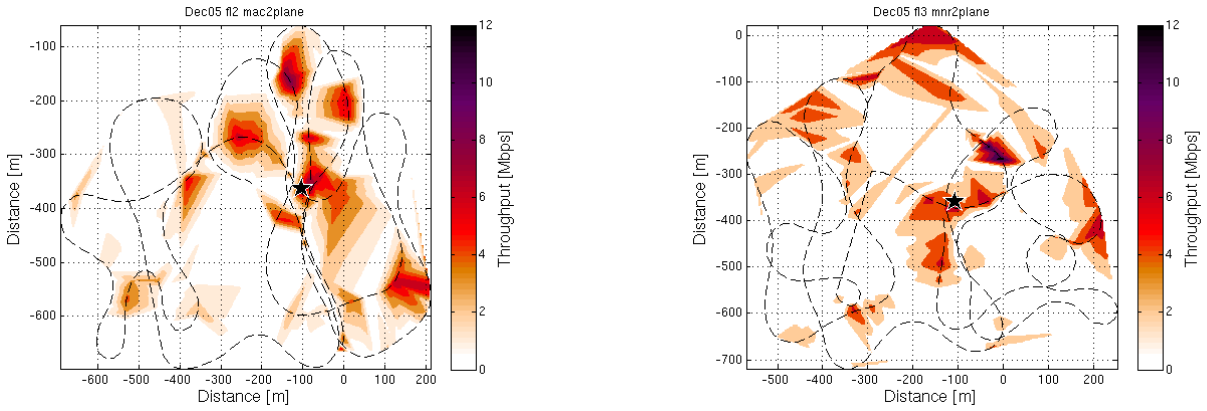
This section includes additional RF environments sampled throughout this research, showing a large spatial diversity in throughput characteristics. These throughput maps were gathered by having an unmanned aircraft fly a surveying route through an environment that included Wi-Fi-enabled ground nodes. Specifically, nodes would run Iperf in either a customized persistent-client mode or a server mode, and the aircraft would apply the other mode to test the Transport Layer capacity of the link.



(a) Odroid source node

(b)

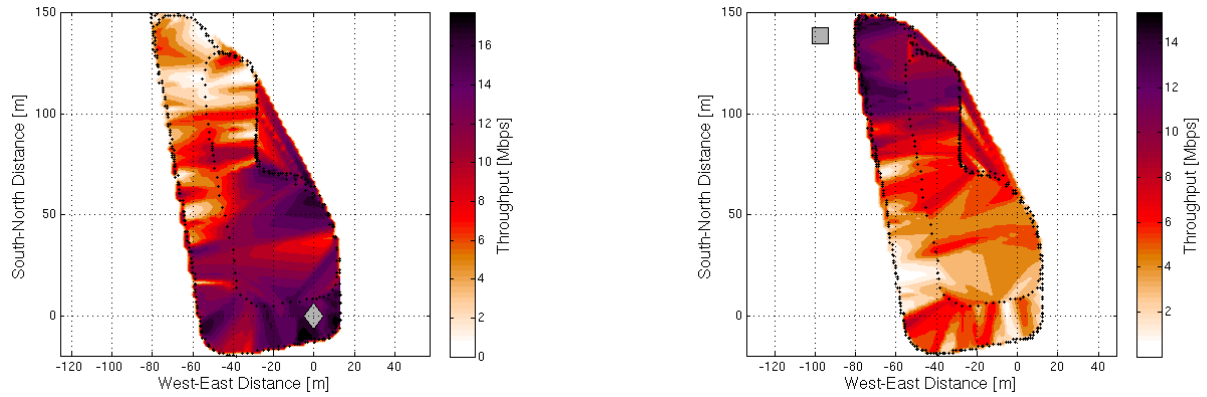
Figure A.1: Sept 29, 2014 Ad hoc environments



(a) Laptop source node

(b) MNR destination node

Figure A.2: Dec 5, 2014 Flight environments



(a) MNR source node

(b) Router destination node

Figure A.3: March 5, 2015 Flight environments