Information Theoretic Sensor Management

by

Jason L. Williams

B.E.(Electronics)(Hons.) B.Inf.Tech., Queensland University of Technology, 1999
M.S.E.E., Air Force Institute of Technology, 2003

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

February, 2007

Signature of Author: ____________________________________

Department of Electrical Engineering and Computer Science
January 12, 2007

Certified by: ____________________________________

John W. Fisher III
Principal Research Scientist, CSAIL
Thesis Supervisor

Certified by: ____________________________________

Alan S. Willsky
Edwin Sibley Webster Professor of Electrical Engineering
Thesis Supervisor

Accepted by: ____________________________________

Arthur C. Smith
Professor of Electrical Engineering
Chair, Committee for Graduate Students

Information Theoretic Sensor Management

by Jason L. Williams

Submitted to the Department of Electrical Engineering
and Computer Science on January 12, 2007
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

Sensor management may be defined as those stochastic control problems in which control values are selected to influence sensing parameters in order to maximize the utility of the resulting measurements for an underlying detection or estimation problem. While problems of this type can be formulated as a dynamic program, the state space of the program is in general infinite, and traditional solution techniques are inapplicable. Despite this fact, many authors have applied simple heuristics such as greedy or myopic controllers with great success.

This thesis studies sensor management problems in which information theoretic quantities such as entropy are utilized to measure detection or estimation performance. The work has two emphases: firstly, we seek performance bounds which guarantee performance of the greedy heuristic and derivatives thereof in certain classes of problems. Secondly, we seek to extend these basic heuristic controllers to find algorithms that provide improved performance and are applicable in larger classes of problems for which the performance bounds do not apply. The primary problem of interest is multiple object tracking and identification; application areas include sensor network management and multifunction radar control.

Utilizing the property of submodularity, as proposed for related problems by different authors, we show that the greedy heuristic applied to sequential selection problems with information theoretic objectives is guaranteed to achieve at least half of the optimal reward. Tighter guarantees are obtained for diffusive problems and for problems involving discounted rewards. Online computable guarantees also provide tighter bounds in specific problems. The basic result applies to open loop selections, where all decisions are made before any observation values are received; we also show that the closed loop greedy heuristic, which utilizes observations received in the interim in its subsequent decisions, possesses the same guarantee relative to the open loop optimal, and that no such guarantee exists relative to the optimal closed loop performance.

The same mathematical property is utilized to obtain an algorithm that exploits the structure of selection problems involving multiple independent objects. The algorithm involves a sequence of integer programs which provide progressively tighter upper bounds to the true optimal reward. An auxiliary problem provides progressively tighter lower bounds, which can be used to terminate when a near-optimal solution has been found. The formulation involves an abstract resource consumption model, which allows observations that expend different amounts of available time.

Finally, we present a heuristic approximation for an object tracking problem in a sensor network, which permits a direct trade-off between estimation performance and energy consumption. We approach the trade-off through a constrained optimization framework, seeking to either optimize estimation performance over a rolling horizon subject to a constraint on energy consumption, or to optimize energy consumption subject to a constraint on estimation performance. Lagrangian relaxation is used alongside a series of heuristic approximations to find a tractable solution that captures the essential structure in the problem.

Thesis Supervisors: John W. Fisher III[†] and Alan S. Willsky[‡]

Title: † Principal Research Scientist,
Computer Science and Artificial Intelligence Laboratory
‡ Edwin Sibley Webster Professor of Electrical Engineering

Acknowledgements

> We ought to give thanks for all fortune: if it is good, because it is good,
> if bad, because it works in us patience, humility and the contempt of this world
> and the hope of our eternal country.
>
> *C.S. Lewis*

It has been a wonderful privilege to have been able to study under and alongside such a tremendous group of people in this institution over the past few years. There are many people whom I must thank for making this opportunity the great experience that it has been. Firstly, I offer my sincerest thanks to my advisors, Dr John Fisher III and Prof Alan Willsky, whose support, counsel and encouragement has guided me through these years. The Army Research Office, the MIT Lincoln Laboratory Advanced Concepts Committee and the Air Force Office of Scientific Research all supported this research at various stages of development.

Thanks go to my committee members, Prof David Castañón (BU) and Prof Dimitri Bertsekas for offering their time and advice. Prof Castañón suggested applying column generation techniques to Section 4.1.2, which resulted in the development in Section 4.3. Various conversations with David Choi, Dan Rudoy and John Weatherwax (MIT Lincoln Laboratory) as well as Michael Schneider (BAE Systems Advanced Information Technologies) provided valuable input in the development of many of the formulations studied. Vikram Krishnamurthy (UBC) and David Choi first pointed me to the recent work applying submodularity to sensor management problems, which led to the results in Chapter 3.

My office mates, Pat Kreidl, Emily Fox and Kush Varshney, have been a constant sounding board for half-baked ideas over the years—I will certainly miss them on a professional level and on a personal level, not to mention my other lab mates in the Stochastic Systems Group. The members of the Eastgate Bible study, the Graduate Christian Fellowship and the Westgate community have been an invaluable source of friendship and support for both Jeanette and I; we will surely miss them as we leave Boston.

I owe my deepest gratitude to my wife, Jeanette, who has followed me around the world on this crazy journey, being an ever-present source of companionship, friendship and humour. I am extremely privileged to benefit from her unwavering love, support and encouragement. Thanks also go to my parents and extended family for their support as they have patiently awaited our return home.

Finally, to the God who creates and sustains, I humbly refer recognition for all success, growth and health with which I have been blessed over these years. The Lord gives and the Lord takes away; may the name of the Lord be praised.

Contents

List of Figures

Chapter 1

Introduction

DETECTION and estimation theory considers the problem of utilizing noise-corrupted observations to infer the state of some underlying process or phenomenon. Examples include detecting the presence of a heart disease using measurements from MRI, estimating ocean currents using image data from satellite, detecting and tracking people using video cameras, and tracking and identifying aircraft in the vicinity of an airport using radar.

Many modern sensors are able to rapidly change mode of operation and steer between physically separated objects. In many problem contexts, substantial performance gains can be obtained by exploiting this ability, adaptively controlling sensors to maximize the utility of the information received. Sensor management deals with such situations: where the objective is to maximize the utility of measurements for an underlying detection or estimation task.

Sensor management problems involving multiple time steps (in which decisions at a particular stage may utilize information received in all prior stages) can be formulated and, conceptually, solved using dynamic programming. However, in general the optimal solution of these problems requires computation and storage of continuous functions with no finite parameterization, hence it is intractable even problems involving small numbers of objects, sensors, control choices and time steps.

This thesis examines several types of sensor resource management problems. We follow three different approaches: firstly, we examine performance guarantees that can be obtained for simple heuristic algorithms applied to certain classes of problems; secondly, we exploit structure that arises in problems involving multiple independent objects to efficiently find optimal or guaranteed near-optimal solutions; and finally, we find a heuristic solution to a specific problem structure that arises in problems involving sensor networks.

■ 1.1 Canonical problem structures

Sensor resource management has received considerable attention from the research community over the past two decades. The following three canonical problem structures, which have been discussed by several authors, provide a rough classification of existing work, and of the problems examined in this thesis:

Waveform selection. The first problem structure involves a single object, which can be observed using different modes of a sensor, but only one mode can be used at a time. The role of the controller is to select the best mode of operation for the sensor in each time step. An example of this problem is in object tracking using radar, in which different signals can be transmitted in order to obtain information about different aspects of the object state (such as position, velocity or identity).

Beam steering. A related problem involves multiple objects observed by a sensor. Each object evolves according to an independent stochastic process. At each time step, the controller may choose which object to observe; the observation models corresponding to different objects are also independent. The role of the controller is to select which object to observe in each time step. An example of this problem is in optical tracking and identification using steerable cameras.

Platform steering. A third problem structure arises when the sensor possesses an internal state that affects which observations are available or the costs of obtaining those observations. The internal state evolves according to a fully observed Markov random process. The controller must choose actions to influence the sensor state such that the usefulness of the observations is optimized. Examples of this structure include control of UAV sensing platforms, and dynamic routing of measurements and models in sensor networks.

These three structures can be combined and extended to scenarios involving waveform selection, beam steering and platform steering with multiple objects and multiple sensors. An additional complication that commonly arises is when observations require different or random time durations (or, more abstractly, costs) to complete.

■ 1.2 Waveform selection and beam steering

The waveform selection problem naturally arises in many different application areas. Its name is derived from active radar and sonar, where the time/frequency characteristics of the transmitted waveform affect the type of information obtained in the return. Many other problems share the same structure, *i.e.*, one in which different control decisions obtain different types of information about the same underlying process. Other examples include:

- Passive sensors (*e.g.*, radar warning receivers) often have limited bandwidth, but can choose which interval of the frequency spectrum to observe at each time. Different choices will return information about different aspects of the phenomenon of interest. A similar example is the use of cameras with controllable pan and zoom to detect, track and identify people or cars.

- In ecological and geological applications, the phenomenon of interest is often comprised of the state of a large interconnected system. The dependencies within the system prevent the type of decomposition that is used in beam steering, and sensor resource management must be approached as a waveform selection problem involving different observations of the full system. Examples of problems of this type include monitoring of air quality, ocean temperature and depth mapping, and weather observation.

- Medical diagnosis concerns the determination of the true physiological state of a patient, which is evolving in time according to an underlying dynamical system. The practitioner has at their disposal a range of tests, each of which provides observations of different aspects of the phenomenon of interest. Associated with each test is a notion of cost, which encompasses time, patient discomfort, and economical considerations. The essential structure of this problem fits within the waveform selection category.

Beam steering may be seen to be a special case of the waveform selection problem. For example, consider the *hyper-object* that encompasses all objects being tracked. Choosing to observe different constituent objects will result in information relevant to different aspects of the hyper-object. Of course it is desirable to exploit the specific structure that exists in the case of beam steering.

Many authors have approached the waveform selection and beam steering problems by proposing an estimation performance measure, and optimizing the measure over the next time step. This approach is commonly referred to as *greedy* or *myopic*, since it does not consider future observation opportunities. Most of the non-myopic extensions of these methods are either tailored to very specific problem structure (observation models, dynamics models, *etc*), or are limited to considering two or three time intervals (longer planning horizons are typically computationally prohibitive). Furthermore, it is unclear when additional planning can be beneficial.

■ 1.3 Sensor networks

Networks of wireless sensors have the potential to provide unique capabilities for monitoring and surveillance due to the close range at which phenomena of interest can be observed. Application areas that have been investigated range from agriculture to ecological and geological monitoring to object tracking and identification. Sensor networks pose a particular challenge for resource management: not only are there short term resource constraints due to limited communication bandwidth, but there are also long term energy constraints due to battery limitations. This necessitates long term planning: for example, excessive energy should not be consumed in obtaining information that can be obtained a little later on at a much lower cost. Failure to do so will result in a reduced operational lifetime for the network.

It is commonly the case that the observations provided by sensors are highly informative if the sensor is in the close vicinity of the phenomenon of interest, and comparatively uninformative otherwise. In the context of object tracking, this has motivated the use of a dynamically assigned leader node, which determines which sensors should take and communicate observations, and stores and updates the knowledge of the object as new observations are obtained. The choice of leader node should naturally vary as the object moves through the network. The resulting structure falls within the framework of platform steering, where the sensor state is the currently activated leader node.

■ 1.4 Contributions and thesis outline

This thesis makes contributions in three areas. Firstly, we obtain performance guarantees that delineate problems in which additional planning is and is not beneficial. We then examine two problems in which long-term planning can be beneficial, finding an efficient integer programming solution that exploits the structure of beam steering, and finally, finding an efficient heuristic sensor management method for object tracking in sensor networks.

■ 1.4.1 Performance guarantees for greedy heuristics

Recent work has resulted in performance guarantees for greedy heuristics in some applications, but there remains no guarantee that is applicable to sequential problems without very special structure in the dynamics and observation models. The analysis in Chapter 3 obtains guarantees similar to the recent work in [46] for the sequential problem structures that commonly arise in waveform selection and beam steering. The result is quite general in that it applies to arbitrary, time varying dynamics and observation models. Several extensions are obtained, including tighter bounds that exploit either process diffusiveness or objectives involving discount factors, and applicability to closed loop problems. The results apply to objectives including mutual information, and the posterior Cramér-Rao bound. Examples demonstrate that the bounds are tight, and counterexamples illuminate larger classes of problems to which they do not apply.

■ 1.4.2 Efficient solution for beam steering problems

The analysis in Chapter 4 exploits the special structure in problems involving large numbers of independent objects to find an efficient solution of the beam steering problem. The analysis from Chapter 3 is utilized to obtain an upper bound on the objective function. Solutions with guaranteed near-optimality are found by simultaneously reducing the upper bound and raising a matching lower bound.

The algorithm has quite general applicability, admitting time varying observation and dynamical models, and observations requiring different time durations to complete. Computational experiments demonstrate application to problems involving 50–80 objects planning over horizons up to 60 time slots. An alternative formulation, which is able to address time invariant rewards with a further computational saving, is also discussed. The methods apply to the same wide range of objectives as Chapter 3, including mutual information and the posterior Cramér-Rao bound.

■ 1.4.3 Sensor network management

In Chapter 5, we seek to trade off estimation performance and energy consumed in an object tracking problem. We approach the trade off between these two quantities by maximizing estimation performance subject to a constraint on energy cost, or the dual of this, *i.e.*, minimizing energy cost subject to a constraint on estimation performance. We assign to each operation (sensing, communication, *etc*) an energy cost, and then we seek to develop a mechanism that allows us to choose only those actions for which the resulting estimation gain received outweighs the energy cost incurred. Our analysis proposes a planning method that is both computable and scalable, yet still captures the essential structure of the underlying trade off. Simulation results demonstrate a dramatic reduction in the communication cost required to achieve a given estimation performance level as compared to previously proposed algorithms.

Chapter 2

Background

THIS section provides an outline of the background theory which we utilize to develop our results. The primary problem of interest is that of detecting, tracking and identifying multiple objects, although many of the methods we discuss could be applied to any other dynamical process.

Sensor management requires an understanding of several related topics: first of all, one must develop a statistical model for the phenomenon of interest; then one must construct an estimator for conducting inference on that phenomenon. One must select an objective that measures how successful the sensor manager decisions have been, and, finally, one must design a controller to make decisions using the available inputs.

In Section 2.1, we briefly outline the development of statistical models for object tracking before describing some of the estimation schemes we utilize in our experiments. Section 2.2 outlines the theory of stochastic control, the category of problems in which sensor management naturally belongs. In Section 2.3, we describe the information theoretic objective functions that we utilize, and certain properties of the objectives that we utilize throughout the thesis. Section 2.4 details some existing results that have been applied to related problems to guarantee performance of simple heuristic algorithms; the focus of Chapter 3 is extending these results to sequential problems. Section 2.5 briefly outlines the theory of linear and integer programming that we utilize in Chapter 4. Finally, Section 2.6 surveys the existing work in the field, and contrasts the approaches presented in the later chapters to the existing methods.

■ 2.1 Dynamical models and estimation

This thesis will be concerned exclusively with sensor management systems that are based upon statistical models. The starting point of such algorithms is a dynamical model which captures mathematically how the physical process evolves, and how the observations taken by the sensor relate to the model variables. Having designed this model, one can then construct an estimator which uses sensor observations to refine one's knowledge of the state of the underlying physical process.

Using a Bayesian formulation, the estimator maintains a representation of the conditional probability density function (PDF) of the process state conditioned on the observations incorporated. This representation is central to the design of sensor management algorithms, which seek to choose sensor actions in order to minimize the uncertainty in the resulting estimate.

In this section, we outline the construction of dynamical models for object tracking, and then briefly examine several estimation methods that one may apply.

■ 2.1.1 Dynamical models

Traditionally, dynamical models for object tracking are based upon simple observations regarding behavior of targets and the laws of physics. For example, if we are tracking an aircraft and it moving at an essentially constant velocity at one instant in time, it will probably still be moving at a constant velocity shortly afterward. Accordingly, we may construct a mathematical model based upon Newtonian dynamics. One common model for non-maneuvering objects hypothesizes that velocity is a random walk, and position is the integral of velocity:

$$\dot{v}(t) = w(t) \tag{2.1}$$

$$\dot{p}(t) = v(t) \tag{2.2}$$

The process $w(t)$ is formally defined as a continuous time white noise with strength $Q(t)$. This strength may be chosen in order to model the expected deviation from the nominal trajectory.

In tracking, the underlying continuous time model is commonly chosen to be a stationary linear Gaussian system. Given any such model,

$$\dot{\boldsymbol{x}}(t) = \mathbf{F}_c\boldsymbol{x}(t) + \boldsymbol{w}(t) \tag{2.3}$$

where $\boldsymbol{w}(t)$ is a zero-mean Gaussian white noise process with strength $\mathbf{Q}_c$, we can construct a discrete-time model which has the equivalent effect at discrete sample points. This model is given by: [67]

$$\boldsymbol{x}_{k+1} = \mathbf{F}\boldsymbol{x}_k + \boldsymbol{w}_k \tag{2.4}$$

where[1]

$$\mathbf{F} = \exp[\mathbf{F}_c\Delta t] \tag{2.5}$$

[1] exp[·] denotes the matrix exponential.

Δt is the time difference between subsequent samples and $\boldsymbol{w}_k$ is a zero-mean discrete time Gaussian white noise process with covariance

$$\mathbf{Q} = \int_0^{\Delta t} \exp[\mathbf{F}_c \tau] \mathbf{Q}_c \exp[\mathbf{F}_c \tau]^T \mathrm{d}\tau \tag{2.6}$$

As an example, we consider tracking in two dimensions using the nominally constant velocity model described above:

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{x}(t) + \boldsymbol{w}(t) \tag{2.7}$$

where $\boldsymbol{x}(t) = [p_x(t)\ v_x(t)\ p_y(t)\ v_y(t)]^T$ and $\boldsymbol{w}(t) = [w_x(t)\ w_y(t)]^T$ is a continuous time zero-mean Gaussian white noise process with strength $\mathbf{Q}_c = q\mathbf{I}_{2\times 2}$. The equivalent discrete-time model becomes:

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}_k + \boldsymbol{w}_k \tag{2.8}$$

where $\boldsymbol{w}_k$ is a discrete time zero-mean Gaussian white noise process with covariance

$$\mathbf{Q} = q \begin{bmatrix} \frac{\Delta t^3}{3} & \frac{\Delta t^2}{2} & 0 & 0 \\ \frac{\Delta t^2}{2} & \Delta t & 0 & 0 \\ 0 & 0 & \frac{\Delta t^3}{3} & \frac{\Delta t^2}{2} \\ 0 & 0 & \frac{\Delta t^2}{2} & \Delta t \end{bmatrix} \tag{2.9}$$

Objects undergoing frequent maneuvers are commonly modelled using jump Markov linear systems. In this case, the dynamical model at any time is a linear system, but the parameters of the linear system change at discrete time instants; these changes are modelled through a finite state Markov chain. While not explicitly explored in this thesis, the jump Markov linear system can be addressed by the methods and guarantees we develop. We refer the reader to [6] for further details of estimation using jump Markov linear systems.

■ 2.1.2 Kalman filter

The Kalman filter is the optimal estimator according to most sensible criteria, including mean square error, mean absolute error and uniform cost, for a linear dynamical system with additive white Gaussian noise and linear observations with additive white Gaussian noise. If we relax the Gaussianity requirement on the noise processes, the Kalman filter remains the optimal linear estimator according to the mean square error criterion. We briefly outline the Kalman filter below; the reader is referred to [3, 6, 27, 67] for more in-depth treatments.

We consider the discrete time linear dynamical system:

$$\boldsymbol{x}_{k+1} = \mathbf{F}_k \boldsymbol{x}_k + \boldsymbol{w}_k \tag{2.10}$$

commencing from $\boldsymbol{x}_0 \sim \mathcal{N}\{\boldsymbol{x}_0; \hat{\boldsymbol{x}}_{0|0}, \mathbf{P}_{0|0}\}$. The dynamics noise w_k is a the white noise process, $\boldsymbol{w}_k \sim \mathcal{N}\{\boldsymbol{w}_k; \mathbf{0}, \mathbf{Q}_k\}$ which is uncorrelated with $\boldsymbol{x}_0$. We assume a linear observation model:

$$\boldsymbol{z}_k = \mathbf{H}_k \boldsymbol{x}_k + \boldsymbol{v}_k \tag{2.11}$$

where $\boldsymbol{v}_k \sim \mathcal{N}\{\boldsymbol{v}_k; \mathbf{0}, \mathbf{R}_k\}$ is a white noise process that is uncorrelated with $\boldsymbol{x}_0$ and with the process $\boldsymbol{v}_k$. The Kalman filter equations include a propagation step:

$$\hat{\boldsymbol{x}}_{k|k-1} = \mathbf{F}_k \hat{\boldsymbol{x}}_{k-1|k-1} \tag{2.12}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k \tag{2.13}$$

and an update step:

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \mathbf{K}_k [\boldsymbol{z}_k - \mathbf{H}_k \hat{\boldsymbol{x}}_{k|k-1}] \tag{2.14}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_{k|k-1} \tag{2.15}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T [\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k]^{-1} \tag{2.16}$$

$\hat{\boldsymbol{x}}_{k|k-1}$ is the estimate of $\boldsymbol{x}_k$ conditioned on observations up to and including time $(k-1)$, while $\mathbf{P}_{k|k-1}$ is the covariance of this error. In the Gaussian case, these two parameters completely describe the posterior distribution, *i.e.*, $p(\boldsymbol{x}_k|\boldsymbol{z}_{0:k-1}) = \mathcal{N}\{\boldsymbol{x}_k; \hat{\boldsymbol{x}}_{k|k-1}, \mathbf{P}_{k|k-1}\}$. Similar comments apply to $\hat{\boldsymbol{x}}_{k|k}$ and $\mathbf{P}_{k|k}$.

Finally, we note that the recursive equations for the covariance $\mathbf{P}_{k|k-1}$ and $\mathbf{P}_{k|k}$, and the gain $\mathbf{K}_k$ are both invariant to the value of the observations received $\boldsymbol{z}_k$. Accordingly, both the filter gain and covariance may be computed offline in advance and stored. As we will see in Section 2.3, in the linear Gaussian case, the uncertainty in an estimate as measured through entropy is dependent only upon the covariance matrix, and hence this too can be calculated offline.

■ 2.1.3 Linearized and extended Kalman filter

While the optimality guarantees for the Kalman filter apply only to linear systems, the basic concept is regularly applied to nonlinear systems through two algorithms known as the linearized and extended Kalman filters. The basic concept is that a mild nonlinearity may be approximated as being linear about a nominal point though a Taylor series expansion. In the case of the linearized Kalman filter, the linearization point is chosen in advance; the extended Kalman filter relinearizes online about the current estimate value. Consequently, the linearized Kalman filter retains the ability to calculate filter gains and covariance matrices in advance, whereas the extended Kalman filter must compute both of these online.

In this document, we assume that the dynamical model is linear, and we present the equations for the linearized and extended Kalman filters for the case in which the only nonlinearity present is in the observation equation. This is most commonly the case in tracking applications. The reader is directed to [68], the primary source for this material, for information on the nonlinear dynamical model case. The model we consider is:

$$\boldsymbol{x}_{k+1} = \mathbf{F}_k \boldsymbol{x}_k + \boldsymbol{w}_k \tag{2.17}$$

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k, k) + \boldsymbol{v}_k \tag{2.18}$$

where, as in Section 2.1.2, $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ are uncorrelated white Gaussian noise processes with known covariance, both of which are uncorrelated with $\boldsymbol{x}_0$. The linearized Kalman filter calculates a linearized measurement model about a pre-specified nominal state trajectory $\{\bar{\boldsymbol{x}}_k\}_{k=1,2,\dots}$:

$$\boldsymbol{z}_k \approx \boldsymbol{h}(\bar{\boldsymbol{x}}_k, k) + \mathbf{H}(\bar{\boldsymbol{x}}_k, k)[\boldsymbol{x}_k - \bar{\boldsymbol{x}}_k] + \boldsymbol{v}_k \tag{2.19}$$

where

$$\mathbf{H}(\bar{\boldsymbol{x}}_k, k) = [\boldsymbol{\nabla}_{\boldsymbol{x}} \boldsymbol{h}(\boldsymbol{x}, k)^T]^T|_{\boldsymbol{x}=\bar{\boldsymbol{x}}_k} \tag{2.20}$$

and $\boldsymbol{\nabla}_{\boldsymbol{x}} \triangleq [\frac{\partial}{\partial \boldsymbol{x}_1}, \frac{\partial}{\partial \boldsymbol{x}_2}, \dots, \frac{\partial}{\partial \boldsymbol{x}_{n_{\boldsymbol{x}}}}]^T$ where $n_{\boldsymbol{x}}$ is the number of elements in the vector $\boldsymbol{x}$. The linearized Kalman filter update equation is therefore:

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \mathbf{K}_k\{\boldsymbol{z}_k - \boldsymbol{h}(\bar{\boldsymbol{x}}_k, k) - \mathbf{H}(\bar{\boldsymbol{x}}_k, k)[\boldsymbol{x}_k - \bar{\boldsymbol{x}}_k]\} \tag{2.21}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{H}(\bar{\boldsymbol{x}}_k, k)\mathbf{P}_{k|k-1} \tag{2.22}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}(\bar{\boldsymbol{x}}_k, k)^T[\mathbf{H}(\bar{\boldsymbol{x}}_k, k)\mathbf{P}_{k|k-1}\mathbf{H}(\bar{\boldsymbol{x}}_k, k)^T + \mathbf{R}_k]^{-1} \tag{2.23}$$

Again we note that the filter gain and covariance are both invariant to the observation values, and hence they can be precomputed.

The extended Kalman filter differs only in the in the point about which the model is linearized. In this case, we linearize about the current state estimate:

$$\boldsymbol{z}_k \approx \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}, k) + \mathbf{H}(\hat{\boldsymbol{x}}_{k|k-1}, k)[\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1}] + \boldsymbol{v}_k \tag{2.24}$$

The extended Kalman filter update equation becomes:

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \mathbf{K}_k[\boldsymbol{z}_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}, k)] \tag{2.25}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{H}(\hat{\boldsymbol{x}}_{k|k-1}, k)\mathbf{P}_{k|k-1} \tag{2.26}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}(\hat{\boldsymbol{x}}_{k|k-1}, k)^T[\mathbf{H}(\hat{\boldsymbol{x}}_{k|k-1}, k)\mathbf{P}_{k|k-1}\mathbf{H}(\hat{\boldsymbol{x}}_{k|k-1}, k)^T + \mathbf{R}_k]^{-1} \tag{2.27}$$

Since the filter gain and covariance are dependent on the state estimate and hence the previous observation values, the extended Kalman filter must be computed online.

2.1.4 Particle filters and importance sampling

In many applications, substantial nonlinearity is encountered in observation models, and the coarse approximation performed by the extended Kalman filter is inadequate. This is particularly true in sensor networks, since the local focus of observations yields much greater nonlinearity in range or bearing observations than arises when sensors are distant from the objects under surveillance. This nonlinearity can result in substantial multimodality in posterior distributions (such as results when one receives two range observations from sensors in different locations) which cannot be efficiently modelled using a Gaussian distribution. We again assume a linear dynamical model (although this is by no means required) and a nonlinear observation model:

$$\boldsymbol{x}_{k+1} = \mathbf{F}_k\boldsymbol{x}_k + \boldsymbol{w}_k \tag{2.28}$$

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k, k) + \boldsymbol{v}_k \tag{2.29}$$

We apply the same assumptions on $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ as in previous sections.

The particle filter [4, 28, 83] is an approximation which is commonly used in problems involving a high degree of nonlinearity and/or non-Gaussianity. The method is based on importance sampling, which enables one to approximate an expectation under one distribution using samples drawn from another distribution. Using a particle filter, the conditional PDF of object state $\boldsymbol{x}_k$ conditioned on observations received up to and including time k, $\boldsymbol{z}_{0:k}$, $p(\boldsymbol{x}_k|\boldsymbol{z}_{0:k})$, is approximated through a set of N_p weighted samples:

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{0:k}) \approx \sum_{i=1}^{N_p} w_k^i \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^i) \tag{2.30}$$

Several variants of the particle filter differ in the way in which this approximated is propagated and updated from step to step. Perhaps the most common (and the easiest to implement) is the Sampling Importance Resampling (SIR) filter. This algorithm approximates the propagation step by using the dynamics model as a proposal distribution, drawing a random sample for each particle from the distribution $\boldsymbol{x}_{k+1}^i \sim p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k^i)$, to yield an approximation of the prior density at the next time step of:

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{z}_{0:k}) \approx \sum_{i=1}^{N_p} w_k^i \delta(\boldsymbol{x}_{k+1} - \boldsymbol{x}_{k+1}^i) \tag{2.31}$$

The algorithm then uses importance sampling to reweight these samples to implement the Bayes update rule for incorporating observations:

$$\begin{align} p(\boldsymbol{x}_{k+1}|\boldsymbol{z}_{0:k+1}) &= \frac{p(\boldsymbol{z}_{k+1}|\boldsymbol{x}_{k+1})p(\boldsymbol{x}_{k+1}|\boldsymbol{z}_{0:k})}{p(\boldsymbol{z}_{k+1}|\boldsymbol{z}_{0:k})} \tag{2.32} \\ &= \frac{\sum_{i=1}^{N_p} w_k^i p(\boldsymbol{z}_{k+1}|\boldsymbol{x}_{k+1}^i)\delta(\boldsymbol{x}_{k+1} - \boldsymbol{x}_{k+1}^i)}{p(\boldsymbol{z}_{k+1}|\boldsymbol{z}_{0:k})} \tag{2.33} \\ &= \sum_{i=1}^{N_p} w_{k+1}^i \delta(\boldsymbol{x}_{k+1} - \boldsymbol{x}_{k+1}^i) \tag{2.34} \end{align}$$

where

$$w_{k+1}^i = \frac{w_k^i p(\boldsymbol{z}_{k+1}|\boldsymbol{x}_{k+1}^i)}{\sum_{j=1}^{N_p} w_k^j p(\boldsymbol{z}_{k+1}|\boldsymbol{x}_{k+1}^j)} \tag{2.35}$$

The final step of the SIR filter is to draw a new set of N_p samples from the updated distribution to reduce the number of samples allocated to unlikely regions and reinitialize the weights to be uniform.

A more sophisticated variant of the particle filter is the Sequential Importance Sampling (SIS) algorithm. Under this algorithm, for each previous sample $\boldsymbol{x}_k^i$, we draw a new sample at the next time step, $\boldsymbol{x}_{k+1}$, from the proposal distribution $q(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k^i, \boldsymbol{z}_{k+1})$. This is commonly approximated using a linearization of the measurement model for $\boldsymbol{z}_{k+1}$ (Eq. (2.29)) about the point $\mathbf{F}_k\boldsymbol{x}_k^i$, as described in Eq. (2.19). This distribution can be obtained using the extended Kalman filter equations: the Dirac delta function $\delta(\boldsymbol{x}_k - \boldsymbol{x}_k^i)$ at time k will diffuse to give:

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k^i) = \mathcal{N}(\boldsymbol{x}_{k+1}; \mathbf{F}_k\boldsymbol{x}_k^i; \mathbf{Q}_k) \tag{2.36}$$

at time $(k+1)$. This distribution can be updated using the EKF update equation (Eq. (2.25)–(2.27)) to obtain:

$$q(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k^i, \boldsymbol{z}_{k+1}) = \mathcal{N}(\boldsymbol{x}_{k+1}; \hat{\boldsymbol{x}}_{k+1}^i, \mathbf{P}_{k+1}^i) \tag{2.37}$$

where

$$\hat{\boldsymbol{x}}_{k+1}^i = \mathbf{F}_k\boldsymbol{x}_k^i + \mathbf{K}_{k+1}^i[\boldsymbol{z}_{k+1} - \boldsymbol{h}(\mathbf{F}_k\boldsymbol{x}_k^i, k)] \tag{2.38}$$

$$\mathbf{P}_{k+1}^i = \mathbf{Q}_k - \mathbf{K}_{k+1}^i\mathbf{H}(\mathbf{F}_k\boldsymbol{x}_k^i, k)\mathbf{Q}_k \tag{2.39}$$

$$\mathbf{K}_{k+1}^i = \mathbf{Q}_k\mathbf{H}(\mathbf{F}_k\boldsymbol{x}_k^i, k)^T[\mathbf{H}(\mathbf{F}\boldsymbol{x}_k^i, k)\mathbf{Q}_k\mathbf{H}(\mathbf{F}\boldsymbol{x}_k^i, k)^T + \mathbf{R}_k]^{-1} \tag{2.40}$$

Because the linearization is operating in a localized region, one can obtain greater accuracy than is possible using the EKF (which uses a single linearization point). A new particle $\boldsymbol{x}_{k+1}^i$ is drawn from the distribution in Eq. (2.37), and the importance sampling weight w_{k+1}^i is calculated by

$$w_{k+1}^i = cw_k^i\frac{p(\boldsymbol{z}_{k+1}|\boldsymbol{x}_{k+1}^i)p(\boldsymbol{x}_{k+1}^i|\boldsymbol{x}_k^i)}{q(\boldsymbol{x}_{k+1}^i|\boldsymbol{x}_k^i, \boldsymbol{z}_{k+1})} \tag{2.41}$$

where c is the normalization constant necessary to ensure that $\sum_{i=1}^{N_p} w_{k+1}^i = 1$, and $p(\boldsymbol{z}_{k+1}|\boldsymbol{x}_{k+1}^i) = \mathcal{N}\{\boldsymbol{z}_{k+1}; \boldsymbol{h}(\boldsymbol{x}_{k+1}^i, k), \mathbf{R}_k\}$. The resulting approximation for the distribution of $\boldsymbol{x}_{k+1}$ conditioned on the measurements $\boldsymbol{z}_{0:k+1}$ is:

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{z}_{0:k+1}) \approx \sum_{i=1}^{N_p} w_{k+1}^i\delta(\boldsymbol{x}_{k+1} - \boldsymbol{x}_{k+1}^i) \tag{2.42}$$

At any point in time, a Gaussian representation can be moment-matched to the particle distribution by calculating the mean and covariance:

$$\hat{\boldsymbol{x}}_k = \sum_{i=1}^{N_p} w_k^i\boldsymbol{x}_k^i \tag{2.43}$$

$$\mathbf{P}_k = \sum_{i=1}^{N_p} w_k^i(\boldsymbol{x}_k^i - \hat{\boldsymbol{x}}_k)(\boldsymbol{x}_k^i - \hat{\boldsymbol{x}}_k)^T \tag{2.44}$$

■ 2.1.5 Graphical models

In general, the complexity of an estimation problem increases exponentially as the number of variables increases. Probabilistic graphical models provide a framework for recognizing and exploiting structure which allows for efficient solution. Here we briefly describe Markov random fields, a variety of undirected graphical model. Further details can be found in [38, 73, 92].

We assume that our model is represented as as graph $\mathcal{G}$ consisting of vertices $\mathcal{V}$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Corresponding to each vertex $v \in \mathcal{V}$ is a random variable x_v and several possible observations (from which we may choose some subset) of that variable, $\{z_v^1, \dots, z_v^{n_v}\}$. Edges represent dependences between the local random variables, *i.e.*, $(v, w) \in \mathcal{E}$ denotes that variables x_v and x_w have direct dependence on each other. All observations are assumed to depend only on the corresponding local random variable.

In this case, the joint distribution function can be shown to factorize over the maximal cliques of the graph. A clique is defined as a set of vertices $\mathcal{C} \subseteq \mathcal{V}$ which are fully connected, *i.e.*, $(v, w) \in \mathcal{E}\ \forall\ v, w \in \mathcal{C}$. A maximal clique is a clique which is not a subset of any other clique in the graph (*i.e.*, a clique for which no other vertex can be added while still retaining full connectivity). Denoting the collection of all maximal cliques as $\mathscr{M}$, the joint distribution of variables and observations can be written as:

$$p(\{x_v, \{z_v^1, \dots, z_v^{n_v}\}\}_{v \in \mathcal{V}}) \propto \prod_{\mathcal{C} \in \mathscr{M}} \psi(\{x_v\}_{v \in \mathcal{C}}) \prod_{v \in \mathcal{V}} \prod_{i=1}^{n_v} \psi(x_v, z_v^i) \tag{2.45}$$

Graphical models are useful in recognizing independence structures which exist. For example, two random variables x_v and x_w $(v, w \in \mathcal{V})$ are independent conditioned on a given set of vertices $\mathcal{D}$ if there is no path connecting vertices v and w which does not pass through any vertex in $\mathcal{D}$. Obviously, if we denote by $\mathcal{N}(v)$ the neighbors of vertex v, then x_v will be independent of all other variables in the graph conditioned on $\mathcal{N}(v)$.

Estimation problems involving undirected graphical models with a tree as the graph structure can be solved efficiently using the belief propagation algorithm. Some problems involving sparse cyclic graphs can be addressed efficiently by combining small numbers of nodes to obtain tree structure (referred to as a junction tree), but in general approximate methods, such as loopy belief propagation, are necessary. Estimation in time series is a classical example of a tree-based model: the Kalman filter (or, more precisely, the Kalman smoother) may be seen to be equivalent to belief propagation specialized to linear Gaussian Markov chains, while [35, 88, 89] extends particle filtering from Markov chains to general graphical models using belief propagation.

■ 2.1.6 Cramér-Rao bound

The Cramér-Rao bound (CRB) [84, 91] provides a lower limit on the mean square error performance achievable by any estimator of an underlying quantity. The simplest and most common form of the bound, presented below in Theorem 2.1, deals with unbiased estimates of nonrandom parameters (*i.e.*, parameters which are not endowed with a prior probability distribution). We omit the various regularity conditions; see [84, 91] for details. The notation $\mathbf{A} \succeq \mathbf{B}$ implies that the matrix $\mathbf{A} - \mathbf{B}$ is positive semi-definite (PSD). We adopt convention from [90] that $\boldsymbol{\Delta}_{\boldsymbol{x}}^{\boldsymbol{z}} = \boldsymbol{\nabla}_{\boldsymbol{x}} \boldsymbol{\nabla}_{\boldsymbol{z}}^T$.

Theorem 2.1. *Let $\boldsymbol{x}$ be a nonrandom vector parameter, and $\boldsymbol{z}$ be an observation with distribution $p(\boldsymbol{z}|\boldsymbol{x})$ parameterized by $\boldsymbol{x}$. Then any unbiased estimator of $\boldsymbol{x}$ based on $\boldsymbol{z}$, $\hat{\boldsymbol{x}}(\boldsymbol{z})$, must satisfy the following bound on covariance:*

$$\mathop{\mathrm{E}}_{\boldsymbol{z}|\boldsymbol{x}} \left\{ [\hat{\boldsymbol{x}}(\boldsymbol{z}) - \boldsymbol{x}][\hat{\boldsymbol{x}}(\boldsymbol{z}) - \boldsymbol{x}]^T \right\} \succeq \mathbf{C}_{\boldsymbol{x}}^{\boldsymbol{z}} \triangleq [\mathbf{J}_{\boldsymbol{x}}^{\boldsymbol{z}}]^{-1}$$

where $\mathbf{J}_{\boldsymbol{x}}^{\boldsymbol{z}}$ is the Fisher information matrix, which can be calculated equivalently through either of the following two forms:

$$\begin{aligned} \mathbf{J}_{\boldsymbol{x}}^{\boldsymbol{z}} &\triangleq \mathop{\mathrm{E}}_{\boldsymbol{z}|\boldsymbol{x}} \left\{ [\boldsymbol{\nabla}_{\boldsymbol{x}} \log p(\boldsymbol{z}|\boldsymbol{x})][\boldsymbol{\nabla}_{\boldsymbol{x}} \log p(\boldsymbol{z}|\boldsymbol{x})]^T \right\} \\ &= \mathop{\mathrm{E}}_{\boldsymbol{z}|\boldsymbol{x}} \{ -\boldsymbol{\Delta}_{\boldsymbol{x}}^{\boldsymbol{x}} \log p(\boldsymbol{z}|\boldsymbol{x}) \} \end{aligned}$$

From the first form above we see that the Fisher information matrix is positive semi-definite.

The posterior Cramér-Rao bound (PCRB) [91] provides a similar performance limit for dealing with random parameters. While the bound takes on the same form, the Fisher information matrix now decomposes into two terms: one involving prior information about the parameter, and another involving information gained from the observation. Because we take an expectation over the possible values of $\boldsymbol{x}$ as well as $\boldsymbol{z}$, the bound applies to any estimator, biased or unbiased.

Theorem 2.2. *Let $\boldsymbol{x}$ be a random vector parameter with probability distribution $p(\boldsymbol{x})$, and $\boldsymbol{z}$ be an observation with model $p(\boldsymbol{z}|\boldsymbol{x})$. Then any estimator of $\boldsymbol{x}$ based on $\boldsymbol{z}$, $\hat{\boldsymbol{x}}(\boldsymbol{z})$, must satisfy the following bound on covariance:*

$$\mathop{\mathrm{E}}_{\boldsymbol{x},\boldsymbol{z}} \left\{ [\hat{\boldsymbol{x}}(\boldsymbol{z}) - \boldsymbol{x}][\hat{\boldsymbol{x}}(\boldsymbol{z}) - \boldsymbol{x}]^T \right\} \succeq \mathbf{C}_{\boldsymbol{x}}^{\boldsymbol{z}} \triangleq [\mathbf{J}_{\boldsymbol{x}}^{\boldsymbol{z}}]^{-1}$$

where $\mathbf{J}_{\boldsymbol{x}}^{\boldsymbol{z}}$ is the Fisher information matrix, which can be calculated equivalently through *any of the following forms:*

$$\begin{aligned}
\mathbf{J}_{\boldsymbol{x}}^{\boldsymbol{z}} &\triangleq \mathop{\mathrm{E}}_{\boldsymbol{x},\boldsymbol{z}} \left\{ [\boldsymbol{\nabla}_{\boldsymbol{x}} \log p(\boldsymbol{x},\boldsymbol{z})][\boldsymbol{\nabla}_{\boldsymbol{x}} \log p(\boldsymbol{x},\boldsymbol{z})]^T \right\} \\
&= \mathop{\mathrm{E}}_{\boldsymbol{x},\boldsymbol{z}} \{ -\boldsymbol{\Delta}_{\boldsymbol{x}}^{\boldsymbol{x}} \log p(\boldsymbol{x},\boldsymbol{z}) \} \\
&= \mathop{\mathrm{E}}_{\boldsymbol{x}} \left\{ [\boldsymbol{\nabla}_{\boldsymbol{x}} \log p(\boldsymbol{x})][\boldsymbol{\nabla}_{\boldsymbol{x}} \log p(\boldsymbol{x})]^T \right\} + \mathop{\mathrm{E}}_{\boldsymbol{x},\boldsymbol{z}} \left\{ [\boldsymbol{\nabla}_{\boldsymbol{x}} \log p(\boldsymbol{z}|\boldsymbol{x})][\boldsymbol{\nabla}_{\boldsymbol{x}} \log p(\boldsymbol{z}|\boldsymbol{x})]^T \right\} \\
&= \mathop{\mathrm{E}}_{\boldsymbol{x}} \{ -\boldsymbol{\Delta}_{\boldsymbol{x}}^{\boldsymbol{x}} \log p(\boldsymbol{x}) \} + \mathop{\mathrm{E}}_{\boldsymbol{x},\boldsymbol{z}} \{ -\boldsymbol{\Delta}_{\boldsymbol{x}}^{\boldsymbol{x}} \log p(\boldsymbol{z}|\boldsymbol{x}) \} \\
&= \mathop{\mathrm{E}}_{\boldsymbol{x},\boldsymbol{z}} \left\{ [\boldsymbol{\nabla}_{\boldsymbol{x}} \log p(\boldsymbol{x}|\boldsymbol{z})][\boldsymbol{\nabla}_{\boldsymbol{x}} \log p(\boldsymbol{x}|\boldsymbol{z})]^T \right\} \\
&= \mathop{\mathrm{E}}_{\boldsymbol{x},\boldsymbol{z}} \{ -\boldsymbol{\Delta}_{\boldsymbol{x}}^{\boldsymbol{x}} \log p(\boldsymbol{x}|\boldsymbol{z}) \}
\end{aligned}$$

The individual terms:

$$\begin{aligned}
\mathbf{J}_{\boldsymbol{x}}^{\emptyset} &\triangleq \mathop{\mathrm{E}}_{\boldsymbol{x}} \{ -\boldsymbol{\Delta}_{\boldsymbol{x}}^{\boldsymbol{x}} \log p(\boldsymbol{x}) \} \\
\bar{\mathbf{J}}_{\boldsymbol{x}}^{\boldsymbol{z}} &\triangleq \mathop{\mathrm{E}}_{\boldsymbol{x},\boldsymbol{z}} \{ -\boldsymbol{\Delta}_{\boldsymbol{x}}^{\boldsymbol{x}} \log p(\boldsymbol{z}|\boldsymbol{x}) \}
\end{aligned}$$

are both positive semi-definite. We also define $\mathbf{C}_{\boldsymbol{x}}^{\emptyset} \triangleq [\mathbf{J}_{\boldsymbol{x}}^{\emptyset}]^{-1}$.

Convenient expressions for calculation of the PCRB in nonlinear filtering problems can be found in [90]. The recursive expressions are similar in form to the Kalman filter equations.

■ 2.2 Markov decision processes

Markov Decision Processes (MDPs) provide a natural way of formulating problems involving sequential structure, in which decisions are made incrementally as additional information is received. We will concern ourselves primarily with problems involving planning over a finite number of steps (so-called finite horizon problems); in practice we will design our controller by selecting an action for the current time considering the following N time steps (referred to as rolling horizon or receding horizon control). The basic problem formulation includes:

State. We denote by $\mathbb{X}_k \in \mathcal{X}$ the decision state of the system at time k. The decision state is a sufficient statistic for all past and present information upon which the controller can make its decisions. The sufficient statistic must be chosen such that future values are independent of past values conditioned on the present value (*i.e.*, it must form a Markov process).

Control. We denote by $u_k \in \mathcal{U}_k^{\mathbb{X}_k}$ the control to be applied to the system at time k. $\mathcal{U}_k^{\mathbb{X}_k} \subseteq \mathcal{U}$ is the set of controls available at time k if the system is in state $\mathbb{X}_k$. In some problem formulations this set will vary with time and state; in others it will remain constant.

Transition. If the state at time k is $\mathbb{X}_k$ and control u_k is applied, then the state at time $(k+1)$, $\mathbb{X}_{k+1}$, will be distributed according to the probability measure $P(\cdot|\mathbb{X}_k; u_k)$.

Reward. The objective of the system is specified as a reward (or cost) to be maximized (or minimized). This consists of two components: the per-stage reward $g_k(\mathbb{X}_k, u_k)$, which is the immediate reward if control u_k is applied at time k from state $\mathbb{X}_k$, and the terminal reward $g_N(\mathbb{X}_N)$, which is the reward associated with arriving in state $\mathbb{X}_N$ on completion of the problem.

The solution of problems with this structure comes in the form of a policy, *i.e.*, a rule that specifies which control one should apply if one arrives in a particular state at a particular time. We denote by $\mu_k : \mathcal{X} \to \mathcal{U}$ the policy for time k, and by $\pi = \{\mu_1, \dots, \mu_N\}$ the time-varying policy for the finite horizon problem. The expected reward to go of a given policy can be found through the following backward recursion:

$$J_k^\pi(\mathbb{X}_k) = g_k(\mathbb{X}_k, \mu_k(\mathbb{X}_k)) + \underset{\mathbb{X}_{k+1} \sim P(\cdot|\mathbb{X}_k, \mu_k(\mathbb{X}_k))}{\mathrm{E}} J_{k+1}^\pi(\mathbb{X}_{k+1}) \tag{2.46}$$

commencing from the terminal condition $J_N^\pi(\mathbb{X}_N) = g_N(\mathbb{X}_N)$. The expected reward to go of the optimal policy can be formulated similarly as a backward recursion:

$$J_k^*(\mathbb{X}_k) = \max_{u_k \in \mathcal{U}_k^{\mathbb{X}_k}} \left\{ g_k(\mathbb{X}_k, u_k) + \underset{\mathbb{X}_{k+1} \sim P(\cdot|\mathbb{X}_k, u_k)}{\mathrm{E}} J_{k+1}^*(\mathbb{X}_{k+1}) \right\} \tag{2.47}$$

commencing from the same terminal condition, $J_N^*(\mathbb{X}_N) = g_N(\mathbb{X}_N)$. The optimal policy is implicity specified by the optimal reward to go through the expression

$$\mu_k^*(\mathbb{X}_k) = \arg \max_{u_k \in \mathcal{U}_k^{\mathbb{X}_k}} \left\{ g_k(\mathbb{X}_k, u_k) + \underset{\mathbb{X}_{k+1} \sim P(\cdot|\mathbb{X}_k, u_k)}{\mathrm{E}} J_{k+1}^*(\mathbb{X}_{k+1}) \right\} \tag{2.48}$$

The expression in Eq. (2.46) can be used to evaluate the expected reward of a policy when the cardinality of the state space $\mathcal{X}$ is small enough to allow computation and storage for each element. Furthermore, assuming that the optimization is solvable, Eq. (2.47) and Eq. (2.48) may be used to determine the optimal policy and its expected reward. When the cardinality of the state space $\mathcal{X}$ is infinite, this process in general requires infinite computation and storage, although there are special cases (such as LQG) in which the reward functions admit a finite parameterization.

2.2.1 Partially observed Markov decision processes

Partially observed MDPs (POMDPs) are a special case of MDPs in which one seeks to control a dynamical system for which one never obtains the exact value of the state of the system (denoted $\boldsymbol{x}_k$), but rather only noise-corrupted observations (denoted $\boldsymbol{z}_k$) of some portion of the system state at each time step. In this case, one can reformulate the problem as a fully observed MDP, in which the decision state is either the information vector (*i.e.*, the history of all controls applied to the system and the resulting observations), or the conditional probability distribution of system state conditioned on previously received observations (which forms a sufficient statistic for the information vector [9]).

The fundamental assumption of POMDPs is that the reward per stage and terminal reward, $g_k(\boldsymbol{x}_k, u_k)$ and $g_N(\boldsymbol{x}_N)$, can be expressed as functions of the state of the underlying system. Given the conditional probability distribution of system state, one can then calculate an induced reward as the expected value of the given quantity, *i.e.*,

$$g_k(\mathbb{X}_k, u_k) = \sum_{\boldsymbol{x}_k} g_k(\boldsymbol{x}_k, u_k) p(\boldsymbol{x}_k|\boldsymbol{z}_{0:k-1}; u_{0:k-1}) \tag{2.49}$$

$$g_N(\mathbb{X}_N) = \sum_{\boldsymbol{x}_N} g_N(\boldsymbol{x}_N) p(\boldsymbol{x}_N|\boldsymbol{z}_{0:N-1}; u_{0:N-1}) \tag{2.50}$$

where $\mathbb{X}_k = p(\boldsymbol{x}_k|\boldsymbol{z}_{0:k-1}; u_{0:k-1})$ is the conditional probability distribution which forms the decision state of the system. There is a unique structure which results: the induced reward per stage will be a linear function of the decision state, $\mathbb{X}_k$. In this case, one can show [87] that the reward to go function at all time steps will subsequently be a piecewise linear convex[2] function of the conditional probability distribution $\mathbb{X}_k$, *i.e.*, for some $\mathcal{I}_k$ and some $v_k^i(\cdot)$, $i \in \mathcal{I}_k$,

$$J_k^*(\mathbb{X}_k) = \max_{i \in \mathcal{I}_k} \sum_{\boldsymbol{x}_k} v_k^i(\boldsymbol{x}_k) p(\boldsymbol{x}_k|\boldsymbol{z}_{0:k-1}; u_{0:k-1}) \tag{2.51}$$

Solution strategies for POMDPs exploit this structure extensively, solving for an optimal (or near-optimal) choice of these parameters. The limitation of these methods is that they are restricted to small state spaces, as the size of the set $\mathcal{I}_k$ which is needed in practice grows rapidly with the number of states, observation values and control values. Theoretically, POMDPs are PSPACE-complete (*i.e.*, as hard as any problem which is solvable using an amount memory that is polynomial in the problem size, and unlimited computation time) [15, 78]; empirically, a 1995 study [63] found solution times

[2] or piecewise linear concave in the case of minimizing cost rather than maximizing reward.

in the order of hours for problems involving fifteen underlying system states, fifteen observation values and four actions. Clearly, such strategies are intractable in cases where there is an infinite state space, *e.g.*, when the underlying system state involves continuous elements such as position and velocity. Furthermore, attempts to discretize this type of state space are also unlikely to arrive at a sufficiently small number of states for this class of algorithm to be applicable.

■ 2.2.2 Open loop, closed loop and open loop feedback

The assumption inherent in MDPs is that the decision state is revealed to the controller at each decision stage. The optimal policy uses this new information as it becomes available, and anticipates the arrival of future information through the Markov transition model. This is referred to as closed loop control (CLC). Open loop control (OLC) represents the opposite situation: where one constructs a plan for the finite horizon (*i.e.*, single choice of which control to apply at each time, as opposed to a policy), and neither anticipates the availability of future information, nor utilizes that information as it arrives.

Open Loop Feedback Control (OLFC) is a compromise between these two extremes: like an open loop controller, a plan (rather than a policy) is constructed for the finite horizon at each step of the problem. This plan does not anticipate the availability future information; only a policy can do so. However, unlike an open loop controller, when new information is received it is utilized in constructing an updated plan. The controller operates by constructing a plan for the finite horizon, executing one or more steps of that plan, and then constructing a new plan which incorporates the information received in the interim.

There are many problems in which solving the MDP is intractable, yet open loop plans can be found within computational limitations. The OLFC is a commonly used suboptimal method in these situations. One can prove that the performance of the optimal OLFC is no worse than the optimal OLC [9]; the difference in performance between OLFC and CLC can be arbitrarily large.

■ 2.2.3 Constrained dynamic programming

In Chapter 5 we will consider sensor resource management in sensor networks. In this application, there is a fundamental trade-off which arises between estimation performance and energy cost. A natural way of approaching such a trade-off is as a constrained optimization, optimizing one quantity subject to a constraint on the other.

Constrained dynamic programming has been explored by previous authors in [2,13,18,94]. We describe a method based on Lagrangian relaxation, similar to that in [18], which yields a convenient method of approximate evaluation for the problem examined in Chapter 5.

We seek to minimize the cost over an N-step rolling horizon, *i.e.*, at time k, we minimize the cost incurred in the planning horizon involving steps $\{k, \dots, k+N-1\}$. Denoting by $\mu_k(\mathbb{X}_k)$ the control policy for time k, and by $\pi_k = \{\mu_k, \dots, \mu_{k+N-1}\}$ the set of policies for the next N time steps, we seek the policy corresponding to the optimal solution to the constrained minimization problem:

$$\begin{aligned} \min_{\pi} \mathrm{E}\left[\sum_{i=k}^{k+N-1} g(\mathbb{X}_i, \mu_i(\mathbb{X}_i))\right] & \\ \text{s.t.}\, \mathrm{E}\left[\sum_{i=k}^{k+N-1} G(\mathbb{X}_i, \mu_i(\mathbb{X}_i))\right] \le M & \end{aligned} \tag{2.52}$$

where $g(\mathbb{X}_k, u_k)$ is the per-stage cost and $G(\mathbb{X}_k, u_k)$ is the per-stage contribution to the additive constraint function. We address the constraint through a Lagrangian relaxation, a common approximation method for discrete optimization problems, by defining the dual function:

$$J_k^D(\mathbb{X}_k, \lambda) = \min_{\pi} \mathrm{E}\left[\sum_{i=k}^{k+N-1} g(\mathbb{X}_i, \mu_i(\mathbb{X}_i)) + \lambda\left(\sum_{i=k}^{k+N-1} G(\mathbb{X}_i, \mu_i(\mathbb{X}_i)) - M\right)\right] \tag{2.53}$$

and solving the dual optimization problem involving this function:

$$J_k^L(\mathbb{X}_k) = \max_{\lambda \ge 0} J_k^D(\mathbb{X}_k, \lambda) \tag{2.54}$$

We note that the dual function $J_k^D(\mathbb{X}_k, \lambda)$ takes the form of an unconstrained dynamic program with a modified per-stage cost:

$$\bar{g}(\mathbb{X}_k, u_k, \lambda) = g(\mathbb{X}_k, u_k) + \lambda G(\mathbb{X}_k, u_k) \tag{2.55}$$

The optimization of the dual problem provides a lower bound to the minimum value of the original constrained problem; the presence of a duality gap is possible since the optimization space is discrete. The size of the duality gap is given by the expression $\lambda\, \mathrm{E}[\sum_i G(\mathbb{X}_i, \mu_i(\mathbb{X}_i)) - M]$, where $\pi_k = \{\mu_i(\cdot,\cdot)\}_{i=k:k+N-1}$ is the policy attaining the minimum in Eq. (2.53) for the value of λ attaining the maximum in Eq. (2.54). If it happens that the optimal solution produced by the dual problem has no duality gap, then the resulting solution is also the optimal solution of the original constrained problem. This can occur in one of two ways: either the Lagrange multiplier λ is zero, such that the solution of the unconstrained problem satisfies the constraint, or the solution yields a result for which the constraint is tight. If a duality gap exists, a better solution may exist satisfying the constraint; however, the solution returned would have been optimal if the constraint level had been lower such that the constraint was tight. The method described in [18] avoids a duality gap utilizing randomized policies.

Conceptually, the dual problem in Eq. (2.54) can be solved using a subgradient method [10]. The following expression can be seen to be a supergradient[3] of the dual objective:

$$S(\mathbb{X}_k, \pi_k, \lambda) = \mathrm{E}\left[\sum_i G(\mathbb{X}_i, \mu_i(\mathbb{X}_i)) - M\right] \tag{2.56}$$

In other words, $S(\mathbb{X}_k, \pi_k, \lambda) \in \partial J_k^D(\mathbb{X}_k, \lambda)$, where ∂ denotes the superdifferential, *i.e.*, the set of all supergradients. The subgradient method operates according to the same principle as a gradient search, iteratively stepping in the direction of a subgradient with a decreasing step size [10]. For a single constraint, one may also employ methods such a line search; for multiple constraints the linear programming column generation procedure described in [16, 94] can be more efficient.

2.3 Information theoretic objectives

In some circumstances, the most appropriate choice of reward function might be obvious from the system specification. For example, if a sensing system is being used to estimate the location of a stranded yachtsman in order to minimize the distance from the survivor to where air-dropped supplies land, then a natural objective would be to minimize the expected landing distance, or to maximize the probability that the distance is less than a critical threshold. Each of these relates directly to a specific quantity at a specific time. As the high-level system objective becomes further removed from the performance of the sensing system, the most appropriate choice of reward function becomes less apparent. When an application demands continual tracking of multiple objects without a direct terminal objective, it is unclear what reward function should be selected.

Entropy is a commonly-used measure of uncertainty in many applications including sensor resource management, *e.g.*, [32, 41, 61, 95]. This section explores the definitions of and basic inequalities involving entropy and mutual information. All of the results

[3] Since we are maximizing a non-differentiable concave function rather than minimizing a non-differentiable convex function, subgradients are replaced by supergradients.

presented are well-known, and can be found in classical texts such as [23]. Throughout this document we use Shannon's entropy, as opposed to the generalization referred to as Renyi entropy. We will exploit various properties that are unique to Shannon entropy.

■ 2.3.1 Entropy

Entropy, joint entropy and conditional entropy are defined as:

$$
\begin{aligned}
H(x) &= -\int p(x)\log p(x)\mathrm{d}x && (2.57)\\
H(x,z) &= -\iint p(x,z)\log p(x,z)\mathrm{d}x\mathrm{d}z && (2.58)\\
H(x|z) &= -\int p(z)\int p(x|z)\log p(x|z)\mathrm{d}x\mathrm{d}z && (2.59)\\
&= H(x,z) - H(z) && (2.60)
\end{aligned}
$$

The above definitions relate to differential entropy, which concerns continuous variables. If the underlying sets are discrete, then a counting measure is used, effectively replacing the integral by a summation. In the discrete case, we have $H(x) \geq 0$. It is traditional to use a base-2 logarithm when dealing with discrete variables, and a natural logarithm when dealing with continuous quantities. We will also use a natural logarithm in cases involving a mixture of continuous and discrete quantities.

The conditioning in $H(x|z)$ in Eq. (2.59) is on the *random variable* z, hence an expectation is performed over the possible values that the variable may ultimately assume. We can also condition on a particular value of a random variable:

$$H(x|z=\zeta) = -\int p(x|z=\zeta)\log p(x|z=\zeta)\mathrm{d}x \qquad (2.61)$$

We will sometimes use the notation $H(x|\check{z})$ to denote conditioning on a particular value, *i.e.*, $H(x|\check{z}) \triangleq H(x|z=\check{z})$. Comparing Eq. (2.59) and Eq. (2.61), we observe that:

$$H(x|z) = \int p_z(\zeta)H(x|z=\zeta)\mathrm{d}\zeta \qquad (2.62)$$

■ 2.3.2 Mutual information

Mutual information (MI) is defined as the expected reduction in entropy in one random variable due to observation of another variable:

$$\begin{aligned} I(x;z) &= \iint p(x,z) \log \frac{p(x,z)}{p(x)p(z)} \mathrm{d}x \mathrm{d}z && (2.63) \\ &= H(x) - H(x|z) && (2.64) \\ &= H(z) - H(z|x) && (2.65) \\ &= H(x) + H(z) - H(x,z) && (2.66) \end{aligned}$$

Like conditional entropy, conditional MI can be defined with conditioning on either a random variable, or a particular value. In either case, the conditioning appears in all terms of the definition, *i.e.*, in the case of conditioning on a random variable y,

$$\begin{aligned} I(x;z|y) &= \int p_y(\psi) \iint p(x,z|y=\psi) \log \frac{p(x,z|y=\psi)}{p(x|y=\psi)p(z|y=\psi)} \mathrm{d}x \mathrm{d}z \mathrm{d}\psi && (2.67) \\ &= H(x|y) - H(x|z,y) && (2.68) \\ &= H(z|y) - H(z|x,y) && (2.69) \\ &= H(x|y) + H(z|y) - H(x,z|y) && (2.70) \end{aligned}$$

and in the case of conditioning on a particular value, ψ:

$$\begin{aligned} I(x;z|y=\psi) &= \iint p(x,z|y=\psi) \log \frac{p(x,z|y=\psi)}{p(x|y=\psi)p(z|y=\psi)} \mathrm{d}x \mathrm{d}z && (2.71) \\ &= H(x|y=\psi) - H(x|z,y=\psi) && (2.72) \\ &= H(z|y=\psi) - H(z|x,y=\psi) && (2.73) \\ &= H(x|y=\psi) + H(z|y=\psi) - H(x,z|y=\psi) && (2.74) \end{aligned}$$

Again, we will sometimes use the notation $I(x;z|\check{y})$ to indicate conditioning on a particular value, *i.e.* $I(x;z|\check{y}) \triangleq I(x;z|y=\check{y})$. Also note that, like conditional entropy, we can write:

$$I(x;z|y) = \int p_y(\psi) I(x;z|y=\psi) \mathrm{d}\psi \qquad (2.75)$$

The chain rule of mutual information allows us to expand a mutual information expression into the sum of terms:

$$I(x;z_1,\dots,z_n) = \sum_{i=1}^{n} I(x;z_i|z_1,\dots,z_{i-1}) \qquad (2.76)$$

The i-th term in the sum, $I(x; z_i|z_1, \dots, z_{i-1})$ represents the incremental gain we obtain in our knowledge of x due to the new observation z_i, conditioned on the previous observation random variables $(z_1, \dots, z_{i-1})$.

Suppose we have observations $(z_1, \dots, z_n)$ of an underlying state $(x_1, \dots, x_n)$, where observation z_i is independent of all other observations and underlying states when conditioned on x_i. In this case, we find:

$$\begin{aligned} I(x_1, \dots, x_n; z_i) &= H(z_i) - H(z_i|x_1, \dots, x_n) \\ &= H(z_i) - H(z_i|x_i) \\ &= I(x_i; z_i) \end{aligned} \tag{2.77}$$

In this case, the chain rule may be written as:

$$I(x_1, \dots, x_n; z_1, \dots, z_n) = \sum_{i=1}^{n} I(x_i; z_i|z_1, \dots, z_{i-1}) \tag{2.78}$$

It can be shown (through Jensen's inequality) that mutual information is nonnegative, *i.e.*, $I(x; z) \geq 0$, with equality if and only if x and z are independent.[4] Since $I(x; z) = H(x) - H(x|z)$, this implies that $H(x) \geq H(x|z)$, *i.e.*, that conditioning on a random variable reduces entropy. The following example illustrates that conditioning on a particular value of a random variable may not reduce entropy.

Example 2.1. *Suppose we want to infer a state $x \in \{0, \dots, N\}$ where $p(x = 0) = (N-1)/N$ and $p(x = i) = 1/N^2$, $i \neq 0$ (assume $N > 1$). Calculating the entropy, we obtain $H(x) = \log N - \frac{1}{N} \log \frac{(N-1)^{N-1}}{N} = \frac{1}{N} \log N + \frac{1}{N} \log \frac{N^N}{(N-1)^{N-1}} > 0$. We have an observation $z \in \{0, 1\}$, with $p(z = 0|x = 0) = 1$ and $p(z = 0|x \neq 0) = 0$. If we receive the observation value $z = 0$, then we know that $x = 0$, hence $H(x|z = 0) = 0 < H(x)$. If we receive the observation value $z = 1$ then $H(x|z = 1) = \log N > H(x)$. Conditioning on the random variable, we find $H(x|z) = \frac{1}{N} \log N < H(x)$.*

■ 2.3.3 Kullback-Leibler distance

The relative entropy or Kullback-Leibler (KL) distance is a measure of the difference between two probability distributions, defined as:

$$D(p(x)||q(x)) = \int p(x) \log \frac{p(x)}{q(x)} \mathrm{d}x \tag{2.79}$$

[4] This is also true for conditional MI, with conditioning on either an observation random variable or an observation value, with equality iff x and z are independent under the respective conditioning.

Comparing Eq. (2.79) with Eq. (2.63), we obtain:

$$I(x;z) = D(p(x,z)||p(x)p(z)) \tag{2.80}$$

Manipulating Eq. (2.63) we also obtain:

$$I(x;z) = \int p(z) \int p(x|z) \log \frac{p(x|z)}{p(x)} \mathrm{d}x\mathrm{d}z = \mathop{\mathrm{E}}_{z} D(p(x|z)||p(x)) \tag{2.81}$$

Therefore the MI between x and z is equivalent to the expected KL distance between the posterior distribution $p(x|z)$ and the prior distribution $p(x)$.

Another interesting relationship can be obtained by considering the expected KL distance between the posterior given a set of observations $\{z_1, \dots, z_n\}$ from which we may choose a single observation, and the posterior given the single observation z_u ($u \in \{1, \dots, n\}$):

$$\begin{aligned}
&\mathrm{E}D(p(x|z_1, \dots, z_n)||p(x|z_u)) \\
&\quad = \int p(z_1, \dots, z_n) \int p(x|z_1, \dots, z_n) \log \frac{p(x|z_1, \dots, z_n)}{p(x|z_u)} \mathrm{d}x\mathrm{d}z_1, \dots, \mathrm{d}z_n \\
&\quad = \int p(x, z_1, \dots, z_n) \log \frac{p(x, z_1, \dots, z_n)p(z_u)}{p(x, z_u)p(z_1, \dots, z_n)} \mathrm{d}x\mathrm{d}z_1, \dots, \mathrm{d}z_n \\
&\quad = \int p(x, z_1, \dots, z_n) \left[\log \frac{p(x)p(z_u)}{p(x, z_u)} + \log \frac{p(z_1, \dots, z_n, x)}{p(x)p(z_1, \dots, z_n)}\right] \mathrm{d}x\mathrm{d}z_1, \dots, \mathrm{d}z_n \\
&\quad = -I(x; z_u) + I(x; z_1, \dots, z_n)
\end{aligned}$$

Since the second term is invariant to the choice of u, we obtain the result that choosing u to maximize the MI between the state x and the observation z_u is equivalent to minimizing the expected KL distance between the posterior distribution of x given *all* observations and the posterior given only the chosen observation z_u.

■ 2.3.4 Linear Gaussian models

Entropy is closely related to variance for Gaussian distributions. The entropy of an n-dimensional multivariate Gaussian distribution with covariance $\mathbf{P}$ is equal to $\frac{1}{2}\log|2\pi e\mathbf{P}| = \frac{n}{2}\log 2\pi e + \frac{1}{2}\log|\mathbf{P}|$. Thus, under linear-Gaussian assumptions, minimizing conditional entropy is equivalent to minimizing the determinant of the posterior covariance, or the volume of the uncertainty hyper-ellipsoid.

Suppose $\boldsymbol{x}$ and $\boldsymbol{z}$ are jointly Gaussian random variables with covariance:

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{\boldsymbol{x}} & \mathbf{P}_{\boldsymbol{xz}} \\ \mathbf{P}_{\boldsymbol{xz}}^T & \mathbf{P}_{\boldsymbol{z}} \end{bmatrix}$$

Then the mutual information between $\boldsymbol{x}$ and $\boldsymbol{z}$ is given by:

$$I(\boldsymbol{x};\boldsymbol{z}) = \frac{1}{2}\log\frac{|\mathbf{P}_{\boldsymbol{x}}|}{|\mathbf{P}_{\boldsymbol{x}} - \mathbf{P}_{\boldsymbol{xz}}\mathbf{P}_{\boldsymbol{z}}^{-1}\mathbf{P}_{\boldsymbol{xz}}^T|} \tag{2.82}$$

$$= \frac{1}{2}\log\frac{|\mathbf{P}_{\boldsymbol{z}}|}{|\mathbf{P}_{\boldsymbol{z}} - \mathbf{P}_{\boldsymbol{xz}}^T\mathbf{P}_{\boldsymbol{x}}^{-1}\mathbf{P}_{\boldsymbol{xz}}|} \tag{2.83}$$

In the classical linear Gaussian case (to which the Kalman filter applies), where $\boldsymbol{z} = \mathbf{H}\boldsymbol{x} + \boldsymbol{v}$ and $\boldsymbol{v} \sim \mathcal{N}\{\boldsymbol{v}; \mathbf{0}, \mathbf{R}\}$ is independent of $\boldsymbol{x}$,

$$I(\boldsymbol{x};\boldsymbol{z}) = \frac{1}{2}\log\frac{|\mathbf{P}_{\boldsymbol{x}}|}{|\mathbf{P}_{\boldsymbol{x}} - \mathbf{P}_{\boldsymbol{x}}\mathbf{H}^T(\mathbf{H}\mathbf{P}_{\boldsymbol{x}}\mathbf{H}^T + \mathbf{R})^{-1}\mathbf{H}\mathbf{P}_{\boldsymbol{x}}|} \tag{2.84}$$

$$= \frac{1}{2}\log\frac{|\mathbf{P}_{\boldsymbol{x}}^{-1} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}|}{|\mathbf{P}_{\boldsymbol{x}}^{-1}|} \tag{2.85}$$

$$= \frac{1}{2}\log\frac{|\mathbf{H}\mathbf{P}_{\boldsymbol{x}}\mathbf{H}^T + \mathbf{R}|}{|\mathbf{R}|} \tag{2.86}$$

Furthermore, if $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ are jointly Gaussian then:

$$I(\boldsymbol{x};\boldsymbol{z}|\boldsymbol{y}) = I(\boldsymbol{x};\boldsymbol{z}|\boldsymbol{y} = \boldsymbol{\psi})\ \forall\ \boldsymbol{\psi} \tag{2.87}$$

The result of Eq. (2.87) is due to the fact that the posterior covariance in a Kalman filter is not affected by the observation value (as discussed in Section 2.1.2), and that entropy and MI are uniquely determined by the covariance of a Gaussian distribution.

■ 2.3.5 Axioms resulting in entropy

One may show that Shannon's entropy is the unique (up to a multiplicative constant) real-valued measure of the uncertainty in a discrete probability distribution which satisfies the following three axioms [7]. We assume $x \in \{x_1, \dots, x_n\}$ where $n < \infty$, and use the notation $p_{x_i} = P[x = x_i]$, and $H(p_{x_1}, \dots, p_{x_n}) = H(x)$.

1. $H(p_{x_1}, \dots, p_{x_n})$ is a continuous[5] function of the probability distribution $(p_{x_1}, \dots, p_{x_n})$ (defined for all n).

2. $H(p_{x_1}, \dots, p_{x_n})$ is permutation symmetric, *i.e.*, if $\pi(x)$ is a permutation of x then $H(p_{x_1}, \dots, p_{x_n}) = H(p_{\pi(x_1)}, \dots, p_{\pi(x_n)})$.

3. If $x_{n,1}$ and $x_{n,2}$ partition the event x_n (so that $p_{x_{n,1}} + p_{x_{n,2}} = p_{x_n} > 0$) then:

$$H(p_{x_1}, \dots, p_{x_{n-1}}, p_{x_{n,1}}, p_{x_{n,2}}) = H(p_{x_1}, \dots, p_{x_n}) + p(x_n)H\left(\frac{p_{x_{n,1}}}{p_{x_n}}, \frac{p_{x_{n,2}}}{p_{x_n}}\right)$$

[5] This condition can be relaxed to $H(x)$ being a Lebesgue integrable function of $p(x)$ (see [7]).

The final axiom relates to additivity of the measure: in effect it requires that, if we receive (in y) part of the information in a random variable x, then the uncertainty in x must be equal to the uncertainty in y plus the expected uncertainty remaining in x after y has been revealed.

2.3.6 Formulations and geometry

A common formulation for using entropy as an objective for sensor resource management is to seek to minimize the joint entropy of the state to be estimated over a rolling horizon. In this case, the canonical problem that we seek to solve is to find at time k the non-stationary policy $\pi_k = \{\mu_k, \dots, \mu_{k+N-1}\}$ which minimizes the expected entropy over the next N time steps conditioned on values of the observations already received:

$$\pi_k = \arg\min_{\mu_k,\dots,\mu_{k+N-1}} H(\boldsymbol{x}_k, \dots, \boldsymbol{x}_{k+N-1} | \check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1}, \boldsymbol{z}_k^{\mu_k(\mathbb{X}_k)}, \dots, \boldsymbol{z}_{k+N-1}^{\mu_{k+N-1}(\mathbb{X}_{k+N-1})}) \quad (2.88)$$

where $\mathbb{X}_k$ an appropriate choice of the decision state (discussed below). If we have a problem involving estimation of the state of multiple objects, we simply define $\boldsymbol{x}_k$ to be the joint state of the objects. Applying Eq. (2.72), we obtain the equivalent formulation: [25]

$$\begin{aligned}\pi_k &= \arg\min_{\mu_k,\dots,\mu_{k+N-1}} \Big[H(\boldsymbol{x}_k, \dots, \boldsymbol{x}_{k+N-1} | \check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1}) \\ &\qquad - I(\boldsymbol{x}_k, \dots, \boldsymbol{x}_{k+N-1}; \boldsymbol{z}_k^{\mu_k(\mathbb{X}_k)}, \dots, \boldsymbol{z}_{k+N-1}^{\mu_{k+N-1}(\mathbb{X}_{k+N-1})} | \check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1}) \Big] && (2.89) \\ &= \arg\max_{\mu_k,\dots,\mu_{k+N-1}} I(\boldsymbol{x}_k, \dots, \boldsymbol{x}_{k+N-1}; \boldsymbol{z}_k^{\mu_k(\mathbb{X}_k)}, \dots, \boldsymbol{z}_{k+N-1}^{\mu_{k+N-1}(\mathbb{X}_{k+N-1})} | \check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1}) && (2.90) \\ &= \arg\max_{\mu_k,\dots,\mu_{k+N-1}} \sum_{l=k}^{k+N-1} I(\boldsymbol{x}_l; \boldsymbol{z}_l^{\mu_l(\mathbb{X}_l)} | \check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1}, \boldsymbol{z}_k^{\mu_k(\mathbb{X}_k)}, \dots, \boldsymbol{z}_{l-1}^{\mu_{l-1}(\mathbb{X}_{l-1})}) && (2.91)\end{aligned}$$

Eq. (2.91) results from applying Eq. (2.78), assuming that the observations at time i are independent of each other and the remainder of the state conditioned on the state at time i.

This problem can be formulated as a MDP in which the reward per stage is chosen to be $g_k(\mathbb{X}_k, u_k) = I(\boldsymbol{x}_k; \boldsymbol{z}_k^{u_k} | \check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1})$. We choose the decision state to be the conditional PDF $\mathbb{X}_k = p(\boldsymbol{x}_k | \check{\boldsymbol{z}}_{0:k-1})$. Although conditioning is denoted on the history of observations $\{\check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1}\}$, the current conditional PDF is a sufficient statistic for all observations in calculation of the reward and the decision state at the next time. The structure of this MDP is similar to a POMDP in that the decision state is the conditional PDF of the underlying state. However, the reward per stage cannot be expressed as a linear function of the conditional PDF, and the reward to go is not piecewise linear convex.

From [23], the mutual information $I(x;z)$ is a concave function of $p(x)$ for a given $p(z|x)$, and a convex function of $p(z|x)$ for a given $p(x)$. Accordingly, by taking the maximum of the reward per stage over several different candidate observations, we are taking the point-wise maximum of several different concave functions of the PDF $p(x)$, which will in general result in a function which is non-concave and non-convex. This is illustrated in the following example.

Example 2.2. *Consider the problem in which the underlying state $x_k \in \{-1, 0, 1\}$ has a uniform prior distribution, and transitions according to the rule $p(x_k = i|x_{k-1} = i) = 1 - 2\epsilon$, and $p(x_k = i|x_{k-1} = j) = \epsilon$, $i \neq j$. Assume we have two observations available to us, z_k^1, $z_k^2 \in \{0, 1\}$, with the following models:*

$$p(z_k^1 = 1|x_k) = \begin{cases} 1-\delta, & x_k = -1 \\ \delta, & x_k = 0 \\ 0.5, & x_k = 1 \end{cases} \qquad p(z_k^2 = 1|x_k) = \begin{cases} 0.5, & x_k = -1 \\ \delta, & x_k = 0 \\ 1-\delta, & x_k = 1 \end{cases}$$

Contour plots of the optimal reward to go function for a single time step and for four time steps are shown in Fig. 2.1, with $\epsilon = 0.075$ and $\delta = 0.1$. The diagrams illustrate the non-concave structure which results.

The structure underlying the maximization problem within a single stage can also be revealed through these basic geometric observations. For example, suppose we are choosing between different observations z^1 and z^2 which share the same cardinality and have models $p(z^1|x)$ and $p(z^2|x)$. Consider a continuous relaxation of the problem in which we define the "quasi-observation" z^α with $p(z^\alpha|x) = \alpha p(z^1|x) + (1-\alpha)p(z^2|x)$ for $\alpha \in [0, 1]$:

$$u = \arg\max_{\alpha \in [0,1]} I(x; z^\alpha)$$

In this case, the observation model $p(z^\alpha|x)$ is a linear function of α and, as discussed above, the mutual information $I(x;z^\alpha)$ is a convex function of the observation model $p(z^\alpha|x)$ for a given prior distribution $p(x)$. Thus this is a convex maximization, which only confirms that the optimal solution lies at an integer point, again exposing the combinatorial complexity of the problem. This is illustrated in the following example.

Example 2.3. *Consider a single stage of the example from Example 2.2. Assume a prior distribution $p(x_k = -1) = 0.5$ and $p(x_k = 0) = p(x_k = 1) = 0.25$. The mutual* *information of the state and the quasi-observation z^α is shown as a function of α in Fig. 2.2. The convexity of the function implies that gradient-based methods will only converge to an extreme point, not necessarily to a solution which is good in any sense.*

It should be noted that these observations relate to a particular choice of parameterization and continuous relaxation. Given the choice of objective, however, there are no known parameterizations which avoid these difficulties. Furthermore, there are known complexity results: for example, selection of the best n-element subset of observations to maximize MI is NP-complete [46].

■ 2.4 Set functions, submodularity and greedy heuristics

Of the methods discussed in the previous section, the greedy heuristic and extensions thereof provide the only generally applicable solution to the sensor management problem which is able to handle problems involving a large state space. The remarkable characteristic of this algorithm is that, in certain circumstances, one can establish bounds on the loss of performance for using the greedy method rather than the optimal method (which is intractable). Sections 2.4.1, 2.4.2 and 2.4.3 provide the theoretical background required to derive these bounds, mostly from [75] and [26], after which Sections 2.4.4 and 2.4.5 present proofs of the bounds from existing literature. These bounds will be adapted to alternative problem structures in Chapter 3.

Throughout this section (and Chapter 3) we assume open loop control, *i.e.*, we make all of our observation selections before any observation values are received. In practice, the methods described could be employed in an OLFC manner, as described in Section 2.2.2.

■ 2.4.1 Set functions and increments

A set function is a real-valued function which takes as its input subsets of a given set. For example, consider the function $f : 2^{\mathcal{U}} \to \mathbb{R}$ (where $2^{\mathcal{U}}$ denotes the set of subsets of the finite set $\mathcal{U}$) defined as:

$$f(\mathcal{A}) = I(x; z^{\mathcal{A}})$$

where $z^{\mathcal{A}}$ denotes the observations corresponding to the set $\mathcal{A} \subseteq \mathcal{U}$. Thus $f(\mathcal{A})$ would denote the information learned about the state x by obtaining the set of observations $z^{\mathcal{A}}$.

Definition 2.1 (Nonnegative)**.** *A set function f is nonnegative if $f(\mathcal{A}) \geq 0 \; \forall \; \mathcal{A}$.*

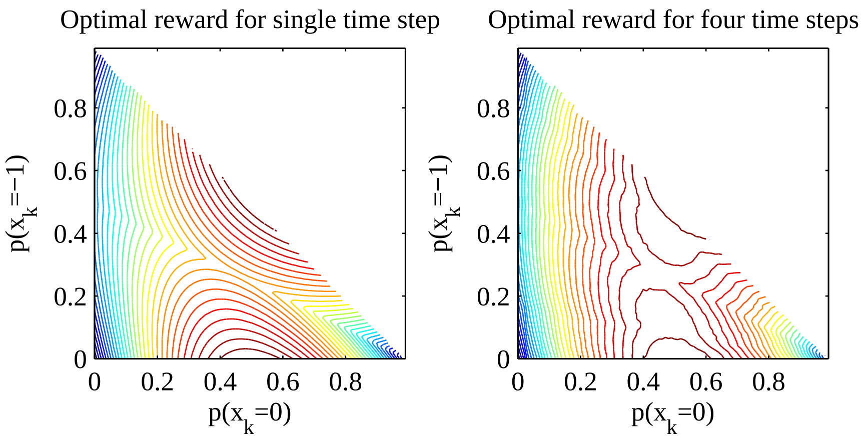


Figure 2.1. Contour plots of the optimal reward to go function for a single time step and for four time steps. Smaller values are shown in blue while larger values are shown in red.

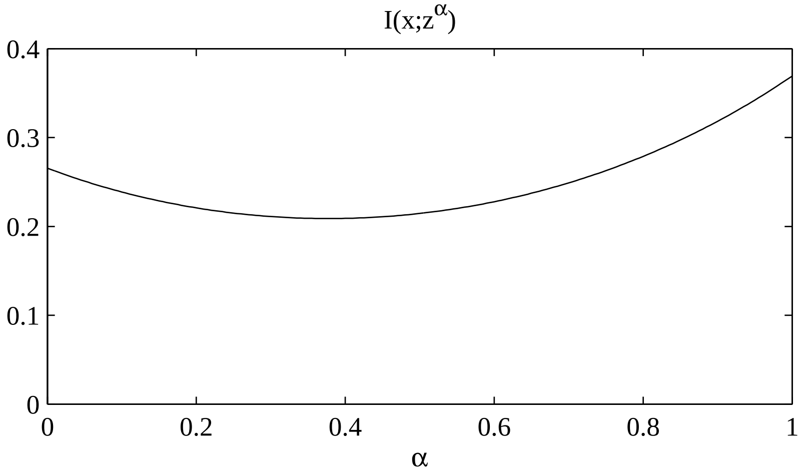


Figure 2.2. Reward in single stage continuous relaxation as a function of the parameter α.

Definition 2.2 (Nondecreasing)**.** *A set function f is non-decreasing if $f(\mathcal{B}) \geq f(\mathcal{A}) \; \forall \; \mathcal{B} \supseteq \mathcal{A}$.*

Obviously, a non-decreasing function will be nonnegative iff $f(\emptyset) \geq 0$. MI is an example of a nonnegative, non-decreasing function, since $I(x; \emptyset) = 0$ and $I(x; z^{\mathcal{B}}) - I(x; z^{\mathcal{A}}) = I(x; z^{\mathcal{B} \setminus \mathcal{A}} | z^{\mathcal{A}}) \geq 0$. Intuitively, this corresponds to the notion that including more observations must increase the information (*i.e.*, on average, the entropy is reduced).

Definition 2.3 (Increment function)**.** *We denote the single element increment by $\rho_j(\mathcal{A}) \triangleq f(\mathcal{A} \cup \{j\}) - f(\mathcal{A})$, and the set increment by $\rho_{\mathcal{B}}(\mathcal{A}) \triangleq f(\mathcal{A} \cup \mathcal{B}) - f(\mathcal{A})$.*

Applying the chain rule in reverse, the increment function for MI is equivalent to $\rho_{\mathcal{B}}(\mathcal{A}) = I(x; z^{\mathcal{B}} | z^{\mathcal{A}})$.

■ 2.4.2 Submodularity

Submodularity captures the notion that as we select more observations, the value of the remaining unselected observations decreases, *i.e.*, the notion of diminishing returns.

Definition 2.4 (Submodular)**.** *A set function f is submodular if $f(\mathcal{C} \cup \mathcal{A}) - f(\mathcal{A}) \geq f(\mathcal{C} \cup \mathcal{B}) - f(\mathcal{B}) \; \forall \; \mathcal{B} \supseteq \mathcal{A}$.*

From Definition 2.3, we note that $\rho_{\mathcal{C}}(\mathcal{A}) \geq \rho_{\mathcal{C}}(\mathcal{B}) \; \forall \; \mathcal{B} \supseteq \mathcal{A}$ for any increment function ρ arising from a submodular function f. The following lemma due to Krause and Guestrin [46] establishes conditions under which mutual information is a submodular set function.

Lemma 2.1. *If the observations are conditionally independent conditioned on the state, then the mutual information between the state and the subset of observations selected is submodular.*

Proof. Consider $\mathcal{B} \supseteq \mathcal{A}$:

$$\begin{aligned}
I(x; z^{\mathcal{C}\cup\mathcal{A}}) - I(x; z^{\mathcal{A}}) &\overset{(a)}{=} I(x; z^{\mathcal{C}\setminus\mathcal{A}}|z^{\mathcal{A}}) \\
&\overset{(b)}{=} I(x; z^{\mathcal{C}\setminus\mathcal{B}}|z^{\mathcal{A}}) + I(x; z^{\mathcal{C}\cap(\mathcal{B}\setminus\mathcal{A})}|z^{\mathcal{A}\cup(\mathcal{C}\setminus\mathcal{B})}) \\
&\overset{(c)}{\geq} I(x; z^{\mathcal{C}\setminus\mathcal{B}}|z^{\mathcal{A}}) \\
&\overset{(d)}{=} H(z^{\mathcal{C}\setminus\mathcal{B}}|z^{\mathcal{A}}) - H(z^{\mathcal{C}\setminus\mathcal{B}}|x, z^{\mathcal{A}}) \\
&\overset{(e)}{=} H(z^{\mathcal{C}\setminus\mathcal{B}}|z^{\mathcal{A}}) - H(z^{\mathcal{C}\setminus\mathcal{B}}|x) \\
&\overset{(f)}{\geq} H(z^{\mathcal{C}\setminus\mathcal{B}}|z^{\mathcal{B}}) - H(z^{\mathcal{C}\setminus\mathcal{B}}|x) \\
&\overset{(g)}{=} I(x; z^{\mathcal{C}\setminus\mathcal{B}}|z^{\mathcal{B}}) \\
&\overset{(h)}{=} I(x; z^{\mathcal{C}\cup\mathcal{B}}) - I(x; z^{\mathcal{B}})
\end{aligned}$$

(a), (b) and (h) result from the chain rule, (c) from nonnegativity, (d) and (g) from the definition of mutual information, (e) from the assumption that observations are independent conditioned on x, and (f) from the fact that conditioning reduces entropy. □

The simple result that we will utilize from submodularity is that $I(x; z^{\mathcal{C}}|z^{\mathcal{A}}) \geq I(x; z^{\mathcal{C}}|z^{\mathcal{B}})\ \forall\ \mathcal{B} \supseteq \mathcal{A}$. As discussed above, this may be intuitively understood as the notion of diminishing returns: that the new observations $z^{\mathcal{C}}$ are less valuable if the set of observations already obtained is larger.

The proof of Lemma 2.1 relies on the fact that conditioning reduces entropy. While this is true on average, it is necessarily not true for every value of the conditioning variable. Consequently, our proofs exploiting submodularity will apply to open loop control (where the value of future actions is averaged over all values of current observations) but not closed loop control (where the choice of future actions may change depending on the values of current observations).

Throughout this document, we will assume that the reward function f is nonnegative, non-decreasing and submodular, properties which mutual information satisfies.

■ 2.4.3 Independence systems and matroids

In many problems of interest, the set of observation subsets that we may select possesses particular structure. *Independence systems* provide a basic structure for which we can construct any valid set iteratively by commencing with an empty set and adding one element at a time in any order, maintaining a valid set at all times. The essential characteristic is therefore that any subset of a valid set is valid.

Definition 2.5 (Independence system)**.** *$(\mathcal{U}, \mathscr{F})$ is an independence system if $\mathscr{F}$ is a collection of subsets of $\mathcal{U}$ such that if $\mathcal{A} \in \mathscr{F}$ then $\mathcal{B} \in \mathscr{F}\ \forall\ \mathcal{B} \subseteq \mathcal{A}$. The members of $\mathscr{F}$ are termed independent sets, while subsets of $\mathcal{U}$ which are not members of $\mathscr{F}$ are termed dependent sets.*

The following example illustrates collections of sets which do and do not form independence systems.

Example 2.4. *Let $\mathcal{U} = \{a, b, c, d\}$, $\mathscr{F}_1 = \{\emptyset, \{a\}, \{b\}, \{c\}, \{d\}, \{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\}\}$, $\mathscr{F}_2 = \{\emptyset, \{a\}, \{b\}, \{c\}, \{d\}, \{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{a, b, c\}\}$ and $\mathscr{F}_3 = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{c, d\}\}$. Then $(\mathcal{U}, \mathscr{F}_1)$ and $(\mathcal{U}, \mathscr{F}_2)$ are independence systems, while $(\mathcal{U}, \mathscr{F}_3)$ is not since $\{d\} \subseteq \{c, d\}$ and $\{c, d\} \in \mathscr{F}_3$, but $\{d\} \notin \mathscr{F}_3$.*

We construct a subset by commencing with an empty set ($\mathcal{A}_0 = \emptyset$) and adding a new element at each iteration ($\mathcal{A}_i = \mathcal{A}_{i-1} \cup \{u_i\}$), ensuring that $\mathcal{A}_i \in \mathscr{F}\ \forall\ i$. When $(\mathcal{U}, \mathscr{F})$ is an independence system we are guaranteed that, if for some i, $\mathcal{A}_i \cup \{u\} \notin \mathscr{F}$ then $\mathcal{A}_j \cup \{u\} \notin \mathscr{F}\ \forall\ j > i$, *i.e.*, if we cannot add an element at a particular iteration, then we cannot add it at any later iteration either. The collection $\mathscr{F}_3$ in the above example violates this: we cannot extend $\mathcal{A}_0 = \emptyset$ with the element d in the first iteration, yet if we choose element c in the first iteration, then we can extend $\mathcal{A}_1 = \{c\}$ with the element d in the second iteration.

The iterative process for constructing a set terminates when we reach a point where adding any more elements yields a dependent (*i.e.*, invalid) set. Such a set is referred to as being *maximal*.

Definition 2.6 (Maximal)**.** *A set $\mathcal{A} \in \mathscr{F}$ is maximal if $\mathcal{A} \cup \{b\} \notin \mathscr{F}\ \forall\ b \in \mathcal{U} \backslash \mathcal{A}$.*

Matroids are a particular type of independence system for which efficient optimization algorithms exist for certain problems. The structure of a matroid is analogous with the that structure results from associating each element of $\mathcal{U}$ with a column of a matrix. Independent sets correspond to subsets of columns which are linearly independent, while dependent sets correspond to columns which are linearly dependent. We illustrate this below Example 2.5.

Definition 2.7 (Matroid)**.** *A matroid $(\mathcal{U}, \mathscr{F})$ is an independence system in which, for all $\mathcal{N} \subseteq \mathcal{U}$, all maximal sets of the collection $\mathscr{F}^{\mathcal{N}} \triangleq \{\mathcal{A} \in \mathscr{F} | \mathcal{A} \subseteq \mathcal{N}\}$ have the same cardinality.*

The collection $\mathscr{F}^{\mathcal{N}}$ represents the collection of sets in $\mathscr{F}$ whose elements are all contained in $\mathcal{N}$. Note that the definition does not require $\mathcal{N} \in \mathscr{F}$. The following lemma establishes an equivalent definition of a matroid.

Lemma 2.2. *An independence system $(\mathcal{U}, \mathscr{F})$ is a matroid if and only if $\forall\, \mathcal{A}, \mathcal{B} \in \mathscr{F}$ such that $|\mathcal{A}| < |\mathcal{B}|$, $\exists\, u \in \mathcal{B} \backslash \mathcal{A}$ such that $\mathcal{A} \cup \{u\} \in \mathscr{F}$.*

Proof. Only if: Consider $\mathscr{F}^{\mathcal{A}\cup\mathcal{B}}$ for any $\mathcal{A}, \mathcal{B} \in \mathscr{F}$ with $|\mathcal{A}| < |\mathcal{B}|$. Since $\mathcal{B} \in \mathscr{F}^{\mathcal{A}\cup\mathcal{B}}$, $\mathcal{A}$ cannot be maximal in $\mathscr{F}^{\mathcal{A}\cup\mathcal{B}}$, hence $\exists\, u \in \mathcal{B} \backslash \mathcal{A}$ such that $\mathcal{A} \cup \{u\} \in \mathscr{F}$.

If: Consider $\mathscr{F}^{\mathcal{N}}$ for any $\mathcal{N} \subseteq \mathcal{U}$. Let $\mathcal{A}$ and $\mathcal{B}$ be two maximal sets in $\mathscr{F}^{\mathcal{N}}$; note that $\mathcal{A}$, $\mathcal{B} \subseteq \mathcal{U}$. If $|\mathcal{A}| < |\mathcal{B}|$ then $\exists\, u \in \mathcal{B} \backslash \mathcal{A}$ such that $\mathcal{A} \cup \{u\} \in \mathscr{F}$, hence $\mathcal{A} \cup \{u\} \in \mathscr{F}^{\mathcal{N}}$ (noting the definition of $\mathscr{F}^{\mathcal{N}}$), contradicting the maximality of $\mathcal{A}$. □

The following example illustrates independence systems which are and are not matroids.

Example 2.5. *Let $\mathcal{U} = \{a, b, c, d\}$, $\mathscr{F}_1 = \{\emptyset, \{a\}, \{b\}, \{c\}, \{d\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}\}$, $\mathscr{F}_2 = \{\emptyset, \{a\}, \{b\}, \{c\}, \{d\}, \{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\}\}$ and $\mathscr{F}_3 = \{\emptyset, \{a\}, \{b\}, \{c\}, \{d\}, \{a, b\}, \{c, d\}\}$. Then $(\mathcal{U}, \mathscr{F}_1)$ and $(\mathcal{U}, \mathscr{F}_2)$ are matroids, while $(\mathcal{U}, \mathscr{F}_3)$ is not (applying Lemma 2.2 with $\mathcal{A} = \{a\}$ and $\mathcal{B} = \{c, d\}$).*

As discussed earlier, the structure of a matroid is analogous with the structure of linearly independent columns in a matroid. As an illustration of this, consider the matrices $\mathbf{M}_i = [\boldsymbol{c}_i^a\, \boldsymbol{c}_i^b\, \boldsymbol{c}_i^c\, \boldsymbol{c}_i^d]$, where $\boldsymbol{c}_i^u$ is the matrix column associated with element u in the matrix corresponding to $\mathscr{F}_i$. In the case of $\mathscr{F}_1$, the columns are such that $\boldsymbol{c}_1^a$ and $\boldsymbol{c}_1^b$ are linearly dependent, and $\boldsymbol{c}_1^c$ and $\boldsymbol{c}_1^d$ are linearly dependent, *e.g.*,

$$\boldsymbol{c}_1^a = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \boldsymbol{c}_1^b = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \boldsymbol{c}_1^c = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \boldsymbol{c}_1^d = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

This prohibits elements a and b from being selected together, and similarly for c and d. In the case of $\mathscr{F}_2$, the columns are such that any two are linearly independent, *e.g.*,

$$\boldsymbol{c}_2^a = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \boldsymbol{c}_2^b = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \boldsymbol{c}_2^c = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \boldsymbol{c}_2^d = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

The independence systems corresponding to each of the following problems may be seen to be a matroid:

- Selecting the best observation out of a set of at each time step over an N-step planning horizon (*e.g.*, $\mathscr{F}_1$ above)

- Selecting the best k-element subset of observations out of a set of n (*e.g.*, $\mathscr{F}_2$ above)
- The combination of these two: selecting the best k_i observations out of a set of n_i at each time step i over an N-step planning horizon
- Selecting up to k_i observations out of a set of n_i at each time step i over an N-step planning horizon, such that no more than K observations selected in total

■ 2.4.4 Greedy heuristic for matroids

The following is a simplified version of the theorem in [77] which establishes the more general result that the greedy algorithm applied to an independence system resulting from the intersection of P matroids achieves a reward no less than $\frac{1}{P+1}\times$ the reward achieved by the optimal set. The proof has been simplified extensively in order to specialize to the single matroid case; the simplifications illuminate the possibility of a different alternative theorem which will be possible for a different style of selection structure as discussed in Chapter 3. To our knowledge this bound has not previously been applied in the context of maximizing information.

Definition 2.8. *The greedy algorithm for selecting a set of observations in a matroid* $(\mathcal{U}, \mathscr{F})$ *commences by setting* $\mathcal{G}_0 = \emptyset$. *At each stage* $i = \{1, 2, \dots\}$, *the new element selected is:*

$$g_i = \underset{u \in \mathcal{U} \backslash \mathcal{G}_{i-1} | \mathcal{G}_{i-1} \cup \{u\} \in \mathscr{F}}{\arg\max} \rho_u(\mathcal{G}_{i-1})$$

where $\mathcal{G}_i = \mathcal{G}_{i-1} \cup \{g_i\}$. *The algorithm terminates when the set* $\mathcal{G}_i$ *is maximal.*

Theorem 2.3. *Applied to a submodular, non-decreasing function* $f(\cdot)$ *on a matroid, the greedy algorithm in Definition 2.8 achieves a reward no less than* $0.5\times$ *the reward achieved by the optimal set.*

Proof. Denote by $\mathcal{O}$ the optimal set, and by $\mathcal{G}$ the set chosen by the algorithm in Definition 2.8. Without loss of generality, assume that $|\mathcal{O}| = |\mathcal{G}|$ (since all maximal sets have the same cardinality, and since the objective is non-decreasing). Define $N \triangleq |\mathcal{O}|$, $\mathcal{O}_N = \mathcal{O}$, and for each $i = N-1, N-2, \dots, 1$, take $o_i \in \mathcal{O}_i$ such that $o_i \notin \mathcal{G}_{i-1}$, and $\mathcal{G}_{i-1} \cup \{o_i\} \in \mathscr{F}$ (such an element exists by Lemma 2.2). Define $\mathcal{O}_{i-1} \triangleq \mathcal{O}_i \backslash \{o_i\}$. The bound can be obtained through the following steps:

$$\begin{aligned}
f(\mathcal{O}) - f(\emptyset) &\overset{(a)}{\leq} f(\mathcal{G} \cup \mathcal{O}) - f(\emptyset) \\
&\overset{(b)}{\leq} f(\mathcal{G}) + \sum_{j \in \mathcal{O} \backslash \mathcal{G}} \rho_j(\mathcal{G}) - f(\emptyset) \\
&\overset{(c)}{\leq} f(\mathcal{G}) + \sum_{j \in \mathcal{O}} \rho_j(\mathcal{G}) - f(\emptyset) \\
&\overset{(d)}{=} f(\mathcal{G}) + \sum_{i=1}^{N} \rho_{o_i}(\mathcal{G}) - f(\emptyset) \\
&\overset{(e)}{\leq} f(\mathcal{G}) + \sum_{i=1}^{N} \rho_{o_i}(\mathcal{G}_{i-1}) - f(\emptyset) \\
&\overset{(f)}{\leq} f(\mathcal{G}) + \sum_{i=1}^{N} \rho_{g_i}(\mathcal{G}_{i-1}) - f(\emptyset) \\
&\overset{(g)}{=} 2(f(\mathcal{G}) - f(\emptyset))
\end{aligned}$$

where (a) and (c) result from the non-decreasing property, (b) and (e) result from submodularity, (d) is a simple rearrangement using the above construction $\mathcal{O} = \{o_1, \ldots, o_N\}$, (f) is a consequence of the structure of the greedy algorithm in Definition 2.8, and (g) is a simple rearrangement of (f). □

■ 2.4.5 Greedy heuristic for arbitrary subsets

The following theorem specializes the previous theorem to the case in which $\mathscr{F}$ consists of all K-element subsets of observations, as opposed to an arbitrary matroid. In this case, the bound obtainable is tighter. The theorem comes from [76], which addresses the more general case of a possibly decreasing reward function. Krause and Guestrin [46] first recognized that the bound can be applied to sensor selection problems with information theoretic objectives.

Theorem 2.4. *Suppose that* $\mathscr{F} = \{\mathcal{N} \subseteq \mathcal{U} \text{ s.t. } |\mathcal{N}| \leq K\}$*. Then the greedy algorithm applied to the non-decreasing submodular function* $f(\cdot)$ *on the independence system* $(\mathcal{U}, \mathscr{F})$ *achieves a reward no less than* $(1 - 1/e) \approx 0.632\times$ *the reward achieved by the optimal set.*

Proof. Denote by $\mathcal{O}$ the optimal K-element subset, and by $\mathcal{G}$ the set chosen by the algorithm in Definition 2.8. For each stage of the greedy algorithm, $i \in \{1, \ldots, K\}$, we can write from line (b) of the proof of Theorem 2.3:

$$f(\mathcal{O}) \leq f(\mathcal{G}_i) + \sum_{j \in \mathcal{O} \backslash \mathcal{G}_i} \rho_j(\mathcal{G}_i)$$

By definition of g_{i+1} in Definition 2.8, $\rho_{g_{i+1}}(\mathcal{G}_i) \geq \rho_j(\mathcal{G}_i) \; \forall \; j$, hence

$$\begin{aligned} f(\mathcal{O}) &\leq f(\mathcal{G}_i) + \sum_{j \in \mathcal{O} \backslash \mathcal{G}_i} \rho_{g_{i+1}}(\mathcal{G}_i) \\ &= f(\mathcal{G}_i) + |\mathcal{O} \backslash \mathcal{G}_i| \rho_{g_{i+1}}(\mathcal{G}_i) \\ &\leq f(\mathcal{G}_i) + K \rho_{g_{i+1}}(\mathcal{G}_i) \end{aligned}$$

By definition of the increment function ρ, we can write

$$f(\mathcal{G}_i) = f(\emptyset) + \sum_{j=1}^{i} \rho_{g_j}(\mathcal{G}_{j-1})$$

where we use the convention that $\mathcal{G}_0 = \emptyset$. Thus we can write $\forall \; i \in \{1, \ldots, K\}$:

$$f(\mathcal{O}) - f(\emptyset) \leq \sum_{j=1}^{i} \rho_{g_j}(\mathcal{G}_{j-1}) + K \rho_{g_{i+1}}(\mathcal{G}_i) \tag{2.92}$$

Now consider the linear program in variables $\rho_1, \ldots, \rho_{K+1}$, parameterized by Z:

$$\begin{aligned} P(Z) = &\min \sum_{j=1}^{K} \rho_j \\ &\text{s.t. } Z \leq \sum_{j=1}^{i} \rho_j + K \rho_{i+1}, \; i \in \{1, \ldots, K\} \end{aligned} \tag{2.93}$$

Taking the dual of the linear program:

$$\begin{aligned} D(Z) = &\max \sum_{j=1}^{K} Z x_j \\ &\text{s.t. } K x_i + \sum_{j=i+1}^{K} x_j = 1, \; i \in \{1, \ldots, K\} \end{aligned}$$

The system of constraints has a single solution which can found through a backward recursion commencing with x_K:

$$x_K = \frac{1}{K}, \; x_{K-1} = \frac{K-1}{K^2}, \; \ldots, \; x_1 = \frac{(K-1)^{K-1}}{K^K}$$

which yields

$$D(Z) = Z\left[1 - \left(\frac{K-1}{K}\right)^K\right]$$

which, by strong duality, is also the solution to $P(Z)$. Thus, any set $\mathcal{O}$ which respects the series of inequalities in Eq. (2.92) must have

$$\sum_{j=1}^{K} \rho_{g_j}(\mathcal{G}_{j-1}) = f(\mathcal{G}) - f(\emptyset) \geq \left[1 - \left(\frac{K-1}{K}\right)^K\right][f(\mathcal{O}) - f(\emptyset)]$$

Finally, note that

$$\left[1 - \left(\frac{K-1}{K}\right)^K\right] > 1 - 1/e \;\forall\; K > 1; \;\left[1 - \left(\frac{K-1}{K}\right)^K\right] \to 1 - 1/e,\; K \to \infty$$

Thus

$$f(\mathcal{G}) - f(\emptyset) \geq [1 - 1/e][f(\mathcal{O}) - f(\emptyset)] \geq 0.632[f(\mathcal{O}) - f(\emptyset)]$$

□

■ 2.5 Linear and integer programming

Chapter 4 will utilize an integer programming formulation to solve certain sensor resource management problems with manageable complexity. This section briefly outlines the idea behind linear and integer programming; the primary source for the material is [12].

■ 2.5.1 Linear programming

Linear programming is concerned with solving problems of the form:

$$\begin{aligned} &\min_{\boldsymbol{x}} \boldsymbol{c}^T\boldsymbol{x} \\ &\text{s.t. } \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b} \end{aligned} \tag{2.94}$$

Any problem of the form Eq. (2.94) can be converted to the standard form:

$$\begin{aligned} &\min_{\boldsymbol{x}} \boldsymbol{c}^T\boldsymbol{x} \\ &\text{s.t. } \mathbf{A}\boldsymbol{x} = \boldsymbol{b} \\ &\qquad \boldsymbol{x} \geq 0 \end{aligned} \tag{2.95}$$

The two primary mechanisms for solving problems of this type are simplex methods and interior point methods. Primal-dual methods simultaneously manipulate both the original problem (referred to as the primal problem), and a dual problem which provides a lower bound on the optimal objective value achievable. The difference between the current primal solution and the dual solution is referred to as the duality gap; this provides an upper bound on the improvement possible through further optimization. Interior point methods have been observed to be able to reduce the duality gap by a factor γ in a problem of size n in an average number of iterations of $O(\log n \log \gamma)$; the worst-case behavior is $O(\sqrt{n} \log \gamma)$.

■ 2.5.2 Column generation and constraint generation

In many large scale problems, it is desirable to be able to find a solution without explicitly considering all of the optimization variables. Column generation is a method which is used alongside the revised simplex method to achieve this goal. The method involves the iterative solution of a problem involving a small subset of the variables in the full problem. We assume availability of an efficient algorithm that tests whether incorporation of additional variables would be able to improve on the present solution. Occasionally this algorithm is executed, producing additional variables to be added to the subset. The process terminates when the problem involving the current subset of variables reaches an optimal solution and the algorithm producing new variables finds that there are no more variables able to produce further improvement.

Constraint generation is commonly used to solve large scale problems without explicitly considering all of the constraints. The method involves iterative solution of a problem involving the a small subset of the constraints in the full problem. We assume availability of an efficient algorithm that tests whether the current solution violates any of the constraints in the full problem, and returns one or more violated constraints if any exist. The method proceeds by optimizing the problem involving the subset of constraints, occasionally executing the algorithm to produce additional constraints that were previously violated. The process terminates when we find an optimal solution to the subproblem which does not violate any constraints in the full problem. Constraint generation may be interpreted as column generation applied to the dual problem.

■ 2.5.3 Integer programming

Integer programming deals with problems similar to Eqs. (2.94) and (2.95), but in which the optimization variables are constrained to take on integer values:

$$\begin{aligned} \min_{\boldsymbol{x}} \ & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t. } & \mathbf{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq 0 \\ & \boldsymbol{x} \text{ integer} \end{aligned} \tag{2.96}$$

Some problem structures possess a property in which, when the integer constraint is relaxed, there remains an integer point that attains the optimal objective. A common example is network flow problems in which the problem data takes on integer values. In this case, solution of the linear program (with the integrality constraint relaxed) can provide the optimal solution to the integer program. In general the addition of the integer constraint dramatically increases the computational complexity of finding a solution.

Relaxations

Two different relaxations are commonly used in integer programming: the linear programming relaxation, and the Lagrangian relaxation. The linear programming relaxation is exactly that described in the previous section: solving the linear program which results from relaxing the integrality constraint. If the problem possesses the necessary structure that there is an integer point that attains the optimal objective in the relaxed problem, then this point is also optimal in the original integer programming problem. In general this will not be the case, however the solution of the linear programming relaxation provides a lower bound to the solution of the integer program (since a wider range of solutions is considered).

As described in Section 2.2.3, Lagrangian relaxation involves solution of the Lagrangian dual problem. By weak duality, the dual problem also provides a lower bound on the optimal cost attainable in the integer program. However, since the primal problem involves a discrete optimization strong duality does not hold, and there may be a duality gap (*i.e.*, in general there will not be an integer programming solution that obtains the same cost as the solution of the Lagrangian dual). It can be shown that the Lagrangian relaxation provides a tighter lower bound than the linear programming relaxation.

Cutting plane methods

Let $\mathcal{X}$ be the set of feasible solutions to the integer program in Eq. (2.96) (*i.e.*, the integer points which satisfy the various constraints), and let $CH(\mathcal{X})$ be the convex hull of these points. Then the optimal solution of Eq. (2.96) is also an optimal solution of the following linear program:

$$\begin{aligned} &\min_{\boldsymbol{x}} \boldsymbol{c}^T \boldsymbol{x} \\ &\text{s.t. } \boldsymbol{x} \in CH(\mathcal{X}) \end{aligned} \tag{2.97}$$

Cutting plane methods, one of the two most common methods for solving integer programs, exploit this fact. We solve a series of linear programs, commencing with the linear programming relaxation. If, at any stage, we find an integer solution that is optimal, this is the optimal solution to the original problem. At each iteration, we add a constraint that is violated by the solution of the linear program in the current iteration, but is satisfied by every integer solution in the the original problem (*i.e.*, every point in $\mathcal{X}$). Thus the feasible region is slowly reduced, and approaches $CH(\mathcal{X})$. There are two difficulties associated with this method: firstly, it can be difficult to find constraints with the necessary characteristics; and secondly, it may be necessary to generate a very large number of constraints in order to obtain the integer solution.

Branch and bound

Branch and bound is the other commonly used method for solving integer programming problems. The basic concept is to divide the feasible region into sub-regions, and simultaneously search for "good" feasible solutions and tight lower bounds within in each sub-region. Any time we find that a sub-region R has a lower bound that is greater than a feasible solution found in another sub-region, the region R can be discarded. For example, suppose that we are dealing with a binary problem, where the feasible set is $\mathcal{X} = \{0, 1\}^N$. Suppose we branch on variable $\boldsymbol{x}_1$, *i.e.*, we divide the feasible region up into two sub-regions, where in the first we fix $\boldsymbol{x}_1 = 0$, and in the second we fix $\boldsymbol{x}_1 = 1$. Suppose we find a feasible solution within the sub-region in which $\boldsymbol{x}_1 = 1$ with objective 3.5, and, furthermore we find that the objective of the sub-problem in which we fix the value $\boldsymbol{x}_1 = 0$ is bounded below by 4.5. Then the optimal solution cannot have $\boldsymbol{x}_1 = 0$, so this sub-region may be discarded without further investigation. Linear programming relaxations and cutting plane methods are commonly used in concert with a branch and bound approach to provide the lower bounds.

2.6 Related work

The attention received by sensor resource management has steadily increased over the past two decades. The sections below summarize a number of strategies proposed by different authors. We coarsely categorize the material, although many of the methods do not fit precisely into any one category. We conclude in Section 2.6.7 by contrasting our approach to the work described.

2.6.1 POMDP and POMDP-like models

In [55, 56, 57], Krishnamurthy and Evans present two sensor management methods based upon POMDP methods. In [56, 57], the beam scheduling problem is cast as a multi-arm bandit, assuming that the conditional PDF of unobserved objects remains unchanged between decision intervals (this is slightly less restrictive than requiring the state itself to remain unchanged). Under these assumptions, it is proven that there exists an optimal policy in the form of index rule, and that the index function is piecewise linear concave. In [55], Krishnamurthy proposes a similar method for waveform selection problems in which the per stage reward is approximated by a piecewise linear concave function of the conditional PDF. In this regime, the reward to go function remains piecewise linear concave at each time in the backward recursion and POMDP methods can be applied. In [56], it is suggested that the continuous kinematic quantities could be discretized into coarse quantities such as “near” and “far”. A similar method is proposed for adaptive target detection in [60]. Computational examples in [55, 59] utilize underlying state space alphabets of three, six and 25. This reveals the primary limitation of this category of work: its inability to address problems with large state spaces. Many problems of practical interest cannot be represented with this restriction.

Castañón [18] formulates the problem of beam scheduling and waveform selection for identification of a large number of objects as a constrained dynamic program. By relaxing the sample path constraints to being constraints in expectation, a dual solution, which decouples the problem into a series of single object problems coupled only through the search for the correct values of the Lagrange multipliers, can be found using a method similar to that discussed in Section 2.2.3. By requiring observations at different times in the planning horizon to have identical characteristics, observations needing different time durations to complete are naturally addressed. The method is extended in [16] to produce a lower bound on the classification error performance in a sensor network. Again, the primary limitation of this method is the requirement for the state space to be small enough that traditional POMDP solution methods should be able to address the decoupled single object problem. In the context of object identification, this state space alphabet size restriction precludes addition of latent states such as object features (observations will often be dependent conditioned on the object class, but the object state can be expanded to incorporate continuous object features in order to regain the required conditional independence).

2.6.2 Model simplifications

Washburn, *et al* [93] observe that, after a minor transformation of the cost function, the solution method from multi-arm bandit problems method may be applied to beam steering, assuming that the state of unobserved objects remains unchanged. The policy based on this assumption is then used as a base policy in a roll-out with a one or two step look-ahead. The authors also suggest methods for practical application such as selecting as the decision state an estimate of the covariance matrix rather than conditional PDF, and simplifying stochastic disturbance to simple models such as detection and no detection. The ideas are explored further in [85].

2.6.3 Suboptimal control

Common suboptimal control methods such as roll-out [9] have also been applied to sensor management problems. Nedich, *et al* [74] consider tracking move-stop targets, and utilize an approximation of the cost to go of a heuristic base policy which captures the structure of the future reward for the given scenario. He and Chong [29] describe how a simulation-based roll-out method with an unspecified base policy could be used in combination with particle filtering.

2.6.4 Greedy heuristics and extensions

Many authors have approached the problem of waveform selection and beam steering using greedy heuristics which choose at each time the action which maximizes some instantaneous reward function. Information theoretic objectives are commonly used with this method; this may be expressed equivalently as

- Minimizing the conditional entropy of the state at the current time conditioned on the new observation (and on the values of the previous observations)
- Maximizing the mutual information between the state at the current time and the new observation

- Maximizing the Kullback-Leibler distance between the prior and posterior distribution.
- Minimizing the Kullback-Leibler distance between the posterior distribution and the posterior distribution that would be obtained if all possible observations were incorporated.

Use of these objectives in classification problems can be traced back as early as [61]. Hintz [32] and Kastella [41] appear to be two of the earliest instances in sensor fusion literature. Formally, if $g_k(\mathbb{X}_k, u_k)$ is the reward for applying control u_k at time k, then the greedy heuristic operates according to the following rule:

$$\mu_k^g(\mathbb{X}_k) = \arg \max_{u_k \in \mathcal{U}_k^{\mathbb{X}_k}} g_k(\mathbb{X}_k, u_k) \tag{2.98}$$

McIntyre and Hintz [69] apply the greedy heuristic, choosing mutual information as the objective in each stage to trade off the competing tasks of searching for new objects and maintaining existing tracks. The framework is extended to consider higher level goals in [31]. Kershaw and Evans [42] propose a method of adaptive waveform selection for radar/sonar applications. An analysis of the signal ambiguity function provides a prediction of the of the posterior covariance matrix. The use of a greedy heuristic is justified by a restriction to constant energy pulses. Optimization criteria include posterior mean square error and validation gate volume. The method is extended to consider the impact of clutter through Probabilistic Data Association (PDA) filter in [43].

Mahler [66] discusses the use of a generalization of Kullback-Leibler distance, global Csiszár c-discrimination, for sensor resource management within the finite set statistics framework, *i.e.*, where the number of objects is unknown and the PDF is invariant to any permutation of the objects. Kreucher, *et al* [49, 50, 51, 52, 54] apply a greedy heuristic to the sensor management problem in which the joint PDF of a varying number of objects is maintained using a particle filter. The objective function used is Renyi entropy; motivations for using this criterion are discussed in [50, 51]. Extensions to a two-step look-ahead using additional simulation, and a roll-out approach using a heuristic reward to go capturing the structure of long-term reward due to expected visibility and obscuration of objects are proposed in [53].

Kolba, *et al* [44] apply the greedy heuristic with an information objective to landmine detection, addressing the additional complexity which occurs when sensor motion is constrained. Singh, *et al* [86] show how the control variates method can be used to reduce the variance of estimates of Kullback-Leibler divergence (equivalent to mutual information) for use in sensor management; their experiments use a greedy heuristic.

Kalandros and Pao [40] propose a method which allows for control of process covariance matrices, *e.g.*, ensuring that the covariance matrix of a process $\mathbf{P}$ meets a specification $\mathbf{S}$ in the sense that $\mathbf{P} - \mathbf{S} \succeq 0$. This objective allows the system designer to dictate a more specific performance requirement. The solution method uses a greedy heuristic.

Zhao, *et al* [95] discuss object tracking in sensor networks, proposing methods based on greedy heuristics where the estimation objectives include nearest neighbor, Mahalanobis distance, entropy and Kullback-Leibler distance; inconsistencies with the measures proposed in this paper are discussed in [25].

Chhetri, *et al* [21] examine scheduling of radar and IR sensors for object tracking to minimize the mean square error over the next N time steps. The method proposed utilizes a linearized Kalman filter for evaluation of the error predictions, and performs a brute-force enumeration of all sequences within the planning horizon. Experiments are performed using planning horizons of one, two and three. In [20], the sensor network object tracking problem is approached by minimizing energy consumption subject to a constraint on estimation performance (measured using Cramér-Rao bounds). The method constructs an open loop plan by considering each candidate solution in ascending order of cost, and evaluating the estimation performance until a feasible solution is found (*i.e.*, one which meets the estimation criterion). Computational examples use planning horizons of one, two and three.

Logothetis and Isaksson [65] provide an algorithm for pruning the search tree in the problems involving control of linear Gauss-Markov systems with information theoretic criteria. If two candidate sequences obtain covariance matrices $\mathbf{P}_1$ and $\mathbf{P}_2$, and $\mathbf{P}_1 \succeq \mathbf{P}_2$, then the total reward of any extension of the first sequence will be less than or equal to the total reward of the same extension of the second sequence, thus the first sequence can be pruned from the search tree. Computational examples demonstrate a reduction of the tree width by a factor of around five.

Zwaga and Driessen examine the problem of selecting revisit rate and dwell time for a multifunction radar to minimize the total duty cycle consumed subject to a constraint on the post-update covariance [97] and prediction covariance [96]. Both methods consider only the current observation interval.

■ 2.6.5 Existing work on performance guarantees

Despite the popularity of the greedy heuristic, little work has been done to find guarantees of performance. In [17], Castañón shows that a greedy heuristic is optimal for the problem of dynamic hypothesis testing (*e.g.*, searching for an object among a finite set of positions) with symmetric measurement distributions (*i.e.*, P[missed detection] = P[false alarm]) according to the minimum probability of error criterion. In [33], Howard, *et al* prove optimality of greedy methods for the problem of beam scheduling of independent one-dimensional Gauss-Markov processes when the cost per stage is set to the sum of the error variances. The method does not extend to multi-dimensional processes.

Krause and Guestrin [46] apply results from submodular optimization theory to establish the surprising and elegant result that the greedy heuristic applied to the sensor subset selection algorithm (choosing the best n-element subset) is guaranteed to achieve performance within a multiple of $(1 - 1/e)$ of optimality (with mutual information as the objective), as discussed in Section 2.4.5. A similar performance guarantee is also established in [46] for the budgeted case, in which each observation incurs a cost, and there is a maximum total cost that can be expended. For this latter bound to apply, it is necessary to perform a greedy selection commencing from every three-element subset. The paper also establishes a theoretical guarantee that no polynomial time algorithm can improve on the performance bound unless $P = NP$, and discusses issues which arise when the reward to go values are computed approximately. This analysis is applied to the selection of sensor placements in [47], and the sensor placement model is extended to incorporate communication cost in [48].

■ 2.6.6 Other relevant work

Berry and Fogg [8] discuss the merits of entropy as a criterion for radar control, and demonstrate its application to sample problems. The suggested solutions include minimizing the resources necessary to satisfy a constraint on entropy, and selecting which targets to observe in a planning horizon in order to minimize entropy. No clear guidance is given on efficient implementation of the optimization problem resulting from either case. Moran, *et al* [70] discuss sensor management for radar, incorporating both selection of which waveform from a library to transmit at any given time, and how to design the waveform library *a priori*.

Hernandez, *et al* [30] utilize the posterior Cramér-Rao bound as a criterion for iterative deployment and utilization of sonar sensor resources for submarine tracking.

Sensor positions are determined using a greedy[6] simulated annealing search, where the objective is the estimation error at a particular time instant in the future. Selection of the subset of sensors to activate during tracking is performed either using brute-force enumeration or a genetic algorithm; the subset remains fixed in between sensor deployments.

The problem of selecting the subset of sensors to activate in a single time slot to minimize energy consumption subject to a mean square error estimation performance constraint is considered in [22]. An integer programming formulation using branch and bound techniques enables optimal solution of problems involving 50–70 sensors in tens of milliseconds. The branch and bound method used exploits quadratic structure that results when the performance criterion is based on two states (*i.e.*, position in two dimensions).

■ 2.6.7 Contrast to our contributions

As outlined in Section 2.6.4, many authors have applied greedy heuristics and short-time extensions (*e.g.*, using open loop plans over two or three time steps) to sensor management problems using criteria such as mutual information, mean square error and the posterior Cramér-Rao bound. Thus it is surprising that little work has been done toward obtaining performance guarantees for these methods. As discussed in Section 2.6.5, [46] is the first generally applicable performance guarantee to a problem with structure resembling the type which results to sensor management. However, this result is not directly applicable to sensor management problems involving sequential estimation (*e.g.*, object tracking), where there is typically a set of observations corresponding to each time slot, the elements of which correspond to the modes in which we may operate the sensor in that time slot, rather than a single set of observations. The typical constraint structure is one which permits selection of one element (or a small number of elements) from each of these sets, rather than a total of n elements from a larger subset.

The analysis in Chapter 3 extends the performance guarantees of [46] to problems involving this structure, providing the surprising result that a similar guarantee applies to the greedy heuristic applied in a sequential fashion, even though future observation opportunities are ignored. The result is applicable to a large class of time-varying models. Several extensions are obtained, including tighter bounds that exploit either

[6] The search is greedy in that proposed locations are accepted with probability 1 if the objective is improved and probability 0 otherwise.

process diffusiveness or objectives involving discount factors, and applicability to closed loop problems. We also show that several of the results may be applied to the posterior Cramér-Rao bound. Examples demonstrate that the bounds are tight, and counterexamples illuminate larger classes of problems to which they do not apply.

Many of the existing works that provide non-myopic sensor management either rely on very specific problem structure, and hence they are not generally applicable, or they do not allow extensions to planning horizons past two or three time slots due to the computational complexity. The development in Chapter 4 provides an integer programming approach that exploits submodularity to find optimal or near-optimal open loop plans for problems involving multiple objects over much longer planning horizons; experiments utilize up to 60 time slots. The method can be applied to any submodular, nondecreasing objective function, and does not require any specific structure in dynamics or observation models.

Finally, Chapter 5 approaches the problem of sensor management in sensor networks using a constrained dynamic programming formulation. The trade off between estimation performance and communication cost is formulated by maximizing estimation performance subject to a constraint on energy cost, or the dual of this, *i.e.*, minimizing energy cost subject to a constraint on estimation performance. Heuristic approximations that exploit the problem structure of tracking a single object using a network of sensors again enable planning over dramatically increased horizons. The method is both computable and scalable, yet still captures the essential structure of the underlying trade off. Simulation results demonstrate a significant reduction in the communication cost required to achieve a given estimation performance level as compared to previously proposed algorithms.

Chapter 3

Greedy heuristics and performance guarantees

THE performance guarantees presented in Section 2.4 apply to algorithms with a particular selection structure: firstly, where one can select any set within a matroid, and secondly where one can select any arbitrary subset of a given cardinality. This section develops a bound which is closely related to the matroid selection case, except that we apply additional structure, in which we have N subsets of observations and from each we can select a subset of a given cardinality. This structure is naturally applicable to dynamic models, where each subset of observations corresponds to a different time stage of the problem, *e.g.*, we can select one observation at each time stage. Our selection algorithm allows us to select at each time the observation which maximizes the reward at that time, ignorant of the remainder of the time horizon. The analysis establishes that the same performance guarantee that applies to the matroid selection problem also applies to this problem.

We commence the chapter by deriving the simplest form of the performance guarantee, with both online and offline variants, in Section 3.1. Section 3.2 examines a potentially tighter guarantee which exists for processes exhibiting diffusive characteristics, while Section 3.3 presents a similar guarantee for problems involving a discounted objective. Section 3.5 then extends the results of Sections 3.1–3.3 to closed loop policies.

While the results in Sections 3.1–3.5 are presented in terms of mutual information, they are applicable to a wider class of objectives. Sections 3.1 and 3.3 apply to any submodular, non-decreasing objective for which the reward of an empty set is zero (similar to the requirements for the results in Sections 2.4.4 and 2.4.5). The additional requirements in Sections 3.2 and 3.5 are discussed as the results are presented. In Section 3.6, we demonstrate that the log of the determinant of the Fisher information matrix is also submodular and non-decreasing, and thus that the various guarantees of

Sections 2.4.4, 2.4.5, 3.1 and 3.3 can also be applied to the determinant of the covariance through the Cramér-Rao bound.

Section 3.8 presents examples of some slightly different problem structures which do not fit into the structure examined in Section 3.1 and Section 3.2, but are still able to be addressed by the prior work discussed in Section 2.4.4. Finally, Section 3.9 presents a negative result regarding a particular extension of the greedy heuristic to platform steering problems.

■ 3.1 A simple performance guarantee

To commence, consider a simple sequential estimation problem involving two time steps, where at each step we must choose a single observation (*e.g.*, in which mode to operate a sensor) from a different set of observations. The goal is to maximize the information obtained about an underlying quantity X. Let $\{o_1, o_2\}$ denote the optimal choice for the two stages, which maximizes $I(X; z_1^{o_1}, z_2^{o_2})$. Let $\{g_1, g_2\}$ denote the choice made by the greedy heuristic, where g_1 is chosen to maximize $I(X; z_1^{g_1})$ and g_2 is chosen to maximize $I(X; z_2^{g_2}|z_1^{g_1})$ (where conditioning is on the random variable $z_1^{g_1}$, not on the resulting observation value). Then the following analysis establishes a performance guarantee for the greedy algorithm:

$$\begin{aligned}
I(X; z_1^{o_1}, z_2^{o_2}) &\overset{(a)}{\leq} I(X; z_1^{g_1}, z_2^{g_2}, z_1^{o_1}, z_2^{o_2}) \\
&\overset{(b)}{=} I(X; z_1^{g_1}) + I(X; z_2^{g_2}|z_1^{g_1}) + I(X; z_1^{o_1}|z_1^{g_1}, z_2^{g_2}) + I(x; z_2^{o_2}|z_1^{g_1}, z_2^{g_2}, z_1^{o_1}) \\
&\overset{(c)}{\leq} I(X; z_1^{g_1}) + I(X; z_2^{g_2}|z_1^{g_1}) + I(X; z_1^{o_1}) + I(x; z_2^{o_2}|z_1^{g_1}) \\
&\overset{(d)}{\leq} 2I(X; z_1^{g_1}) + 2I(X; z_2^{g_2}|z_1^{g_1}) \\
&\overset{(e)}{=} 2I(X; z_1^{g_1}, z_2^{g_2})
\end{aligned}$$

where (a) results from the non-decreasing property of MI, (b) is an application of the MI chain rule, (c) results from submodularity (assuming that all observations are independent conditioned on X), (d) from the definition of the greedy heuristic, and (e) from a reverse application of the chain rule. Thus the optimal performance can be no more than twice that of the greedy heuristic, or, conversely, the performance of the greedy heuristic is at least half that of the optimal.[1]

Theorem 3.1 presents this result in its most general form; the proof directly follows the above steps. The following assumption establishes the basic structure: we have N

[1] Note that this is considering only open loop control; we will discuss closed loop control in Section 3.5.

sets of observations, and we can select a specified number of observations from each set in an arbitrary order.

Assumption 3.1. *There are N sets of observations, $\{\{z_1^1, \dots, z_1^{n_1}\}, \{z_2^1, \dots, z_2^{n_2}\}, \dots, \{z_N^1, \dots, z_N^{n_N}\}\}$, which are mutually independent conditioned on the quantity to be estimated (X). Any k_i observations can be chosen out of the i-th set ($\{z_i^1, \dots, z_i^{n_i}\}$). The sequence $(w_1, \dots, w_M)$ (where $w_i \in \{1, \dots, N\}\ \forall\ i$) specifies the order in which we visit observation sets using the greedy heuristic (i.e., in the i-th stage we select a previously unselected observation out of the w_i-th set).*

Obviously we require $|\{j \in \{1, \dots, M\}|w_j = i\}| = k_i\ \forall\ i$ (*i.e.*, we visit the i-th set of observations k_i times, selecting a single additional observation at each time), thus $\sum_{i=1}^N k_i = M$. The abstraction of the observation set sequence $(w_1, \dots, w_M)$ allows us to visit observation sets more than once (allowing us to select multiple observations from each set) and in any order. The greedy heuristic operating on this structure is defined below.

Definition 3.1. *The greedy heuristic operates according to the following rule:*

$$g_j = \underset{u \in \{1,\dots,n_{w_j}\}}{\arg\max}\ I(X; z_{w_j}^u | z_{w_1}^{g_1}, \dots, z_{w_{j-1}}^{g_{j-1}})$$

We assume without loss of generality that the same observation is not selected twice since the reward for selecting an observation that was already selected is zero. We are now ready to state the general form of the performance guarantee.

Theorem 3.1. *Under Assumption 3.1, the greedy heuristic in Definition 3.1 has performance guaranteed by the following expression:*

$$I(X; z_{w_1}^{o_1}, \dots, z_{w_M}^{o_M}) \leq 2I(X; z_{w_1}^{g_1}, \dots, z_{w_M}^{g_M})$$

where $\{z_{w_1}^{o_1}, \dots, z_{w_M}^{o_M}\}$ is the optimal set of observations, i.e., the one which maximizes $I(X; z_{w_1}^{o_1}, \dots, z_{w_M}^{o_M})$.

Proof. The performance guarantee is obtained through the following steps:

$$
\begin{aligned}
&I(X;z^{o_1}_{w_1},\ldots,z^{o_M}_{w_M}) \\
&\overset{(a)}{\leq} I(X;z^{g_1}_{w_1},\ldots,z^{g_M}_{w_M},z^{o_1}_{w_1},\ldots,z^{o_M}_{w_M}) \\
&\overset{(b)}{=} \sum_{j=1}^{M} \left\{ I(X;z^{g_j}_{w_j}|z^{g_1}_{w_1},\ldots,z^{g_{j-1}}_{w_{j-1}}) + I(X;z^{o_j}_{w_j}|z^{g_1}_{w_1},\ldots,z^{g_M}_{w_M},z^{o_1}_{w_1},\ldots,z^{o_{j-1}}_{w_{j-1}}) \right\} \\
&\overset{(c)}{\leq} \sum_{j=1}^{M} \left\{ I(X;z^{g_j}_{w_j}|z^{g_1}_{w_1},\ldots,z^{g_{j-1}}_{w_{j-1}}) + I(X;z^{o_j}_{w_j}|z^{g_1}_{w_1},\ldots,z^{g_{j-1}}_{w_j}) \right\} \\
&\overset{(d)}{\leq} 2\sum_{j=1}^{M} I(X;z^{g_j}_{w_j}|z^{g_1}_{w_1},\ldots,z^{g_{j-1}}_{w_{j-1}}) \\
&\overset{(e)}{=} 2I(X;z^{g_1}_{w_1},\ldots,z^{g_M}_{w_M})
\end{aligned}
$$

where (a) is due to the non-decreasing property of MI, (b) is an application of the MI chain rule, (c) results from submodularity, (d) from Definition 3.1, and (e) from a reverse application of the chain rule. □

■ 3.1.1 Comparison to matroid guarantee

The prior work using matroids (discussed in Section 2.4.3) provides another algorithm with the same guarantee for problems of this structure. However, to achieve the guarantee on matroids it is necessary to consider every observation at every stage of the problem. Computationally, it is far more desirable to be able to proceed in a dynamic system by selecting observations at time k considering only the observations available at that time, disregarding future time steps (indeed, all of the previous works described in Section 2.6.4 do just that). The freedom of choice of the order in which we visit observation sets in Theorem 3.1 extends the performance guarantee to this commonly used selection structure.

■ 3.1.2 Tightness of bound

The bound derived in Theorem 3.1 can be arbitrarily close to tight, as the following example shows.

Example 3.1. *Consider a problem with $X = [a,b]^T$ where a and b are independent binary random variables with $P(a=0) = P(a=1) = 0.5$ and $P(b=0) = 0.5 - \epsilon$; $P(b=1) = 0.5 + \epsilon$ for some $\epsilon > 0$. We have two sets of observations with $n_1 = 2$,*

$n_2 = 1$ and $k_1 = k_2 = 1$. In the first set of observations we may measure $z_1^1 = a$ for reward $I(X; z_1^1) = H(a) = 1$, or $z_1^2 = b$ for reward $I(X; z_1^2) = H(b) = 1 - \delta(\epsilon)$, where $\delta(\epsilon) > 0 \; \forall \; \epsilon > 0$, and $\delta(\epsilon) \to 0$ as $\epsilon \to 0$. At the second stage we have one choice, $z_2^1 = a$. Our walk is $w = (1, 2)$, i.e., we visit the first set of observations once, followed by the second set.

The greedy algorithm selects at the first stage to observe $z_1^1 = a$, as it yields a higher reward (1) than $z_1^2 = b$ $(1 - \delta(\epsilon))$. At the second stage, the algorithm already has the exact value for a, hence the observation at the second stage yields zero reward. The total reward is 1.

Compare this result to the optimal sequence, which selects observation $z_1^2 = b$ for reward $1 - \delta(\epsilon)$, and then gains a reward of 1 from the second observation z_2^1. The total reward is $2 - \delta(\epsilon)$. By choosing ϵ arbitrarily close to zero, we may make the ratio of optimal reward to greedy reward, $2 - \delta(\epsilon)$, arbitrarily close to 2.

■ 3.1.3 Online version of guarantee

Modifying step (c) of Theorem 3.1, we can also obtain an online performance guarantee, which will often be substantially tighter in practice (as demonstrated in Sections 3.1.4 and 3.1.5).

Theorem 3.2. *Under the same assumptions as Theorem 3.1, for each $i \in \{1, \dots, N\}$ define $\bar{k}_i = \min\{k_i, n_i - k_i\}$, and for each $j \in \{1, \dots, \bar{k}_i\}$ define*

$$\bar{g}_i^j = \underset{u \in \{1,\dots,n_i\} - \{\bar{g}_i^l | l < j\}}{\arg\max} \; I(X; z_i^u | z_{w_1}^{g_1}, \dots, z_{w_M}^{g_M}) \tag{3.1}$$

Then the following two performance guarantees, which are computable online, apply:

$$I(X; z_{w_1}^{o_1}, \dots, z_{w_M}^{o_M}) \le I(X; z_{w_1}^{g_1}, \dots, z_{w_M}^{g_M}) + \sum_{i=1}^{N} \sum_{j=1}^{\bar{k}_i} I(X; z_i^{\bar{g}_i^j} | z_{w_1}^{g_1}, \dots, z_{w_M}^{g_M}) \tag{3.2}$$

$$\le I(X; z_{w_1}^{g_1}, \dots, z_{w_M}^{g_M}) + \sum_{i=1}^{N} \bar{k}_i I(X; z_i^{\bar{g}_i^1} | z_{w_1}^{g_1}, \dots, z_{w_M}^{g_M}) \tag{3.3}$$

Proof. The expression in Eq. (3.2) is obtained directly from step (b) of Theorem 3.1 through submodularity and the definition of $\bar{g}_i^j$ in Eq. (3.1). Eq. (3.3) uses the fact that $I(X; z_i^{\bar{g}^{j_1}} | z_{w_1}^{g_1}, \dots, z_{w_M}^{g_M}) \ge I(X; z_i^{\bar{g}^{j_2}} | z_{w_1}^{g_1}, \dots, z_{w_M}^{g_M})$ for any $j_1 \le j_2$. □

The online bound in Theorem 3.2 can be used to calculate an upper bound for the optimal reward starting from *any* sequence of observation choices, not just the choice made by the greedy heuristic in Definition 3.1, $(g_1, \dots, g_M)$. The online bound will tend to be tight in cases where the amount of information remaining after choosing the set of observations is small.

■ 3.1.4 Example: beam steering

Consider the beam steering problem in which two objects are being tracked. Each object evolves according to a linear Gaussian process:

$$\boldsymbol{x}^i_{k+1} = \mathbf{F}\boldsymbol{x}^i_k + \boldsymbol{w}^i_k$$

where $\boldsymbol{w}^i_k \sim \mathcal{N}\{\boldsymbol{w}_k; \mathbf{0}, \mathbf{Q}\}$ are independent white Gaussian noise processes. The state $\boldsymbol{x}^i_k$ is assumed to be position and velocity in two dimensions ($\boldsymbol{x}^i_k = [p^i_x \; v^i_x \; p^i_y \; v^i_y]^T$), where velocity is modelled as a continuous-time random walk with constant diffusion strength q (independently in each dimension), and position is the integral of velocity. Denoting the sampling interval as $T = 1$, the corresponding discrete-time model is:

$$\mathbf{F} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}; \; \mathbf{Q} = q \begin{bmatrix} \frac{T^3}{3} & \frac{T^2}{2} & 0 & 0 \\ \frac{T^2}{2} & T & 0 & 0 \\ 0 & 0 & \frac{T^3}{3} & \frac{T^2}{2} \\ 0 & 0 & \frac{T^2}{2} & T \end{bmatrix}$$

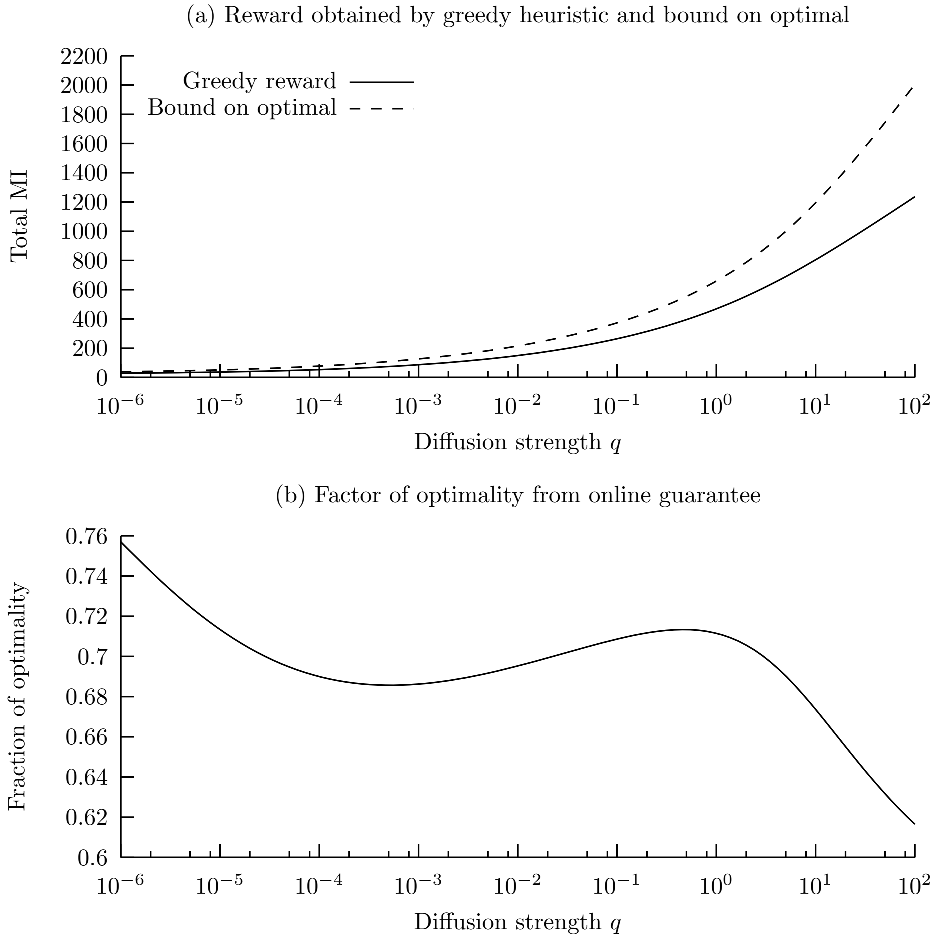
At each time instant we may choose between linear Gaussian measurements of the position of either object:

$$\boldsymbol{z}^i_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \boldsymbol{x}^i_k + \boldsymbol{v}^i_k$$

where $\boldsymbol{v}^i_k \sim \mathcal{N}\{\boldsymbol{v}^i_k; \mathbf{0}, \mathbf{I}\}$ are independent white Gaussian noise processes, independent of $\boldsymbol{w}^j_k \; \forall \; j, k$. The objects are tracked over a period of 200 time steps, commencing from an initial distribution $\boldsymbol{x}_0 \sim \mathcal{N}\{\boldsymbol{x}_0; \mathbf{0}, \mathbf{P}_0\}$, where

$$\mathbf{P}_0 = \begin{bmatrix} 0.5 & 0.1 & 0 & 0 \\ 0.1 & 0.05 & 0 & 0 \\ 0 & 0 & 0.5 & 0.1 \\ 0 & 0 & 0.1 & 0.05 \end{bmatrix}$$

Fig. 3.1 shows the bound on the fraction of optimality according to the guarantee of



(a) Reward obtained by greedy heuristic and bound on optimal

(b) Factor of optimality from online guarantee

Figure 3.1. (a) shows total reward accrued by the greedy heuristic in the 200 time steps for different diffusion strength values (q), and the bound on optimal obtained through Theorem 3.2. (b) shows the ratio of these curves, providing the factor of optimality guaranteed by the bound.

Theorem 3.2 as a function of the diffusion strength q. In this example, the quantity being estimated, X, is the combination of the states of both objects over all time steps in the problem:

$$X = [\boldsymbol{x}_1^1;\ \boldsymbol{x}_1^2;\ \boldsymbol{x}_2^1;\ \boldsymbol{x}_2^2;\ \ldots;\ \boldsymbol{x}_{200}^1;\ \boldsymbol{x}_{200}^2]$$

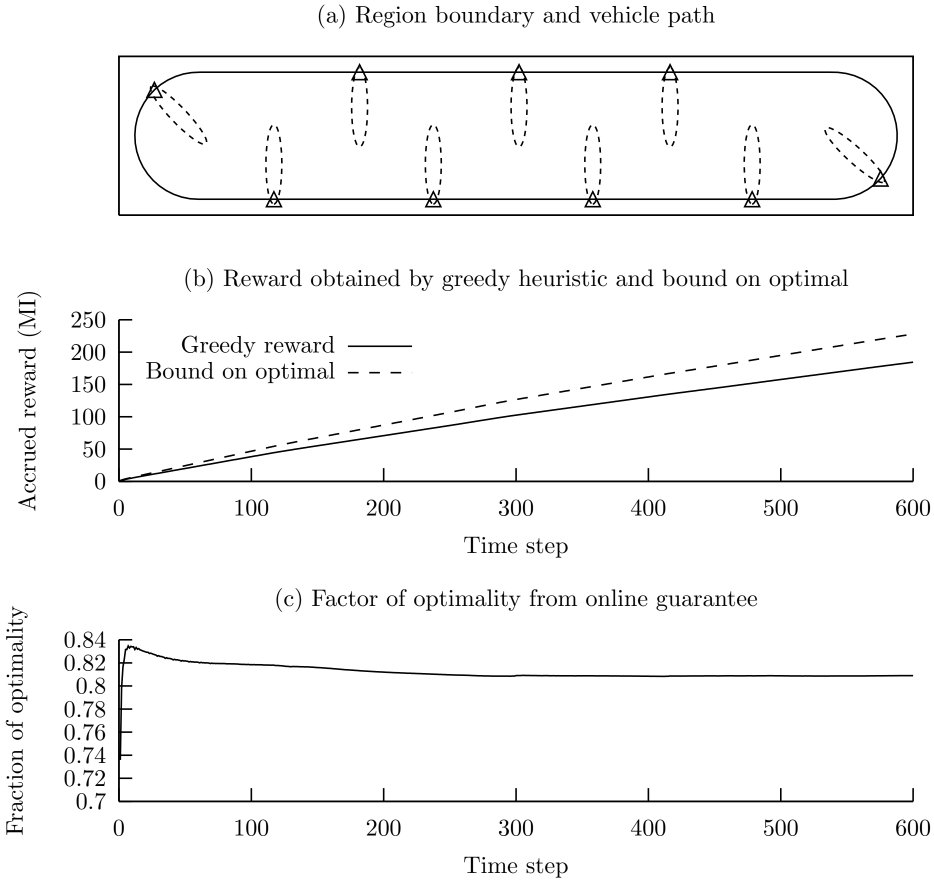
Examining Fig. 3.1, the greedy controller obtains close to the optimal amount of information as diffusion decreases since the measurements that were declined become highly correlated with the measurements that were chosen.

■ 3.1.5 Example: waveform selection

Suppose that we are using a surface vehicle travelling at a constant velocity along a fixed path (as illustrated in Fig. 3.2(a)) to map the depth of the ocean floor in a particular region. Assume that, at any position on the path (such as the points denoted by '△'), we may steer our sensor to measure the depth of any point within a given region around the current position (as depicted by the dotted ellipses), and that we receive a linear measurement of the depth corrupted by Gaussian noise with variance R. Suppose that we model the depth of the ocean floor as a Gauss-Markov random field with a 500×100 thin membrane grid model where neighboring node attractions are uniformly equal to q. One cycle of the vehicle path takes 300 time steps to complete.

Defining the state X to be the vector containing one element for each cell in the 500×100 grid, the structure of the problem can be seen to be waveform selection: at each time we choose between observations which convey information about different aspects of the same underlying phenomenon.

Fig. 3.2(b) shows the accrual of reward over time as well as the bound on the optimal sequence obtained using Theorem 3.2 for each time step when $q = 100$ and $R = 1/40$, while Fig. 3.2(c) shows the ratio between the achieved performance and the optimal sequence bound over time. The graph indicates that the greedy heuristic achieves at least 0.8× the optimal reward. The tightness of the online bound depends on particular model characteristics: if $q = R = 1$, then the guarantee ratio is much closer to the value of the offline bound (*i.e.*, 0.5). Fig. 3.3 shows snapshots of how the uncertainty in the depth estimate progresses over time. The images display the marginal entropy of each cell in the grid.

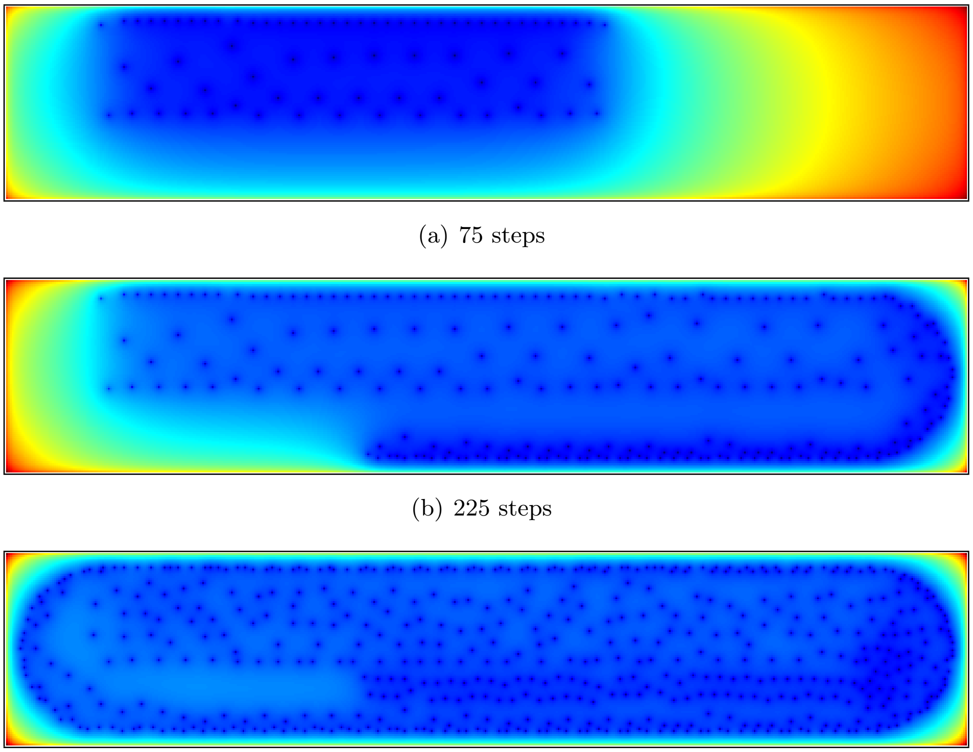


(a) Region boundary and vehicle path

(b) Reward obtained by greedy heuristic and bound on optimal

(c) Factor of optimality from online guarantee

Figure 3.2. (a) shows region boundary and vehicle path (counter-clockwise, starting from the left end of the lower straight segment). When the vehicle is located at a '△' mark, any one grid element with center inside the surrounding dotted ellipse may be measured. (b) graphs reward accrued by the greedy heuristic after different periods of time, and the bound on the optimal sequence for the same time period. (c) shows the ratio of these two curves, providing the factor of optimality guaranteed by the bound.



(a) 75 steps

(b) 225 steps

(c) 525 steps

Figure 3.3. Marginal entropy of each grid cell after 75, 225 and 525 steps. Blue indicates the lowest uncertainty, while red indicates the highest. Vehicle path is clockwise, commencing from top-left. Each revolution takes 300 steps.

3.2 Exploiting diffusiveness

In problems such as object tracking, the kinematic quantities of interest evolve according to a diffusive process, in which correlation between states at different time instants reduces as the time difference increases. Intuitively, one would expect that a greedy algorithm would be closer to optimal in situations in which the diffusion strength is high. This section develops a performance guarantee which exploits the diffusiveness of the underlying process to obtain a tighter bound on performance.

The general form of the result, stated in Theorem 3.3, deals with an arbitrary graph (in the sense of Section 2.1.5) in the latent structure. The simpler cases involving trees and chains are discussed in the sequel. The theorem is limited to only choosing a single observation from each set; the proof of Theorem 3.3 exploits this fact. The basic model structure is set up in Assumption 3.2.

Assumption 3.2. *Let the latent structure which we seek to infer consist of an undirected graph $\mathcal{G}$ with nodes $X = \{x_1, \dots, x_L\}$, with an arbitrary interconnection structure. Assume that each node has a set of observations $\{z_i^1, \dots, z_i^{n_i}\}$, which are independent of each other and all other nodes and observations in the graph conditioned on x_i. We may select a single observation from each set. Let $(w_1, \dots, w_L)$ be a sequence which determines the order in which nodes are visited ($w_i \in \{1, \dots, L\}\ \forall\ i$); we assume that each node is visited exactly once.*

The results of Section 3.1 were applicable to any submodular, non-decreasing objective for which the reward of an empty set was zero. In this section, we exploit an additional property of mutual information which holds under Assumption 3.2, that for any set of conditioning observations $z^{\mathcal{A}}$:

$$\begin{aligned} I(X; z_i^j|z^{\mathcal{A}}) &= H(z_i^j|z^{\mathcal{A}}) - H(z_i^j|X, z^{\mathcal{A}}) \\ &= H(z_i^j|z^{\mathcal{A}}) - H(z_i^j|x_j) \\ &= I(x_i; z_i^j|z^{\mathcal{A}}) \end{aligned} \tag{3.4}$$

We then utilize this property in order to exploit process diffusiveness. The general form of the diffusive characteristic is stated in Assumption 3.3. This is a strong assumption that is difficult to establish globally for any given model; we examine it for one simple model in Section 3.2.3. In Section 3.2.1 we present an online computable guarantee which exploits the characteristic to whatever extent it exists in a particular selection problem. In Section 3.2.2 we then specialize the assumption to cases where the latent graph structure is a tree or a chain.

Assumption 3.3. *Under the structure in Assumption 3.2, let the graph $\mathcal{G}$ have the diffusive property in which there exists $\alpha < 1$ such that for each $i \in \{1, \dots, L\}$ and each observation $z_{w_i}^j$ at node x_{w_i},*

$$I(x_{\mathcal{N}(w_i)}; z_{w_i}^j | z_{w_1}^{g_1}, \dots, z_{w_{i-1}}^{g_{i-1}}) \leq \alpha I(x_{w_i}; z_{w_i}^j | z_{w_1}^{g_1}, \dots, z_{w_{i-1}}^{g_{i-1}})$$

where $x_{\mathcal{N}(w_i)}$ denotes the neighbors of node x_{w_i} in the latent structure graph $\mathcal{G}$.

Assumption 3.3 states that the information which the observation $z_{w_i}^j$ contains about x_{w_i} is discounted by a factor of at least α when compared to the information it contains about the remainder of the graph. Theorem 3.3 uses this property to bound the loss of optimality associated with the greedy choice to be a factor of $(1 + \alpha)$ rather than 2.

Theorem 3.3. *Under Assumptions 3.2 and 3.3, the performance of the greedy heuristic in Definition 3.1 satisfies the following guarantee:*

$$I(X; z_{w_1}^{o_1}, \dots, z_{w_L}^{o_L}) \leq (1 + \alpha) I(X; z_{w_1}^{g_1}, \dots, z_{w_L}^{g_L})$$

Proof. To establish an induction step, assume that (as trivially holds for $j = 1$),

$$I(X; z_{w_1}^{o_1}, \dots, z_{w_L}^{o_L}) \leq (1 + \alpha) I(X; z_{w_1}^{g_1}, \dots, z_{w_{j-1}}^{g_{j-1}}) + I(X; z_{w_j}^{o_j}, \dots, z_{w_L}^{o_L} | z_{w_1}^{g_1}, \dots, z_{w_{j-1}}^{g_{j-1}}) \tag{3.5}$$

Manipulating the second term in Eq. (3.5),

$$\begin{aligned}
&I(X; z_{w_j}^{o_j}, \dots, z_{w_L}^{o_L} | z_{w_1}^{g_1}, \dots, z_{w_{j-1}}^{g_{j-1}}) \\
&\overset{(a)}{=} I(x_{w_j}; z_{w_j}^{o_j} | z_{w_1}^{g_1}, \dots, z_{w_{j-1}}^{g_{j-1}}) + I(x_{w_{j+1}}, \dots, x_{w_L}; z_{w_{j+1}}^{o_{j+1}}, \dots, z_{w_L}^{o_L} | z_{w_1}^{g_1}, \dots, z_{w_{j-1}}^{g_{j-1}}, z_{w_j}^{o_j}) \\
&\overset{(b)}{\leq} I(x_{w_j}; z_{w_j}^{g_j} | z_{w_1}^{g_1}, \dots, z_{w_{j-1}}^{g_{j-1}}) + I(x_{w_{j+1}}, \dots, x_{w_L}; z_{w_{j+1}}^{o_{j+1}}, \dots, z_{w_L}^{o_L} | z_{w_1}^{g_1}, \dots, z_{w_{j-1}}^{g_{j-1}}) \\
&\overset{(c)}{\leq} I(x_{w_j}; z_{w_j}^{g_j} | z_{w_1}^{g_1}, \dots, z_{w_{j-1}}^{g_{j-1}}) + I(x_{w_{j+1}}, \dots, x_{w_L}; z_{w_j}^{g_j}, z_{w_{j+1}}^{o_{j+1}}, \dots, z_{w_L}^{o_L} | z_{w_1}^{g_1}, \dots, z_{w_{j-1}}^{g_{j-1}}) \\
&\overset{(d)}{=} I(x_{w_j}; z_{w_j}^{g_j} | z_{w_1}^{g_1}, \dots, z_{w_{j-1}}^{g_{j-1}}) + I(x_{\mathcal{N}(w_j)}; z_{w_j}^{g_j} | z_{w_1}^{g_1}, \dots, z_{w_{j-1}}^{g_{j-1}}) \\
&\qquad + I(x_{w_{j+1}}, \dots, x_{w_L}; z_{w_{j+1}}^{o_{j+1}}, \dots, z_{w_L}^{o_L} | z_{w_1}^{g_1}, \dots, z_{w_j}^{g_j}) \\
&\overset{(e)}{\leq} (1 + \alpha) I(x_{w_j}; z_{w_j}^{g_j} | z_{w_1}^{g_1}, \dots, z_{w_{j-1}}^{g_{j-1}}) + I(x_{w_{j+1}}, \dots, x_{w_L}; z_{w_{j+1}}^{o_{j+1}}, \dots, z_{w_L}^{o_L} | z_{w_1}^{g_1}, \dots, z_{w_j}^{g_j}) \\
&\overset{(f)}{=} (1 + \alpha) I(X; z_{w_j}^{g_j} | z_{w_1}^{g_1}, \dots, z_{w_{j-1}}^{g_{j-1}}) + I(X; z_{w_{j+1}}^{o_{j+1}}, \dots, z_{w_L}^{o_L} | z_{w_1}^{g_1}, \dots, z_{w_j}^{g_j})
\end{aligned}$$

where (a) and (f) result from the chain rule, from independence of $z_{w_j}^{o_j}$ and $z_{w_j}^{g_j}$ on the remaining latent structure conditioned on x_{w_j}, and from independence of $\{z_{w_{j+1}}^{o_{j+1}}, \dots, z_{w_L}^{o_L}\}$

on the remaining latent structure conditioned on $\{x_{w_{j+1}}, \dots, x_{w_L}\}$; (b) from submodularity and the definition of the greedy heuristic; (c) from the non-decreasing property; (d) from the chain rule (noting that $z_{w_j}^{g_j}$ is independent of all nodes remaining to be visited conditioned on the neighbors of x_{w_j}); and (e) from the assumed diffusive property of Assumption 3.3. Replacing the second term in Eq. (3.5) with the final result in (f), we obtain a strengthened bound:

$$I(X; z_{w_1}^{o_1}, \dots, z_{w_L}^{o_L}) \leq (1+\alpha) I(X; z_{w_1}^{g_1}, \dots, z_{w_j}^{g_j}) + I(X; z_{w_{j+1}}^{o_{j+1}}, \dots, z_{w_L}^{o_L} | z_{w_1}^{g_1}, \dots, z_{w_j}^{g_j})$$

Applying this induction step L times, we obtain the desired result. □

■ 3.2.1 Online guarantee

For many models the diffusive property is difficult to establish globally. Following from step (d) of Theorem 3.3, one may obtain an online computable bound which does not require the property of Assumption 3.3 to hold globally, but exploits it to whatever extent it exists in a particular selection problem.

Theorem 3.4. *Under the model of Assumption 3.2, but not requiring the diffusive property of Assumption 3.3, the following performance guarantee, which can be computed online, applies to the greedy heuristic of Definition 3.1:*

$$I(X; z_{w_1}^{o_1}, \dots, z_{w_L}^{o_L}) \leq I(X; z_{w_1}^{g_1}, \dots, z_{w_L}^{g_L}) + \sum_{j=1}^{L} I(x_{\mathcal{N}(w_j)}; z_{w_j}^{g_j} | z_{w_1}^{g_1}, \dots, z_{w_{j-1}}^{g_{j-1}})$$

Proof. The proof directly follows Theorem 3.3. Commence by assuming (for induction) that:

$$I(X; z_{w_1}^{o_1}, \dots, z_{w_L}^{o_L}) \leq I(X; z_{w_1}^{g_1}, \dots, z_{w_{j-1}}^{g_{j-1}}) + \sum_{i=1}^{j-1} I(x_{\mathcal{N}(w_i)}; z_{w_i}^{g_i} | z_{w_1}^{g_1}, \dots, z_{w_{i-1}}^{g_{i-1}}) + I(X; z_{w_j}^{o_j}, \dots, z_{w_L}^{o_L} | z_{w_1}^{g_1}, \dots, z_{w_{j-1}}^{g_{j-1}}) \quad (3.6)$$

Eq. (3.6) trivially holds for $j = 1$. If we assume that it holds for j then step (d) of Theorem 3.3 obtains an upper bound to the the final term in Eq. (3.6) which establishes that Eq. (3.6) also holds for $(j+1)$. Applying the induction step L times, we obtain the desired result. □

■ 3.2.2 Specialization to trees and chains

In the common case where the latent structure $X = \{x_1, \dots, x_L\}$ forms a tree, we may avoid including *all* neighbors a node in the condition of Assumption 3.3 and in the result of Theorem 3.4. The modified assumptions are presented below. An additional requirement on the sequence $(w_1, \dots, w_L)$ is necessary to exploit the tree structure.

Assumption 3.4. *Let the latent structure which we seek to infer consist of an undirected graph $\mathcal{G}$ with nodes $X = \{x_1, \dots, x_L\}$, which form a tree. Assume that each node has a set of observations $\{z_i^1, \dots, z_i^{n_i}\}$, which are independent of each other and all other nodes and observations in the graph conditioned on x_i. We may select a single observation from each set. Let $(w_1, \dots, w_L)$ be a sequence which determines the order in which nodes are visited ($w_i \in \{1, \dots, L\}$ $\forall$ i); we assume that each node is visited exactly once. We assume that the sequence is "bottom-up", i.e., that no node is visited before all of its children have been visited.*

Assumption 3.5. *Under the structure in Assumption 3.4, let the graph $\mathcal{G}$ have the diffusive property in which there exists $\alpha < 1$ such that for each $i \in \{1, \dots, L\}$ and each observation $z_{w_i}^j$ at node x_{w_i},*

$$I(x_{\pi(w_i)}; z_{w_i}^j | z_{w_1}^{g_1}, \dots, z_{w_{i-1}}^{g_{i-1}}) \leq \alpha I(x_{w_i}; z_{w_i}^j | z_{w_1}^{g_1}, \dots, z_{w_{i-1}}^{g_{i-1}})$$

where $x_{\pi(w_i)}$ denotes the parent of node x_{w_i} in the latent structure graph $\mathcal{G}$.

Theorem 3.3 holds under Assumptions 3.4 and 3.5; the proof passes directly once $x_{\mathcal{N}(w_j)}$ is replaced by $x_{\pi(w_j)}$ in step (*d*). The modified statement of Theorem 3.4 is included below. Again, the proof passes directly once $x_{\mathcal{N}(w_i)}$ is replaced by $x_{\pi(w_i)}$.

Theorem 3.5. *Under the model of Assumption 3.4, but not requiring the diffusive property of Assumption 3.5, the following performance guarantee, which can be computed online, applies to the greedy heuristic of Definition 3.1:*

$$I(X; z_{w_1}^{o_1}, \dots, z_{w_L}^{o_L}) \leq I(X; z_{w_1}^{g_1}, \dots, z_{w_L}^{g_L}) + \sum_{j=1}^{L} I(x_{\pi(w_j)}; z_{w_j}^{g_j} | z_{w_1}^{g_1}, \dots, z_{w_{j-1}}^{g_{j-1}})$$

The most common application of the diffusive model is in Markov chains (a special case of a tree), where the i-th node corresponds to time i. In this case, the sequence is simply $w_i = i$, *i.e.*, we visit the nodes in time order. Choosing the final node in the chain to be the tree root, this sequence respects the bottom-up requirement, and the diffusive requirement becomes:

$$I(x_{k+1}; z_k^j | z_1^{g_1}, \dots, z_{k-1}^{g_{k-1}}) \leq \alpha I(x_k; z_k^j | z_1^{g_1}, \dots, z_{k-1}^{g_{k-1}}) \tag{3.7}$$

3.2.3 Establishing the diffusive property

As an example, we establish the diffusive property for a simple one-dimensional stationary linear Gauss-Markov chain model. The performance guarantee is uninteresting in this case since the greedy heuristic may be easily shown to be optimal. Nevertheless, some intuition may be gained from the structure of the condition which results. The dynamics model and observation models are given by:

$$x_{k+1} = f x_k + w_k \tag{3.8}$$

$$z_k^j = h^j x_k + v_k^j, \ j \in \{1, \dots, n\} \tag{3.9}$$

where $w_k \sim \mathcal{N}\{w_k; 0, q\}$ and $v_k^j \sim \mathcal{N}\{v_k^j; 0, r^j\}$. We let $\tilde{q} = q/f^2$ and $\tilde{r}^j = r^j/(h^j)^2$. The greedy heuristic in this model corresponds to choosing the observation z_k^j with the smallest normalized variance $\tilde{r}^j$. Denoting the covariance of x_k conditioned on the prior observations as $P_{k|k-1}$, the terms involved in Eq. (3.7) can be evaluated as:

$$I(x_k; z_k^j | z_1^{g_1}, \dots, z_{k-1}^{g_{k-1}}) = \frac{1}{2} \log \left[1 + \frac{P_{k|k-1}}{\tilde{r}^j} \right] \tag{3.10}$$

$$I(x_{k+1}; z_k^j | z_1^{g_1}, \dots, z_{k-1}^{g_{k-1}}) = \frac{1}{2} \log \left[1 + \frac{P_{k|k-1}^2}{(\tilde{r}^j + \tilde{q}) P_{k|k-1} + \tilde{r}^j \tilde{q}} \right] \tag{3.11}$$

If $P_{k|k-1}$ can take on any value on the positive real line then no $\alpha < 1$ exists, since:

$$\lim_{P \to \infty} \left\{ \frac{\frac{1}{2} \log \left[1 + \frac{P^2}{(\tilde{r}^j + \tilde{q})P + \tilde{r}^j \tilde{q}} \right]}{\frac{1}{2} \log \left[1 + \frac{P}{\tilde{r}^j} \right]} \right\} = 1$$

Thus we seek a range for $P_{k|k-1}$ such that there does exist an $\alpha < 1$ for which Eq. (3.7) is satisfied. If such a result is obtained, then the diffusive property is established as long as the covariance remains within this range during operation.

Substituting Eq. (3.10) and Eq. (3.11) into Eq. (3.7) and exponentiating each side, we need to find the range of P for which

$$b_\alpha(P) = \frac{1 + \frac{P^2}{(\tilde{r}^j + \tilde{q})P + \tilde{r}^j \tilde{q}}}{\left[1 + \frac{P}{\tilde{r}^j} \right]^\alpha} \leq 1 \tag{3.12}$$

Note that $b_\alpha(0) = 1$ for any α. Furthermore, $\frac{d}{dP} b_\alpha(P)$ may be shown to be negative for $P \in [0, a)$ and positive for $P \in (a, \infty)$ (for some positive a). Hence $b_\alpha(P)$ reduces from a value of unity initially before increasing monotonically and eventually crossing unity. For any given α, there will be a unique non-zero value of P for which $b_\alpha(P) = 1$. For a given value of P, we can easily solve Eq. (3.12) to find the smallest value of α for which the expression is satisfied:

$$\alpha^*(P) = \frac{\log\left[\frac{(P+\tilde{r}^j)(P+\tilde{q})}{(\tilde{r}^j+\tilde{q})P+\tilde{r}^j\tilde{q}}\right]}{\log(P+\tilde{r}^j)} \tag{3.13}$$

Hence, for any $P \in [0, P_0]$, Eq. (3.12) is satisfied for any $\alpha \in [\alpha^*(P_0), 1]$. The strongest diffusion coefficient is shown in Fig. 3.4 as a function of the covariance upper limit P_0 for various values of $\tilde{r}$ and $\tilde{q}$. Different values of the dynamics model parameter f will yield different steady state covariances, and hence select different operating points on the curve.

While closed-form analysis is very difficult for multidimensional linear Gaussian systems, nor for nonlinear and/or non-Gaussian systems, the general intuition of the single dimensional linear Gaussian case may be applied. For example, many systems will satisfy some degree of diffusiveness as long as all states remain within some certainly level. The examples in Sections 3.2.4 and 3.2.5 demonstrate the use of the online performance guarantee in cases where the diffusive condition has not been established globally.

■ 3.2.4 Example: beam steering revisited

Consider the beam steering scenario presented in Section 3.1.4. The performance bound obtained using the online analysis form Theorem 3.5 is shown in Fig. 3.5. As expected, the bound tightens as the diffusion strength increases. In this example, position states are directly observable, while velocity states are only observable through the induced impact on position. The guarantee is substantially tighter when all states are directly observable, as shown in the following example.

■ 3.2.5 Example: bearings only measurements

Consider an object which moves in two dimensions according to a Gaussian random walk:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{w}_k$$

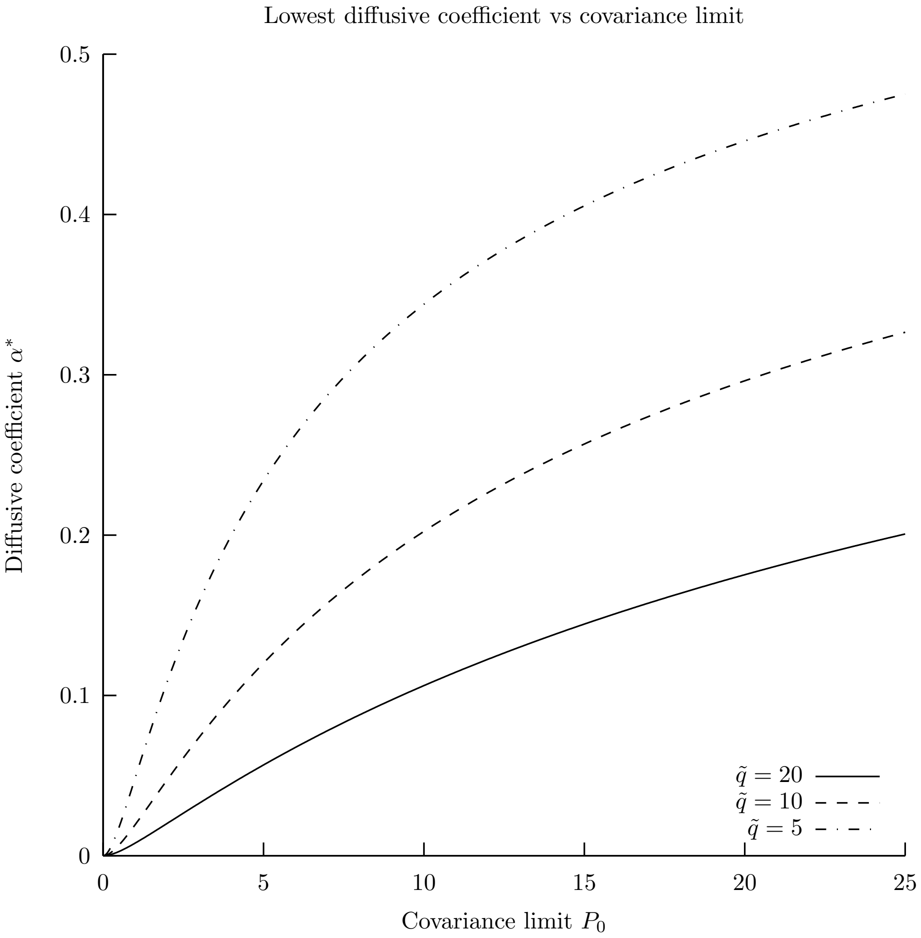
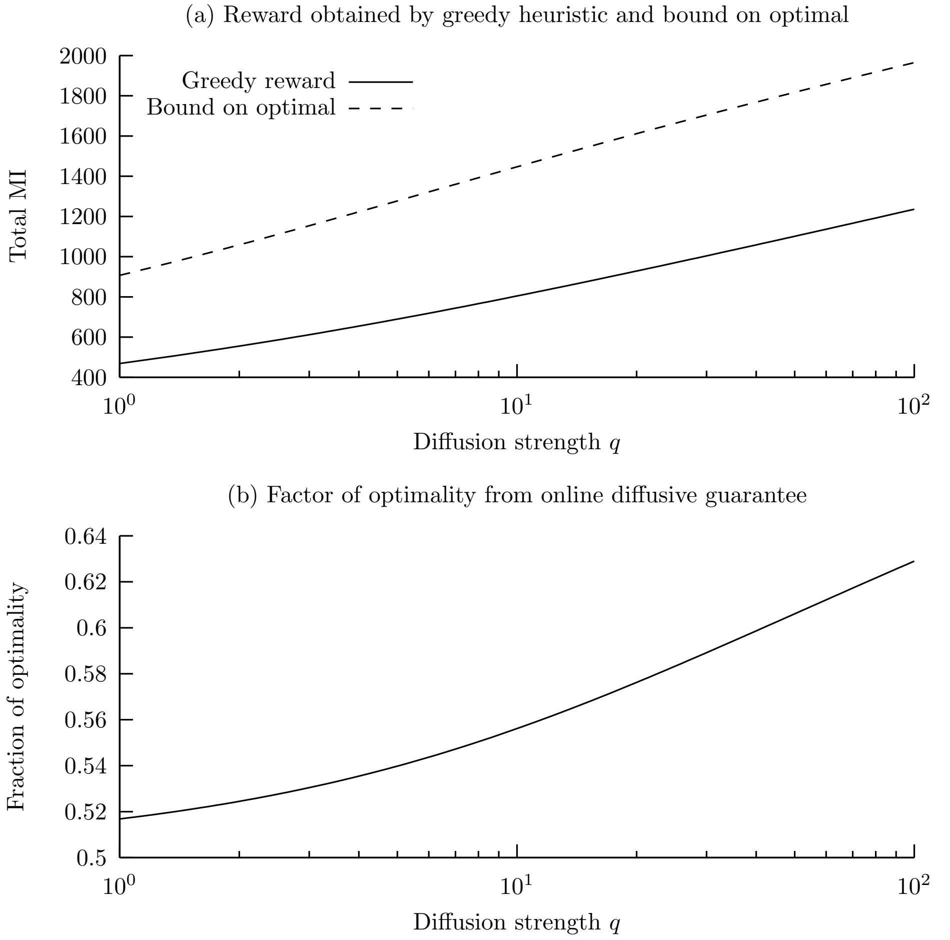


Figure 3.4. Strongest diffusive coefficient versus covariance upper limit for various values of $\tilde{q}$, with $\tilde{r} = 1$. Note that lower values of α^* correspond to stronger diffusion.



(a) Reward obtained by greedy heuristic and bound on optimal

(b) Factor of optimality from online diffusive guarantee

Figure 3.5. (a) shows total reward accrued by the greedy heuristic in the 200 time steps for different diffusion strength values (q), and the bound on optimal obtained through Theorem 3.5. (b) shows the ratio of these curves, providing the factor of optimality guaranteed by the bound.

where $\boldsymbol{w}_k \sim \mathcal{N}\{\boldsymbol{w}_k; \mathbf{0}, \mathbf{I}\}$. The initial position of the object is distributed according to $\boldsymbol{x}_0 \sim \mathcal{N}\{\boldsymbol{x}_0; \mathbf{0}, \mathbf{I}\}$. Assume that bearing observations are available from four sensors positioned at $(\pm 100, \pm 100)$, but that only one observation may be utilized at any instant. Simulations were run for 200 time steps. The total reward and the bound obtained from Theorem 3.5 are shown in Fig. 3.6(a) as a function of the measurement noise standard deviation (in degrees). The results demonstrate that the performance guarantee becomes stronger as the measurement noise decreases; the same effect occurs if the observation noise is held constant and the dynamics noise increased. Fig. 3.6(b) shows the ratio of the greedy performance to the upper bound on optimal, demonstrating that the greedy heuristic is guaranteed to be within a factor of 0.77 of optimal with a measurement standard deviation of 0.1 degrees.

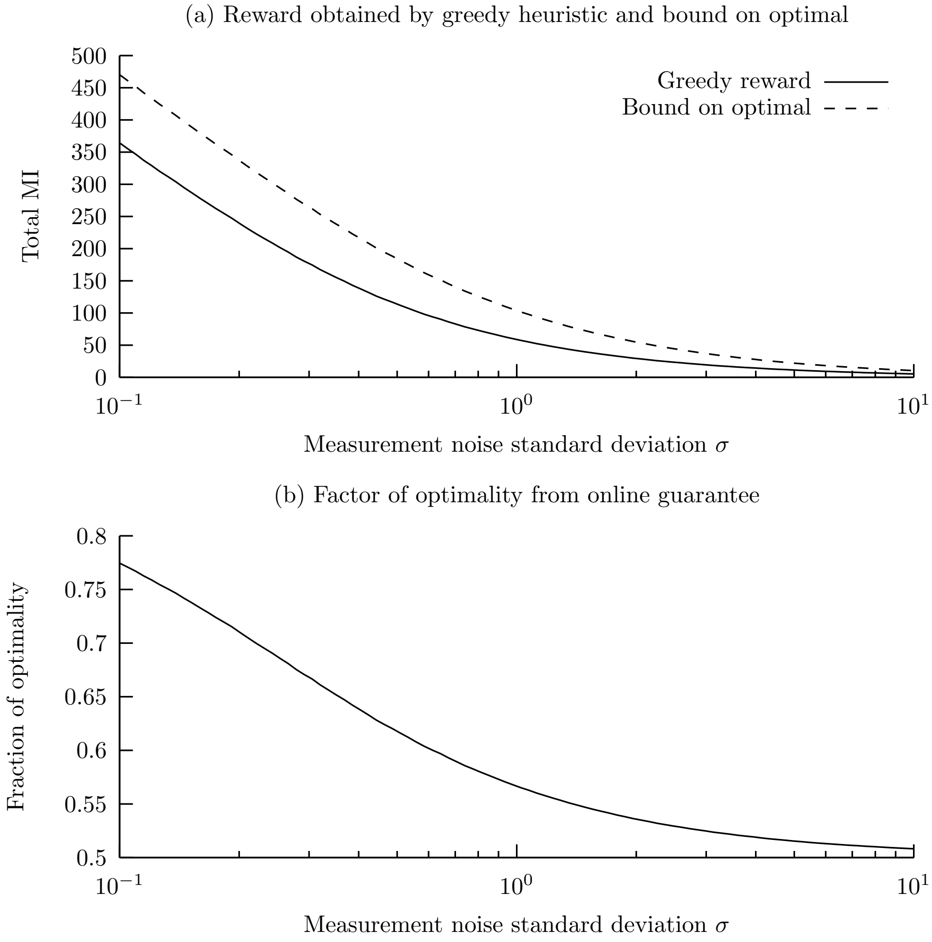
In this example, we utilized the closed loop greedy heuristic examined in Section 3.5, hence it was necessary to use multiple Monte Carlo simulations to compute the online guarantee. Tracking was performed using an extended Kalman filter, hence the bounds are approximate (the EKF variances were used to calculate the rewards). In this scenario, the low degree of nonlinearity in the observation model provides confidence that the inaccuracy in the rewards is insignificant.

3.3 Discounted rewards

In Sections 3.1 and 3.2 the objective we were seeking to optimize was the mutual information between the state and observations through the planning horizon, and the optimal open loop reward to which we compared was $I(X; z_{w_1}^{o_1}, \dots, z_{w_M}^{o_M})$. In some sequential estimation problems, it is not only desirable to maximize the information obtained within a particular period of time, but also how quickly, within the planning horizon, the information is obtained. One way[2] of capturing this notion is to incorporate a discount factor in the objective, reducing the value of information obtained later in the problem. Subsequently, our optimization problem is changed from:

$$\min_{u_1,\dots,u_M} I(X; z_1^{u_1}, \dots, z_N^{u_M}) = \min_{u_1,\dots,u_M} \sum_{k=1}^{M} I(X; z_k^{u_k} | z_1^{u_1}, \dots, z_{k-1}^{u_{k-1}})$$

[2] Perhaps the most natural way of capturing this would be to reformulate the problem, choosing as the objective (to be minimized) the expected time required to reduce the uncertainty below a desired criterion. The resulting problem is intractable, hence the approximate method using MI is an appealing substitute.



(a) Reward obtained by greedy heuristic and bound on optimal

(b) Factor of optimality from online guarantee

Figure 3.6. (a) shows average total reward accrued by the greedy heuristic in the 200 time steps for different diffusion strength values (q), and the bound on optimal obtained through Theorem 3.5. (b) shows the ratio of these curves, providing the factor of optimality guaranteed by the bound.

to:

$$\min_{u_1,\ldots,u_M} \sum_{k=1}^{M} \alpha^{k-1} I(X; z_k^{u_k} | z_1^{u_1}, \ldots, z_{k-1}^{u_{k-1}})$$

where $\alpha < 1$. Not surprising, the performance guarantee for the greedy heuristic becomes tighter as the discount factor is decreased, as Theorem 3.6 establishes. We define the abbreviated notation for the optimal reward to go from stage i to the end of the problem conditioned on the previous observation choices $(u_1, \ldots, u_k)$:

$$J_i^o[(u_1, \ldots, u_k)] \triangleq \sum_{j=i}^{M} \alpha^{j-1} I(X; z_{w_j}^{o_j} | z_{w_1}^{u_1}, \ldots, z_{w_k}^{u_k}, z_{w_i}^{o_i}, \ldots, z_{w_{j-1}}^{o_{j-1}})$$

and the reward so far for the greedy heuristic in i stages:

$$J_{\to i}^g \triangleq \sum_{j=1}^{i} \alpha^{j-1} I(X; z_{w_j}^{g_j} | z_{w_1}^{g_1}, \ldots, z_{w_{j-1}}^{g_{j-1}})$$

We will use the following lemma in the proof of Theorem 3.6.

Lemma 3.1. *The optimal reward to go for the discounted sequence satisfies the relationship:*

$$J_{i+1}^o[(g_1, \ldots, g_{i-1})] \leq \alpha^i I(X; z_{w_i}^{g_i} | z_{w_1}^{g_1}, \ldots, z_{w_{i-1}}^{g_{i-1}}) + J_{i+1}^o[(g_1, \ldots, g_i)]$$

Proof. Expanding the left-hand side and manipulating:

$$\begin{aligned}
J^o_{i+1}[(g_1,\dots,g_{i-1})] &= \sum_{j=i+1}^{M} \alpha^{j-1} I(X; z^{o_j}_{w_j} | z^{g_1}_{w_1},\dots,z^{g_{i-1}}_{w_{i-1}}, z^{o_{i+1}}_{w_{i+1}},\dots,z^{o_{j-1}}_{w_{j-1}}) \\
&\overset{(a)}{=} \alpha^i \Bigg[I(X; z^{o_{i+1}}_{w_{i+1}},\dots,z^{o_M}_{w_M} | z^{g_1}_{w_1},\dots,z^{g_{i-1}}_{w_{i-1}}) \\
&\qquad + \sum_{j=i+1}^{M} (\alpha^{j-i-1}-1) I(X; z^{o_j}_{w_j} | z^{g_1}_{w_1},\dots,z^{g_{i-1}}_{w_{i-1}}, z^{o_{i+1}}_{w_{i+1}},\dots,z^{o_{j-1}}_{w_{j-1}}) \Bigg] \\
&\overset{(b)}{\le} \alpha^i \Bigg[I(X; z^{g_i}_{w_i}, z^{o_{i+1}}_{w_{i+1}},\dots,z^{o_M}_{w_M} | z^{g_1}_{w_1},\dots,z^{g_{i-1}}_{w_{i-1}}) \\
&\qquad + \sum_{j=i+1}^{M} (\alpha^{j-i-1}-1) I(X; z^{o_j}_{w_j} | z^{g_1}_{w_1},\dots,z^{g_{i-1}}_{w_{i-1}}, z^{o_{i+1}}_{w_{i+1}},\dots,z^{o_{j-1}}_{w_{j-1}}) \Bigg] \\
&\overset{(c)}{=} \alpha^i \Bigg[I(X; z^{g_i}_{w_i} | z^{g_1}_{w_1},\dots,z^{g_{i-1}}_{w_{i-1}}) + I(X; z^{o_{i+1}}_{w_{i+1}},\dots,z^{o_M}_{w_M} | z^{g_1}_{w_1},\dots,z^{g_i}_{w_i}) \\
&\qquad + \sum_{j=i+1}^{M} (\alpha^{j-i-1}-1) I(X; z^{o_j}_{w_j} | z^{g_1}_{w_1},\dots,z^{g_{i-1}}_{w_{i-1}}, z^{o_{i+1}}_{w_{i+1}},\dots,z^{o_{j-1}}_{w_{j-1}}) \Bigg] \\
&\overset{(d)}{\le} \alpha^i \Bigg[I(X; z^{g_i}_{w_i} | z^{g_1}_{w_1},\dots,z^{g_{i-1}}_{w_{i-1}}) + I(X; z^{o_{i+1}}_{w_{i+1}},\dots,z^{o_M}_{w_M} | z^{g_1}_{w_1},\dots,z^{g_i}_{w_i}) \\
&\qquad + \sum_{j=i+1}^{M} (\alpha^{j-i-1}-1) I(X; z^{o_j}_{w_j} | z^{g_1}_{w_1},\dots,z^{g_i}_{w_i}, z^{o_{i+1}}_{w_{i+1}},\dots,z^{o_{j-1}}_{w_{j-1}}) \Bigg] \\
&\overset{(e)}{=} \alpha^i I(X; z^{g_i}_{w_i} | z^{g_1}_{w_1},\dots,z^{g_{i-1}}_{w_{i-1}}) + J^o_{i+1}[(g_1,\dots,g_i)]
\end{aligned}$$

where (a) results from adding and subtracting $\alpha^i I(X; z^{o_{i+1}}_{w_{i+1}},\dots,z^{o_M}_{w_M} | z^{g_1}_{w_1},\dots,z^{g_{i-1}}_{w_{i-1}})$, (b) from the non-decreasing property of MI (introducing the additional observation $z^{g_i}_{w_i}$ into the first term), (c) from the MI chain rule, (d) from submodularity (adding the conditioning $z^{g_i}_{w_i}$ into the second term, noting that the coefficient is negative), and (e) from cancelling similar terms and applying the definition of J^o_{i+1}. □

Theorem 3.6. *Under Assumption 3.1, the greedy heuristic in Definition 3.1 has performance guaranteed by the following expression:*

$$\sum_{j=1}^{M} \alpha^{j-1} I(X; z^{o_j}_{w_j} | z^{o_1}_{w_1},\dots,z^{o_{j-1}}_{w_{j-1}}) \le (1+\alpha) \sum_{j=1}^{M} \alpha^{j-1} I(X; z^{g_j}_{w_j} | z^{g_1}_{w_1},\dots,z^{g_{j-1}}_{w_{j-1}})$$

where $\{z_{w_1}^{o_1}, \dots, z_{w_M}^{o_M}\}$ *is the optimal set of observations, i.e., the one which maximizes*

$$\sum_{j=1}^{M} \alpha^{j-1} I(X; z_{w_j}^{o_j} | z_{w_1}^{o_1}, \dots, z_{w_{j-1}}^{o_{j-1}})$$

Proof. In the abbreviated notation, we seek to prove:

$$J_1^o[\emptyset] \leq (1+\alpha) J_{\to M}^g$$

The proof follows an induction on the expression

$$J_1^o[\emptyset] \leq (1+\alpha) J_{\to i-1}^g + J_i^o[(g_1, \dots, g_{i-1})]$$

which is trivially true for $i = 1$. Suppose it is true for i; manipulating the expression we obtain:

$$\begin{aligned}
J_1^o[\emptyset] &\leq (1+\alpha) J_{\to i-1}^g + J_i^o[(g_1, \dots, g_{i-1})] \\
&\overset{(a)}{=} (1+\alpha) J_{\to i-1}^g + \alpha^{i-1} I(X; z_{w_i}^{o_i} | z_{w_1}^{g_1}, \dots, z_{w_{i-1}}^{g_{i-1}}) + J_{i+1}^o[(g_1, \dots, g_{i-1}, o_i)] \\
&\overset{(b)}{\leq} (1+\alpha) J_{\to i-1}^g + \alpha^{i-1} I(X; z_{w_i}^{g_i} | z_{w_1}^{g_1}, \dots, z_{w_{i-1}}^{g_{i-1}}) + J_{i+1}^o[(g_1, \dots, g_{i-1}, o_i)] \\
&\overset{(c)}{\leq} (1+\alpha) J_{\to i-1}^g + \alpha^{i-1} I(X; z_{w_i}^{g_i} | z_{w_1}^{g_1}, \dots, z_{w_{i-1}}^{g_{i-1}}) + J_{i+1}^o[(g_1, \dots, g_{i-1})] \\
&\overset{(d)}{\leq} (1+\alpha) J_{\to i-1}^g + (1+\alpha)\alpha^{i-1} I(X; z_{w_i}^{g_i} | z_{w_1}^{g_1}, \dots, z_{w_{i-1}}^{g_{i-1}}) + J_{i+1}^o[(g_1, \dots, g_i)] \\
&= (1+\alpha) J_{\to i}^g + J_{i+1}^o[(g_1, \dots, g_i)]
\end{aligned}$$

where (a) results from the definition of J_i^o, (b) results from the definition of the greedy heuristic, (c) results from submodularity (allowing us to remove the conditioning on o_i from each term in J_{i+1}^o) and (d) from Lemma 3.1.

Applying the induction step M times we get the desired result. □

■ 3.4 Time invariant rewards

The reward function in some problems is well-approximated as being time invariant. In this case, a tighter bound, $(1 - 1/e)\times$, may be obtained. The structure necessary is given in Assumption 3.6.

Assumption 3.6. *At each stage $k \in \{1, \dots, L\}$ there are n observations, $\{z_k^1, \dots, z_k^n\}$, which are mutually independent conditioned on the quantity to be estimated (X). At each stage k we may select any one of the n observations; for each observation i, the observation model $p(z_k^i|X)$ does not change with stage. Selecting the same observation in m different stages results in m conditionally independent observations based on the same model.*

Note that the only difference between this problem and the k-element subset selection problem in Section 2.4.5 is that the same observation can be chosen multiple times in different stages to yield additional information through the same observation model (*e.g.*, effectively providing a reduced variance observation). The following proof obtains the guarantee by transforming the problem into a k-element subset selection problem.

Theorem 3.7. *Under Assumption 3.6, the greedy heuristic in Definition 3.1 has performance guaranteed by the following expression:*

$$(1 - 1/e) I(X; z_1^{o_1}, \dots, z_L^{o_L}) \leq I(X; z_1^{g_1}, \dots, z_L^{g_L})$$

where $\{z_1^{o_1}, \dots, z_L^{o_L}\}$ is the optimal set of observations, i.e., the one which maximizes $I(X; z_1^{o_1}, \dots, z_L^{o_L})$.

Proof. Consider the L-element subset selection problem involving the set of observations

$$\{z^{1,1}, \dots, z^{1,L}, \dots, z^{n,1}, \dots, z^{n,L}\}$$

For all (i, j), the observation model for $z^{i,j}$ is $p_{z_k^i}(z^{i,j}|X)$, *i.e.*, the same as that of z_k^i. Note that any choice of observations maps directly in to a choice in the original problem. For example, if three observations are chosen out of the set $\{z^{i,1}, \dots, z^{i,n}\}$, then we select the observation z_k^i for three time slots k out of the set $\{1, \dots, L\}$. The exact choice of time slots is not important since rewards are invariant to time.

Thus the solution of the L-element subset selection problem, which is within a factor of $(1 - 1/e)\times$ the optimal solution of the L-element subset selection problem by Theorem 2.4, provides a solution that is within $(1 - 1/e)\times$ the optimal solution of the problem structure in Assumption 3.6. □

Example 3.1 relied on non-stationary observation models to demonstrate that the possible difference between the greedy and optimal selections. The following example illustrates the difference that is possible using stationary models and a static state.

Example 3.2. *Suppose that our state consists of four independent binary random variables, $X = [a\ b\ c\ d]^T$, where $H(a) = H(b) = 1$, and $H(c) = H(d) = 1 - \epsilon$ for some small $\epsilon > 0$. In each stage $k \in \{1, 2\}$ there are three observations available, $z_k^1 = [a\ b]^T$, $z_k^2 = [a\ c]^T$ and $z_k^3 = [b\ d]^T$.*

In stage 1, the greedy heuristic selects observation z_1^1 since $I(X; z_1^1) = 2$ whereas $I(X; z_1^2) = I(X; z_1^3) = 2 - \epsilon$. In stage 2, the algorithm has already learned the values of a and b, hence $I(X; z_2^1|z_1^1) = 0$, and $I(X; z_2^2|z_1^1) = I(X; z_2^3|z_1^1) = 1 - \epsilon$. The total reward is $3 - \epsilon$.

An optimal choice is z_1^2 and z_2^3, achieving reward $4 - 2\epsilon$. The ratio of the greedy reward to optimal reward is

$$\frac{3 - \epsilon}{4 - 2\epsilon}$$

which approaches 0.75 as $\epsilon \to 0$. Examining Theorem 2.4, we see that the performance of the greedy heuristic over $K = 2$ stages is guaranteed to be within a factor of $[1 - (1 - 1/K)^K] = 0.75$ of the optimal, hence this factor is the worst possible over two stages.

The intuition behind the scenario in this example is that information about different portions of the state can be obtained in different combinations, therefore it is necessary to use additional planning to ensure that the observations we obtain provide complementary information.

■ 3.5 Closed loop control

The analysis in Sections 3.1-3.3 concentrates on an open loop control structure, *i.e.*, it assumes that all observation choices are made before any observation values are received. Greedy heuristics are often applied in a closed loop setting, in which an observation is chosen, and then its value is received before the next choice is made.

The performance guarantees of Theorems 3.1 and 3.3 both apply to the *expected* performance of the greedy heuristic operating in a closed loop fashion, *i.e.*, in expectation the closed loop greedy heuristic achieves at least half the reward of the optimal *open* loop selection. The expectation operation is necessary in the closed loop case since control choices are random variables that depend on the values of previous observations. Theorem 3.8 establishes the result of Theorem 3.1 for the closed loop heuristic. The same process can be used to establish a closed loop version of Theorem 3.3. To obtain the closed loop guarantee, we need to exploit an additional characteristic of mutual information:

$$I(X; z^{\mathcal{A}}|z^{\mathcal{B}}) = \int I(X; z^{\mathcal{A}}|z^{\mathcal{B}} = \zeta) p_{z^{\mathcal{B}}}(\zeta) \mathrm{d}\zeta \tag{3.14}$$

While the results are presented in terms of mutual information, they apply to any other objective which meets the previous requirements as well as Eq. (3.14).

We define $\hbar_j = (u_1, z_{w_1}^{u_1}, u_2, z_{w_2}^{u_2}, \dots, u_{j-1}, z_{w_{j-1}}^{u_{j-1}})$ to be the history of all observation actions chosen, and the resulting observation values, prior to stage j (this constitutes all the information which we can utilize in choosing our action at time j). Accordingly, $\hbar_1 = \emptyset$, and $\hbar_j = (\hbar_{j-1}, u_j, z_{w_j}^{u_j})$. The greedy heuristic operating in closed loop is defined in Definition 3.2.

Definition 3.2. *Under the same assumptions as Theorem 3.1, define the closed loop greedy heuristic policy μ^g:*

$$\mu_j^g(\hbar_j) = \underset{u \in \{1,\dots,n_{w_j}\}}{\arg\max}\ I(X; z_{w_j}^u | \hbar_j) \tag{3.15}$$

We use the convention that conditioning on $\hbar_i$ in an MI expression is always on the *value*, and hence if $\hbar_i$ contains elements which are random variables we will always include an explicit expectation operator. The expected reward to go from stage j to the end of the planning horizon for the greedy heuristic $\mu_j^g(\hbar_j)$ commencing from the history $\hbar_j$ is denoted as:

$$\begin{aligned} J_j^{\mu^g}(\hbar_j) &= I(X; z_{w_j}^{\mu_j^g(\hbar_j)}, \dots, z_{w_N}^{\mu_N^g(\hbar_N)} | \hbar_j) && (3.16) \\ &= \mathrm{E}\left[\sum_{i=j}^{N} I(X; z_{w_i}^{\mu_i^g(\hbar_i)} | \hbar_i) \middle| \hbar_j \right] && (3.17) \end{aligned}$$

The expectation in Eq. (3.17) is over the random variables corresponding to the actions $\{\mu_{j+1}^g(\hbar_{j+1}), \dots, \mu_N^g(\hbar_N)\}$,[3] along with the observations resulting from the actions, $\{z_{w_j}^{\mu_j^g(\hbar_j)}, \dots, z_{w_N}^{\mu_N^g(\hbar_N)}\}$, where $\hbar_i$ is the concatenation of the previous history sequence $\hbar_{i-1}$ with the new observation action $\mu_i^g(\hbar_i)$ and the new observation value $z_{w_i}^{\mu_i^g(\hbar_i)}$. The expected reward of the greedy heuristic over the full planning horizon is $J_1^{\mu^g}(\emptyset)$. We also define the expected reward accrued by the greedy heuristic up to and including stage j, commencing from an empty history sequence (*i.e.*, $\hbar_1 = \emptyset$), as:

$$J_{\to j}^{\mu^g} = \mathrm{E}\left[\sum_{i=1}^{j} I(X; z_{w_i}^{\mu_i^g(\hbar_i)} | \hbar_i) \right] \tag{3.18}$$

This gives rise to the recursive relationship:

$$J_{\to j}^{\mu^g} = \mathrm{E}[I(X; z_{w_j}^{\mu_j^g(\hbar_j)} | \hbar_j)] + J_{\to j-1}^{\mu^g} \tag{3.19}$$

[3] We assume a deterministic policy, hence the action at stage j is fixed given knowledge of $\hbar_j$.

Comparing Eq. (3.17) with Eq. (3.18), we have $J^{\mu^g}_{\to N} = J^{\mu^g}_1(\emptyset)$. We define $J^{\mu^g}_{\to 0} = 0$.

The reward of the tail of the optimal open loop observation sequence $(o_j, \dots, o_N)$ commencing from the history $\hbar_j$ is denoted by:

$$J^o_j(\hbar_j) = I(X; z^{o_j}_{w_j}, \dots, z^{o_N}_{w_N} | \hbar_j) \tag{3.20}$$

Using the MI chain rule and Eq. (3.14), this can be written recursively as:

$$J^o_j(\hbar_j) = I(X; z^{o_j}_{w_j} | \hbar_j) + \mathop{\mathrm{E}}_{z^{o_j}_{w_j}|\hbar_j} J^o_{j+1}[(\hbar_j, o_j, z^{o_j}_{w_j})] \tag{3.21}$$

where $J^o_{N+1}(\hbar_{N+1}) = 0$. The reward of the optimal open loop observation sequence over the full planning horizon is:

$$J^o_1(\emptyset) = I(X; z^{o_1}_{w_1}, \dots, z^{o_N}_{w_N}) \tag{3.22}$$

We seek to obtain a guarantee on the performance ratio between the optimal open loop observation sequence and the closed loop greedy heuristic. Before we prove the theorem, we establish a simple result in terms of our new notation.

Lemma 3.2. *Given the above definitions:*

$$\mathop{\mathrm{E}}_{z^{o_j}_{w_j}|\hbar_j} J^o_{j+1}[(\hbar_j, o_j, z^{o_j}_{w_j})] \le J^o_{j+1}(\hbar_j) \le I(X; z^{\mu^g_j(\hbar_j)}_{w_j} | \hbar_j) + \mathop{\mathrm{E}}_{z^{\mu^g_j(\hbar_j)}_{w_j}|\hbar_j} J^o_{j+1}[(\hbar_j, \mu^g_j(\hbar_j), z^{\mu^g_j(\hbar_j)}_{w_j})]$$

Proof. Using Eq. (3.20), the first inequality corresponds to:

$$I(X; z^{o_{j+1}}_{w_{j+1}}, \dots, z^{o_N}_{w_N} | \hbar_j, z^{o_j}_{w_j}) \le I(X; z^{o_{j+1}}_{w_{j+1}}, \dots, z^{o_N}_{w_N} | \hbar_j)$$

where conditioning is on the value $\hbar_j$ throughout (as per the convention introduced below Eq. (3.15)), and on the random variable $z^{o_j}_{w_j}$. Therefore, the first inequality results directly from submodularity.

The second inequality results from the non-decreasing property of MI:

$$\begin{aligned}
I(X; z^{o_{j+1}}_{w_{j+1}}, \dots, z^{o_N}_{w_N} | \hbar_j) &\overset{(a)}{\le} I(X; z^{\mu^g_j(\hbar_j)}_{w_j}, z^{o_{j+1}}_{w_{j+1}}, \dots, z^{o_N}_{w_N} | \hbar_j) \\
&\overset{(b)}{=} I(X; z^{\mu^g_j(\hbar_j)}_{w_j} | \hbar_j) + I(z^{o_{j+1}}_{w_{j+1}}, \dots, z^{o_N}_{w_N} | \hbar_j, z^{\mu^g_j(\hbar_j)}_{w_j}) \\
&\overset{(c)}{=} I(X; z^{\mu^g_j(\hbar_j)}_{w_j} | \hbar_j) + \mathop{\mathrm{E}}_{z^{\mu^g_j(\hbar_j)}_{w_j}|\hbar_j} J^o_{j+1}[(\hbar_j, \mu^g_j(\hbar_j), z^{\mu^g_j(\hbar_j)}_{w_j})]
\end{aligned}$$

(a) results from the non-decreasing property, (b) from the chain rule, and (c) from the definition in Eq. (3.20). □

We are now ready to prove our result, that the reward of the optimal open loop sequence is no greater than twice the expected reward of the greedy closed loop heuristic.

Theorem 3.8. *Under the same assumptions as Theorem 3.1,*

$$J_1^o(\emptyset) \le 2J_1^{\mu^g}(\emptyset)$$

i.e., the expected reward of the closed loop greedy heuristic is at least half the reward of the optimal open loop policy.

Proof. To establish an induction, assume that

$$J_1^o(\emptyset) \le 2J^{\mu^g}_{\to j-1} + \mathrm{E}\, J_j^o(\hbar_j) \tag{3.23}$$

Noting that $\hbar_1 = \emptyset$, this trivially holds for $j = 1$ since $J^{\mu^g}_{\to 0} = 0$. Now, assuming that it holds for j, we show that it also holds for $(j+1)$. Applying Eq. (3.21),

$$J_1^o(\emptyset) \le 2J^{\mu^g}_{\to j-1} + \mathrm{E}\left\{ I(X; z_{w_j}^{o_j}|\hbar_j) + \mathop{\mathrm{E}}_{z_{w_j}^{o_j}|\hbar_j} J_{j+1}^o[(\hbar_j, o_j, z_{w_j}^{o_j})] \right\}$$

By the definition of the closed loop greedy heuristic (Definition 3.2),

$$I(X; z_{w_j}^{o_j}|\hbar_j) \le I(X; z_{w_j}^{\mu_j^g(\hbar_j)}|\hbar_j)$$

hence:

$$J_1^o(\emptyset) \le 2J^{\mu^g}_{\to j-1} + \mathrm{E}\left\{ I(X; z_{w_j}^{\mu_j^g(\hbar_j)}|\hbar_j) + \mathop{\mathrm{E}}_{z_{w_j}^{o_j}|\hbar_j} J_{j+1}^o[(\hbar_j, o_j, z_{w_j}^{o_j})] \right\}$$

Applying Lemma 3.2, followed by Eq. (3.19):

$$\begin{aligned} J_1^o(\emptyset) &\le 2J^{\mu^g}_{\to j-1} + \mathrm{E}\left\{ 2I(X; z_{w_j}^{\mu_j^g(\hbar_j)}|\hbar_j) + \mathop{\mathrm{E}}_{z_{w_j}^{\mu_j^g(\hbar_j)}|\hbar_j} J_{j+1}^o[(\hbar_j, \mu_j^g(\hbar_j), z_{w_j}^{\mu_j^g(\hbar_j)})] \right\} \\ &= 2J^{\mu^g}_{\to j} + \mathrm{E}\, J_{j+1}^o(\hbar_{j+1}) \end{aligned}$$

where $\hbar_{j+1} = (\hbar_j, \mu_j^g(\hbar_j), z_{w_j}^{\mu_j^g(\hbar_j)})$. This establishes the induction step.

Applying the induction step N times, we obtain:

$$J_1^o(\emptyset) \le 2J^{\mu^g}_{\to N} + \mathrm{E}\, J_{N+1}^o(\hbar_{N+1}) = 2J_1^{\mu^g}(\emptyset)$$

since $J_{N+1}^o(\hbar_{N+1}) = 0$ and $J^{\mu^g}_{\to N} = J_1^{\mu^g}(\emptyset)$. □

We emphasize that this performance guarantee is for *expected* performance: it does not provide a guarantee for the change in entropy of every sample path. An online bound cannot be obtained on the basis of a single realization, although online bounds similar to Theorems 3.2 and 3.4 could be calculated through Monte Carlo simulation (to approximate the expectation).

■ 3.5.1 Counterexample: closed loop greedy versus closed loop optimal

While Theorem 3.8 provides a performance guarantee with respect to the optimal open loop sequence, there is no guarantee relating the performance of the closed loop greedy heuristic to the optimal closed loop controller, as the following example illustrates. One exception to this is linear Gaussian models, where closed loop policies can perform no better than open loop sequences, so that the open loop guarantee extends to closed loop performance.

Example 3.3. *Consider the following two-stage problem, where* $X = [a, b, c]^T$*, with* $a \in \{1, \dots, N\}$*,* $b \in \{1, \dots, N+1\}$*, and* $c \in \{1, \dots, M\}$*. The prior distribution of each of these is uniform and independent. In the first stage, we may measure* $z_1^1 = a$ *for reward* $\log N$*, or* $z_1^2 = b$ *for reward* $\log(N+1)$*. In the second stage, we may choose* z_2^i*,* $i \in \{1, \dots, N\}$*, where*

$$z_2^i = \begin{cases} c, & i = a \\ d, & \text{otherwise} \end{cases}$$

where d *is independent of* X*, and is uniformly distributed on* $\{1, \dots, M\}$*. The greedy algorithm in the first stage selects the observation* $z_1^2 = b$*, as it yields a higher reward (*$\log(N+1)$*) than* $z_1^1 = a$ *(*$\log N$*). At the second stage, all options have the same reward,* $\frac{1}{N}\log M$*, so we choose one arbitrarily for a total reward of* $\log(N+1) + \frac{1}{N}\log M$*. The optimal algorithm in the first stage selects the observation* $z_1^1 = a$ *for reward* $\log N$*, followed by the observation* z_2^a *for reward* $\log M$*, for total reward* $\log N + \log M$*. The ratio of the greedy reward to the optimal reward is*

$$\frac{\log(N+1) + \frac{1}{N}\log M}{\log N + \log M} \to \frac{1}{N}, \; M \to \infty$$

Hence, by choosing N *and* M *to be large, we can obtain an arbitrarily small ratio between the greedy closed-loop reward and the optimal closed-loop reward.*

The intuition of this example is that the value of the observation at the first time provides guidance to the controller on which observation it should take at the second time. The greedy heuristic is unable to anticipate the later benefit of this guidance. Notice that the reward of any observation z_2^i without conditioning on the first observation z_1^1 is $I(X; z_2^i) = \frac{1}{N} \log M$. In contrast, the reward of z_2^a conditioned on the *value* of the observation $z_1^1 = a$ is $I(X; z_2^a | \check{z}_1^1) = \log M$. This highlights that the property of diminishing returns (*i.e.*, submodularity) is lost when later choices (and rewards) are conditioned earlier observation *values*.

We conjecture that it may be possible to establish a closed loop performance guarantee for diffusive processes, but it is likely to be dramatically weaker than the bounds presented in this chapter.

3.5.2 Counterexample: closed loop greedy versus open loop greedy

An interesting side note is that the closed loop greedy heuristic can actually result in lower performance than the open loop greedy heuristic, as the following example shows.

Example 3.4. *Consider the following three-stage problem, where* $X = [a, b, c]^T$, *with* $a \in \{1, 2\}$, $b \in \{1, \dots, N+1\}$, *and* $c \in \{1, \dots, N\}$, *where* $N \geq 2$. *The prior distribution of each of these is uniform and independent. In the first stage, a single observation is available,* $z_1 = a$. *In the second stage, we may choose* z_2^1, z_2^2 *or* $z_2^3 = c$, *where* z_2^1 *and* z_2^2 *are given by:*

$$z_2^i = \begin{cases} b, & i = a \\ d, & \textit{otherwise} \end{cases}$$

where d *is independent of* X, *and is uniformly distributed on* $\{1, \dots, N+1\}$. *In the third stage, a single observation is available,* $z_3 = b$. *The closed loop greedy algorithm gains reward* $\log 2$ *for the first observation. At the second observation, the value of* a *is known, hence it selects* $z_2^a = b$ *for reward* $\log (N+1)$. *The final observation* $z_3 = b$ *then yields no further reward; the total reward is* $\log 2(N+1)$. *The open loop greedy heuristic gains the same reward* ($\log 2$) *for the first observation. Since there is no prior knowledge of* a, z_2^1 *and* z_2^2 *yield the same reward* ($\frac{1}{2} \log (N+1)$) *which is less than the reward of* z_2^3 ($\log N$), *hence* z_2^3 *is chosen. The final observation yields reward* $\log N$ *for a total reward of* $\log 2N\sqrt{N+1}$. *For any* $N \geq 2$, *the open loop greedy heuristic achieves higher reward than the closed loop greedy heuristic.*

Since the open loop greedy heuristic has performance no better than the optimal open loop sequence, and the closed loop greedy heuristic has performance no worse than half that of the optimal open loop sequence, the ratio of open loop greedy performance to closed loop greedy performance can be no greater than two. The converse is not true since the performance of the closed loop greedy heuristic is not bounded by the performance of the optimal open loop sequence. This can be demonstrated with a slight modification of Example 3.3 in which the observation z_1^2 is made unavailable.

3.5.3 Closed loop subset selection

A simple modification of the proof of Theorem 3.8 can also be used to extend the result of Theorem 2.4 to closed loop selections. In this structure, there is a single set of observations from which we may choose any subset of $\leq K$ elements out of the finite set $\mathcal{U}$. Again, we obtain the value of each observation before making subsequent selections. We may simplify our notation slightly in this case since we have a single pool of observations. We denote the history of observations chosen and the resulting values to be $\hbar_j = (u_1, z^{u_1}, \dots, u_{j-1}, z^{u_{j-1}})$. The optimal choice of observations is denoted as $(o_1, \dots, o_K)$; the ordering within this choice is arbitrary.

The following definitions are consistent with the previous definitions within the new notation:

$$\begin{aligned}
\mu^g(\hbar_j) &= \underset{u \in \mathcal{U} \setminus \{u_1, \dots, u_{j-1}\}}{\arg\max} \; I(X; z^u | \hbar_j) \\
J_{\to j}^{\mu^g} &= \mathrm{E}\left[\sum_{i=1}^{j} I(X; z^{\mu^g(\hbar_i)} | \hbar_i) \right] \\
&= \mathrm{E}[I(X; z^{\mu^g(\hbar_j)} | \hbar_j)] + J_{\to j-1}^{\mu^g} \\
J_j^o(\hbar_j) &= I(X; z^{o_j}, \dots, z^{o_N} | \hbar_j)
\end{aligned}$$

where $\hbar_1 = \emptyset$ and $\hbar_{j+1} = (\hbar_j, \mu^g(\hbar_j), z^{\mu^g(\hbar_j)})$. Lemmas 3.3 and 3.4 establish two results which we use to prove the theorem.

Lemma 3.3. *For all $i \in \{1, \dots, K\}$,*

$$J_1^o(\emptyset) \leq J_{\to i}^{\mu^g} + \mathrm{E}\, J_1^o(\hbar_{i+1})$$

Proof. The proof follows an induction on the desired result. Note that the expression trivially holds for $i = 0$ since $J_{\to 0}^{\mu^g} = 0$ and $\hbar_1 = \emptyset$. Now suppose that the expression holds for $(i-1)$:

$$\begin{aligned}
J_1^o(\emptyset) &\le J_{\to i-1}^{\mu^g} + \mathrm{E}\, J_1^o(\hbar_i) \\
&\overset{(a)}{=} J_{\to i-1}^{\mu^g} + \mathrm{E}[I(X; z^{o_1}, \dots, z^{o_K} | \hbar_i)] \\
&\overset{(b)}{\le} J_{\to i-1}^{\mu^g} + \mathrm{E}[I(X; z^{\mu^g(\hbar_i)}, z^{o_1}, \dots, z^{o_K} | \hbar_i)] \\
&\overset{(c)}{=} J_{\to i-1}^{\mu^g} + \mathrm{E}[I(X; z^{\mu^g(\hbar_i)} | \hbar_i)] + \mathrm{E}[I(X; z^{o_1}, \dots, z^{o_K} | \hbar_i, z^{\mu^g(\hbar_i)})] \\
&\overset{(d)}{=} J_{\to i}^{\mu^g} + \mathrm{E}\left[\mathop{\mathrm{E}}_{z^{\mu^g(\hbar_i)}|\hbar_i} J_1^o[(\hbar_i, \mu^g(\hbar_i), z^{\mu^g(\hbar_i)})] \right] \\
&\overset{(e)}{=} J_{\to i}^{\mu^g} + \mathrm{E}\, J_1^o[(\hbar_{i+1})]
\end{aligned}$$

where (a) uses the definition of J_1^o, (b) results from the non-decreasing property of MI, (c) results from the MI chain rule, (d) uses the definition of $J_{\to i}^{\mu^g}$ and J_1^o, and (e) uses the definition of $\hbar_{i+1}$. □

Lemma 3.4. *For all $i \in \{1, \dots, K\}$,*

$$J_1^o(\hbar_i) \le K I(X; z^{\mu^g(\hbar_i)} | \hbar_i)$$

Proof. The following steps establish the result:

$$\begin{aligned}
J_1^o(\hbar_i) &\overset{(a)}{=} I(X; z^{o_1}, \dots, z^{o_K} | \hbar_i) \\
&\overset{(b)}{=} \sum_{j=1}^{K} I(X; z^{o_j} | \hbar_i, z^{o_1}, \dots, z^{o_{j-1}}) \\
&\overset{(c)}{\le} \sum_{j=1}^{K} I(X; z^{o_j} | \hbar_i) \\
&\overset{(d)}{\le} \sum_{j=1}^{K} I(X; z^{\mu^g(\hbar_i)} | \hbar_i) \\
&= K I(X; z^{\mu^g(\hbar_i)} | \hbar_i)
\end{aligned}$$

where (a) results from the definition of J_1^o, (b) from the MI chain rule, (c) from submodularity, and (d) from the definition of μ^g. □

Theorem 3.9. *The expected reward of the closed loop greedy heuristic in the K-element subset selection problem is at least $(1 - 1/e)\times$ the reward of the optimal open loop sequence, i.e.,*

$$J^{\mu^g}_{\to K} \geq (1 - 1/e) J^o_1(\emptyset)$$

Proof. To commence, note from Lemma 3.4 that, for all $i \in \{1, \dots, K\}$:

$$\begin{aligned} \mathrm{E}\, J^o_1(\hbar_i) &\leq K\, \mathrm{E}[I(X; z^{\mu^g(\hbar_i)}|\hbar_i)] \\ &= K(J^{\mu^g}_{\to i} - J^{\mu^g}_{\to i-1}) \end{aligned}$$

Combining this with Lemma 3.3, we obtain:

$$J^o_1(\emptyset) \leq J^{\mu^g}_{\to i-1} + K(J^{\mu^g}_{\to i} - J^{\mu^g}_{\to i-1}) \quad \forall\, i \in \{1, \dots, K\} \tag{3.24}$$

Letting $\rho_j = J^{\mu^g}_{\to j} - J^{\mu^g}_{\to j-1}$ and $Z = J^o_1(\emptyset)$ and comparing Eq. (3.24) with Eq. (2.93) in Theorem 2.4, we obtain the desired result. □

■ 3.6 Guarantees on the Cramér-Rao bound

While the preceding discussion has focused exclusively on mutual information, the results are applicable to a larger class of objectives. The following analysis shows that the guarantees in Sections 2.4.4, 2.4.5, 3.1 and 3.3 can also yield a guarantee on the posterior Cramér-Rao bound. We continue to assume that observations are independent conditioned on X.

To commence, assume that the objective we seek to maximize is the log of the determinant of the Fisher information: (we will later show that a guarantee on this quantity yields a guarantee on the determinant of the Cramér-Rao bound matrix)

$$D(X; z^{\mathcal{A}}) \triangleq \log \frac{\left|\mathbf{J}^{\emptyset}_X + \sum_{a \in \mathcal{A}} \bar{\mathbf{J}}^{z^a}_X\right|}{|\mathbf{J}^{\emptyset}_X|} \tag{3.25}$$

where $\mathbf{J}^{\emptyset}_X$ and $\bar{\mathbf{J}}^{z^a}_X$ are as defined in Section 2.1.6. We can also define an increment function similar to conditional MI:

$$\begin{aligned} D(X; z^{\mathcal{A}}|z^{\mathcal{B}}) &\triangleq D(X; z^{\mathcal{A}\cup\mathcal{B}}) - D(X; z^{\mathcal{B}}) && (3.26) \\ &= \log \frac{\left|\mathbf{J}^{\emptyset}_X + \sum_{a \in \mathcal{A}\cup\mathcal{B}} \bar{\mathbf{J}}^{z^a}_X\right|}{\left|\mathbf{J}^{\emptyset}_X + \sum_{b \in \mathcal{B}} \bar{\mathbf{J}}^{z^b}_X\right|} && (3.27) \end{aligned}$$

It is easy to see that the reward function $D(X; z^{\mathcal{A}})$ is a non-decreasing, submodular function of the set $\mathcal{A}$ by comparing Eq. (3.25) to the MI of a linear Gaussian process (see Section 2.3.4; also note that $I(X; z^{\mathcal{A}}) = \frac{1}{2}D(X; z^{\mathcal{A}})$ in the linear Gaussian case). The following development derives these properties without using this link. We will require the following three results from linear algebra; for proofs see [1].

Lemma 3.5. *Suppose that* $\mathbf{A} \succeq \mathbf{B} \succ \mathbf{0}$. *Then* $\mathbf{B}^{-1} \succeq \mathbf{A}^{-1} \succ \mathbf{0}$.

Lemma 3.6. *Suppose that* $\mathbf{A} \succeq \mathbf{B} \succ \mathbf{0}$. *Then* $|\mathbf{A}| \geq |\mathbf{B}| > 0$.

Lemma 3.7. *Suppose* $\mathbf{A} \in \mathbb{R}^{n\times m}$ *and* $\mathbf{B} \in \mathbb{R}^{m\times n}$. *Then* $|\mathbf{I} + \mathbf{AB}| = |\mathbf{I} + \mathbf{BA}|$.

Theorem 3.10. $D(X; z^{\mathcal{A}})$ *is a non-decreasing set function of the set* $\mathcal{A}$*, with* $D(X; z^{\emptyset}) = 0$. *Assuming that all observations are independent conditioned on* X*,* $D(X; z^{\mathcal{A}})$ *is a submodular function of* $\mathcal{A}$.

Proof. To show that $D(X; z^{\mathcal{A}})$ is non-decreasing, consider the increment:

$$D(X; z^{\mathcal{A}}|z^{\mathcal{B}}) = \log \frac{\left|\mathbf{J}_X^{\emptyset} + \sum_{a\in\mathcal{A}\cup\mathcal{B}} \bar{\mathbf{J}}_X^{z^a}\right|}{\left|\mathbf{J}_X^{\emptyset} + \sum_{b\in\mathcal{B}} \bar{\mathbf{J}}_X^{z^b}\right|}$$

Since $\mathbf{J}_X^{\emptyset} + \sum_{a\in\mathcal{A}\cup\mathcal{B}} \bar{\mathbf{J}}_X^{z^a} \succeq \mathbf{J}_X^{\emptyset} + \sum_{b\in\mathcal{B}} \bar{\mathbf{J}}_X^{z^b}$, we have by Lemma 3.6 that $|\mathbf{J}_X^{\emptyset} + \sum_{a\in\mathcal{A}\cup\mathcal{B}} \bar{\mathbf{J}}_X^{z^a}| \geq |\mathbf{J}_X^{\emptyset} + \sum_{b\in\mathcal{B}} \bar{\mathbf{J}}_X^{z^b}|$, hence $D(X; z^{\mathcal{A}}|z^{\mathcal{B}}) \geq 0$. If $\mathcal{A} = \emptyset$, we trivially find $D(X; z^{\mathcal{A}}) = 0$.

For submodularity, we need to prove that $\forall\ \mathcal{B} \supseteq \mathcal{A}$,

$$D(X; z^{\mathcal{C}\cup\mathcal{A}}) - D(X; z^{\mathcal{A}}) \geq D(X; z^{\mathcal{C}\cup\mathcal{B}}) - D(X; z^{\mathcal{B}}) \tag{3.28}$$

For convenience, define the short-hand notation:

$$\bar{\mathbf{J}}_X^{\mathcal{C}} \triangleq \bar{\mathbf{J}}_X^{z^{\mathcal{C}}} \triangleq \sum_{c\in\mathcal{C}} \bar{\mathbf{J}}_X^{z^c}$$

$$\mathbf{J}_X^{\mathcal{A}} \triangleq \mathbf{J}_X^{z^{\mathcal{A}}} \triangleq \mathbf{J}_X^{\emptyset} + \sum_{a\in\mathcal{A}} \bar{\mathbf{J}}_X^{z^a}$$

We proceed by forming the difference of the two sides of the expression in Eq. (3.28):

$$\begin{aligned}
[D(X;&z^{\mathcal{C}\cup\mathcal{A}}) - D(X;z^{\mathcal{A}})] - [D(X;z^{\mathcal{C}\cup\mathcal{B}}) - D(X;z^{\mathcal{B}})] \\
&= \left[\log\frac{|\mathbf{J}_X^{\mathcal{A}} + \bar{\mathbf{J}}_X^{\mathcal{C}\setminus\mathcal{A}}|}{|\mathbf{J}_X^{\emptyset}|} - \log\frac{|\mathbf{J}_X^{\mathcal{A}}|}{|\mathbf{J}_X^{\emptyset}|}\right] - \left[\log\frac{|\mathbf{J}_X^{\mathcal{B}} + \bar{\mathbf{J}}_X^{\mathcal{C}\setminus\mathcal{B}}|}{|\mathbf{J}_X^{\emptyset}|} - \log\frac{|\mathbf{J}_X^{\mathcal{B}}|}{|\mathbf{J}_X^{\emptyset}|}\right] \\
&= \log\frac{|\mathbf{J}_X^{\mathcal{A}} + \bar{\mathbf{J}}_X^{\mathcal{C}\setminus\mathcal{A}}|}{|\mathbf{J}_X^{\mathcal{A}}|} - \log\frac{|\mathbf{J}_X^{\mathcal{B}} + \bar{\mathbf{J}}_X^{\mathcal{C}\setminus\mathcal{B}}|}{|\mathbf{J}_X^{\mathcal{B}}|} \\
&= \log\left|\mathbf{I} + \mathbf{J}_X^{\mathcal{A}\,-\frac{1}{2}}\bar{\mathbf{J}}_X^{\mathcal{C}\setminus\mathcal{A}}\mathbf{J}_X^{\mathcal{A}\,-\frac{1}{2}}\right| - \log\left|\mathbf{I} + \mathbf{J}_X^{\mathcal{B}\,-\frac{1}{2}}\bar{\mathbf{J}}_X^{\mathcal{C}\setminus\mathcal{B}}\mathbf{J}_X^{\mathcal{B}\,-\frac{1}{2}}\right|
\end{aligned}$$

where $\mathbf{J}_X^{\mathcal{B}\,-\frac{1}{2}}$ is the inverse of the symmetric square root matrix of $\mathbf{J}_X^{\mathcal{B}}$. Since $\bar{\mathbf{J}}_X^{\mathcal{C}\setminus\mathcal{A}} \succeq \bar{\mathbf{J}}_X^{\mathcal{C}\setminus\mathcal{B}}$, we can write through Lemma 3.6:

$$\geq \log\left|\mathbf{I} + \mathbf{J}_X^{\mathcal{A}\,-\frac{1}{2}}\bar{\mathbf{J}}_X^{\mathcal{C}\setminus\mathcal{B}}\mathbf{J}_X^{\mathcal{A}\,-\frac{1}{2}}\right| - \log\left|\mathbf{I} + \mathbf{J}_X^{\mathcal{B}\,-\frac{1}{2}}\bar{\mathbf{J}}_X^{\mathcal{C}\setminus\mathcal{B}}\mathbf{J}_X^{\mathcal{B}\,-\frac{1}{2}}\right|$$

Factoring $\bar{\mathbf{J}}_X^{\mathcal{C}\setminus\mathcal{B}}$ and applying Lemma 3.7, we obtain:

$$= \log\left|\mathbf{I} + \bar{\mathbf{J}}_X^{\mathcal{C}\setminus\mathcal{B}\,\frac{1}{2}}\mathbf{J}_X^{\mathcal{A}\,-1}\bar{\mathbf{J}}_X^{\mathcal{C}\setminus\mathcal{B}\,\frac{1}{2}}\right| - \log\left|\mathbf{I} + \bar{\mathbf{J}}_X^{\mathcal{C}\setminus\mathcal{B}\,\frac{1}{2}}\mathbf{J}_X^{\mathcal{B}\,-1}\bar{\mathbf{J}}_X^{\mathcal{C}\setminus\mathcal{B}\,\frac{1}{2}}\right|$$

Finally, since $\mathbf{J}_X^{\mathcal{A}\,-1} \succeq \mathbf{J}_X^{\mathcal{B}\,-1}$ (by Lemma 3.5), we obtain through Lemma 3.6:

$$\geq 0$$

This establishes submodularity of $D(X;z^{\mathcal{A}})$. □

Following Theorem 3.10, if we use the greedy selection algorithm on $D(X;z^{\mathcal{A}})$, then we obtain the guarantee from Theorem 3.1 that $D(X;z^{\mathcal{G}}) \geq 0.5D(X;z^{\mathcal{O}})$ where $z^{\mathcal{G}}$ is the set of observations chosen by the greedy heuristic and $z^{\mathcal{O}}$ is the optimal set. The following theorem maps this into a guarantee on the posterior Cramér-Rao bound.

Theorem 3.11. *Let $z^{\mathcal{G}}$ be the set of observations chosen by the greedy heuristic operating on $D(X;z^{\mathcal{A}})$, and let $z^{\mathcal{O}}$ be the optimal set of observations for this objective. Assume that, through one of guarantees (online or offline) in Section 3.1 or 3.3, we have for some β:*

$$D(X;z^{\mathcal{G}}) \geq \beta D(X;z^{\mathcal{O}})$$

Then the determinants of the matrices in the posterior Cramér-Rao bound in Section 2.1.6 satisfy the following inequality:

$$|\mathbf{C}_X^{\mathcal{G}}| \leq |\mathbf{C}_X^{\emptyset}|\left(\frac{|\mathbf{C}_X^{\mathcal{O}}|}{|\mathbf{C}_X^{\emptyset}|}\right)^{\beta}$$

The ratio $|\mathbf{C}_X^{\mathcal{G}}|/|\mathbf{C}_X^{\emptyset}|$ is the fractional reduction of uncertainty (measured through covariance determinant) which is gained through using the selected observations rather than the prior information alone. Thus Theorem 3.11 provides a guarantee on how much of the optimal reduction you lose by using the greedy heuristic. From Section 2.1.6 and Lemma 3.6, the determinant of the error covariance of any estimator of X using the data $z^{\mathcal{G}}$ is lower bounded by $|\mathbf{C}_X^{\mathcal{G}}|$.

Proof. By the definition of $D(X;z^{\mathcal{A}})$ (Eq. (3.25)) and from the assumed condition we have:

$$[\log|\mathbf{J}_X^{z^{\mathcal{G}}}| - \log|\mathbf{J}_X^{\emptyset}|] \geq \beta[\log|\mathbf{J}_X^{z^{\mathcal{O}}}| - \log|\mathbf{J}_X^{\emptyset}|]$$

Substituting in $\mathbf{C}_X^{z^{\mathcal{A}}} = [\mathbf{J}_X^{z^{\mathcal{A}}}]^{-1}$ and using the identity $|\mathbf{A}^{-1}| = |\mathbf{A}|^{-1}$ we obtain:

$$[\log|\mathbf{C}_X^{z^{\mathcal{G}}}| - \log|\mathbf{C}_X^{\emptyset}|] \leq \beta[\log|\mathbf{C}_X^{z^{\mathcal{O}}}| - \log|\mathbf{C}_X^{\emptyset}|]$$

Exponentiating both sides this becomes:

$$\frac{|\mathbf{C}_X^{z^{\mathcal{G}}}|}{|\mathbf{C}_X^{\emptyset}|} \leq \left(\frac{|\mathbf{C}_X^{z^{\mathcal{O}}}|}{|\mathbf{C}_X^{\emptyset}|}\right)^{\beta}$$

which is the desired result. □

The results of Sections 3.2 and 3.5 do not apply to Fisher information since the required properties (Eq. (3.4) and Eq. (3.14) respectively) do not generally apply to $D(X;z^{\mathcal{A}})$. Obviously linear Gaussian processes are an exception to this rule, since $I(X;z^{\mathcal{A}}) = \frac{1}{2}D(X;z^{\mathcal{A}})$ in this case.

■ 3.7 Estimation of rewards

The analysis in Sections 3.1–3.6 assumes that all reward values can be calculated exactly. While this is possible for some common classes of problems (such as linear Gaussian models), approximations are often necessary. The analysis in [46] can be easily extended to the algorithms described in this chapter. As an example, consider the proof of Theorem 3.1, where the greedy heuristic is used with estimated MI rewards,

$$g_j = \underset{g\in\{1,\dots,n_{w_j}\}}{\arg\max}\ \hat{I}(X;z_{w_j}^{g}|z_{w_1}^{g_1},\dots,z_{w_{j-1}}^{g_{j-1}})$$

and the error in the MI estimate is bounded by ϵ, *i.e.*,

$$|I(X; z^{g}_{w_j} | z^{g_1}_{w_1}, \dots, z^{g_{j-1}}_{w_{j-1}}) - \hat{I}(X; z^{g}_{w_j} | z^{g_1}_{w_1}, \dots, z^{g_{j-1}}_{w_{j-1}})| \leq \epsilon$$

From step (c) of Theorem 3.1,

$$\begin{aligned}
I(X; z^{o_1}_{w_1}, \dots, z^{o_M}_{w_M}) &\leq I(X; z^{g_1}_{w_1}, \dots, z^{g_M}_{w_M}) + \sum_{j=1}^{M} I(X; z^{o_j}_{w_j} | z^{g_1}_{w_1}, \dots, z^{g_{j-1}}_{w_j}) \\
&\leq I(X; z^{g_1}_{w_1}, \dots, z^{g_M}_{w_M}) + \sum_{j=1}^{M} \left[\hat{I}(X; z^{o_j}_{w_j} | z^{g_1}_{w_1}, \dots, z^{g_{j-1}}_{w_j}) + \epsilon \right] \\
&\leq I(X; z^{g_1}_{w_1}, \dots, z^{g_M}_{w_M}) + \sum_{j=1}^{M} \left[\hat{I}(X; z^{g_j}_{w_j} | z^{g_1}_{w_1}, \dots, z^{g_{j-1}}_{w_j}) + \epsilon \right] \\
&\leq I(X; z^{g_1}_{w_1}, \dots, z^{g_M}_{w_M}) + \sum_{j=1}^{M} \left[I(X; z^{g_j}_{w_j} | z^{g_1}_{w_1}, \dots, z^{g_{j-1}}_{w_j}) + 2\epsilon \right] \\
&= 2I(X; z^{g_1}_{w_1}, \dots, z^{g_M}_{w_M}) + 2M\epsilon
\end{aligned}$$

Hence the deterioration in the performance guarantee is at most $2M\epsilon$.

3.8 Extension: general matroid problems

The guarantees described in this chapter have concentrated on problem structures involving several sets of observations, in which we select a fixed number of observations from each set. In this section, we briefly demonstrate wider a class of problems that may be addressed using the previous work in [77] (described in Section 2.4.4), which, to our knowledge, has not been previously applied in this context.

As described in Section 2.4.3, $(\mathcal{U}, \mathscr{F})$ is a matroid if $\forall\ \mathcal{A}, \mathcal{B} \in \mathscr{F}$ such that $|\mathcal{A}| < |\mathcal{B}|$, $\exists\ u \in \mathcal{B} \backslash \mathcal{A}$ such that $\mathcal{A} \cup \{u\} \in \mathscr{F}$. Consider the class of problems described in Assumption 3.1, in which we are choosing observations from N sets, and we may choose k_i elements from the i-th set. It is easy to see that this class of selection problems fits in to the matroid class: given any two valid[4] observation selection sets $\mathcal{A}$, $\mathcal{B}$ with $|\mathcal{A}| < |\mathcal{B}|$, pick any set i such that the number of elements in $\mathcal{A}$ from this set is fewer than the number of elements in $\mathcal{B}$ from this set (such an i must exist since $\mathcal{A}$ and $\mathcal{B}$ have different cardinality). Then we can find an element in $\mathcal{B}$ from the i-th set which is not in $\mathcal{A}$, but can be added to $\mathcal{A}$ while maintaining a valid set.

[4] By *valid*, we mean that no more than k_i elements are chosen from the i-th set.

A commonly occurring structure which cannot be addressed within Assumption 3.1 is detailed in Assumption 3.7. The difference is that there is an upper limit on the total number of observations able to be taken, as well as on the number of observations able to be taken from each set. Under this generalization, the greedy heuristic must consider all remaining observations at each stage of the selection problem: we cannot visit one set at a time in an arbitrary order as in Assumption 3.1. This is the key advantage of Theorem 3.1 over the prior work described in Section 2.4.4 when dealing with problems that have the structure of Assumption 3.1.

Assumption 3.7. *There are N sets of observations, $\{\{z_1^1, \dots, z_1^{n_1}\}, \dots, \{z_N^1, \dots, z_N^{n_N}\}\}$, which are mutually independent conditioned on the quantity to be estimated (X). Any k_i observations can be chosen out of the i-th set ($\{z_i^1, \dots, z_i^{n_i}\}$), but the total number of observations chosen cannot exceed K.*

The structure in Assumption 3.7 clearly remains a matroid: as per the previous discussion, take any two observation selection sets $\mathcal{A}$, $\mathcal{B}$ with $|\mathcal{A}| < |\mathcal{B}|$ and pick any set i such that the number of elements in $\mathcal{A}$ from this set is fewer than the number of elements in $\mathcal{B}$ from this set. Then we can find an element in $\mathcal{B}$ from the i-th set which is not in $\mathcal{A}$, but can be added to $\mathcal{A}$ while maintaining a valid set (since $|\mathcal{A}| < |\mathcal{B}| \leq K$).

■ 3.8.1 Example: beam steering

As an example, consider a beam steering problem similar to the one described in Section 3.1.4, but where the total number of observations chosen (of either object) in the 200 steps should not exceed 50 (we assume an open loop control structure, in which we choose all observation actions before obtaining any of the resulting observation values). This may be seen to fit within the structure of Assumption 3.7, so the selection algorithm and guarantee of Section 2.4.4 applies. Fig. 3.7 shows the observations chosen at each time in the previous case (where an observation was chosen in every time step) and in the constrained case in which a total of 50 observations is chosen.

■ 3.9 Extension: platform steering

A problem structure which generalizes the open-loop observation selection problem involves control of sensor state. In this case, the controller simultaneously selects observations in order to control an information state, and controls a finite state, completely observed Markov chain (with a deterministic transition law) which determines the subset of measurements available at each time. We now describe a greedy algorithm which

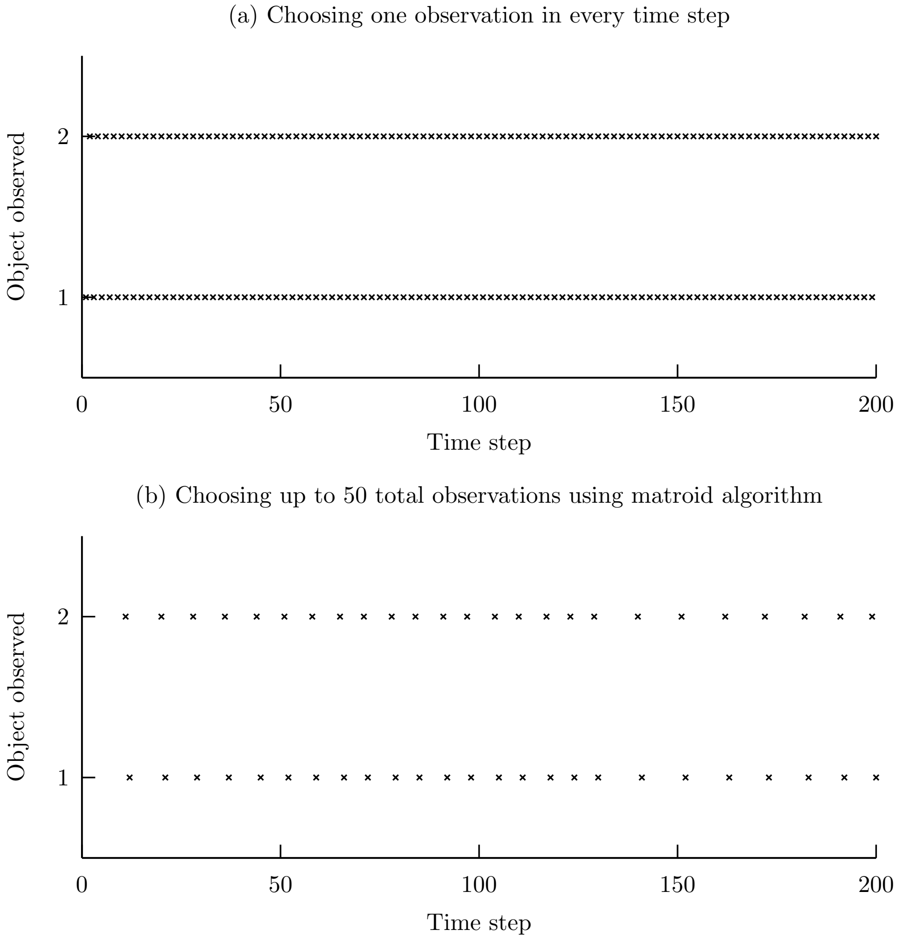


Figure 3.7. (a) shows the observations chosen in the example in Sections 3.1.4 and 3.2.4 when $q = 1$. (b) shows the smaller set of observations chosen in the constrained problem using the matroid selection algorithm.

one may use to control such a problem. We assume that in each sensor state there is a single measurement available.

Definition 3.3. *The greedy algorithm for jointly selecting observations and controlling sensor state commences in the following manner: (s_0 is the initial state of the algorithm which is fixed upon execution)*

Stage 0: *Calculate the reward of the initial observation*

$$J_0 = I(x; s_0)$$

Stage 1: *Consider each possible sensor state which can follow s_0. Calculate the reward*

$$J_1(s_1) = \begin{cases} J_0 + I(x; s_1 | s_0), & s_1 \in \mathcal{S}^{s_0} \\ -\infty, & \textit{otherwise} \end{cases}$$

Stage i: *For each s_i, calculate the highest reward sequence which can precede that state:*

$$s^g_{i-1}(s_i) = \operatorname*{arg\,max}_{s_{i-1} | s_i \in \mathcal{S}^{s_{i-1}}} J_{i-1}(s_{i-1})$$

Then, for each s_i, add the reward of the new observation obtained in that state:

$$J_i(s_i) = J_{i-1}(s^g_{i-1}(s_i)) + I(x; s_i | s^g_{i-1}(s_i), s^g_{i-2}(s^g_{i-1}(s_i)), \dots, s_0)$$

After stage N, calculate $\tilde{s}^g_N = \arg\max_s J_N(s)$. The final sequence is found through the backward recursion $\tilde{s}^g_i = s^g_i(\tilde{s}^g_{i+1})$.

The following example demonstrates that the ratio between the greedy algorithm for open loop joint observation and sensor state control and the optimal open loop algorithm can be arbitrarily close to zero.

Example 3.5. *We seek to control the sensor state $s_k \in \{0, \dots, 2N+1\}$. In stage 0 we commence from sensor state $s_0 = 0$. From sensor state 0 we can transition to any other sensor state. From sensor state $s \neq 0$, we can stay in the same state, or transition to sensor state $(s-1)$ (provided that $s > 1$) or $(s+1)$ (provided that $s < 2N+1$).*

The unobserved state, x, about which we seek to gather information is static ($x_1 = x_2 = \cdots = x$), and consists of $N(2N+1)$ binary elements

$\{x^{1,1},\dots,x^{1,2N+1},\dots,x^{N,1},\dots,x^{N,2N+1}\}$. *The prior distribution of these elements is uniform.*

The observation in sensor state $2j-2$, $j \in \{1,\dots,N\}$ *is uninformative. The observation in sensor state* $2j-1$, $j \in \{1,\dots,N\}$ *at stage* i *provides a direct measurement of the state elements* $\{x^{j,1},\dots,x^{j,i}\}$.

The greedy algorithm commences stage 0 with $J_0 = 0$. *In stage 1, we obtain:*

$$J_1(s_1) = \begin{cases} -\infty, & s_1 = 0 \\ 0, & s_1 \text{ positive and even} \\ 1, & s_1 \text{ odd} \end{cases}$$

Suppose that we commence stage i *with*

$$J_{i-1}(s_{i-1}) = \begin{cases} -\infty, & s_{i-1} = 0 \\ i-2, & s_{i-1} \text{ positive and even} \\ i-1, & s_{i-1} \text{ odd} \end{cases}$$

Settling ties by choosing the state with lower index, we obtain:

$$s^g_{i-1}(s_i) = \begin{cases} \text{undefined}, & s_i = 0 \\ s_i - 1, & s_i \text{ positive and even} \\ s_i, & s_i \text{ odd} \end{cases}$$

Incorporating the new observation, we obtain:

$$J_i(s_i) = \begin{cases} -\infty, & s_i = 0 \\ i-1, & s_i \text{ positive and even} \\ i, & s_i \text{ odd} \end{cases}$$

At the end of stage $2N+1$, *we find that the best sequence remains in any odd-numbered state for all stages, and obtains a reward of* $2N+1$.

Compare this result to the optimal sequence, which visits state i *at stage* i. *The reward gained in each stage is:*

$$I(x; s_i | s_0, \dots, s_{i-1}) = \begin{cases} 0, & i \text{ even} \\ i, & i \text{ odd} \end{cases}$$

The total reward is thus

$$\sum_{j=0}^{N} (2j+1) = N^2 + 2N + 1$$

The ratio of greedy reward to optimal reward for the problem involving $2N+1$ *stages is:*

$$\frac{2N+1}{N^2+2N+1} \to 0,\ N \to \infty$$

■ 3.10 Conclusion

The performance guarantees presented in this chapter provide theoretical basis for simple heuristic algorithms that are widely used in practice. The guarantees apply to both open loop and closed loop operation, and are naturally tighter for diffusive processes, or discounted objectives. The examples presented throughout the chapter demonstrate the applicability of the guarantees to a wide range of waveform selection and beam steering problems, and the substantially stronger online guarantees that can be obtained for specific problems through computation of additional quantities after the greedy selection has been completed.

Chapter 4

Independent objects and integer programming

IN this chapter, we use integer programming methods to construct an open loop plan of which sensor actions to perform within a given planning horizon. We may either execute this entire plan, or execute some portion of the plan before constructing an updated plan (so-called open loop feedback control, as discussed in Section 2.2.2).

The emphasis of our formulation is to exploit the structure which results in sensor management problems involving observation of multiple independent objects. In addition to the previous assumption that observations should be independent conditioned on the state, three new assumptions must be met for this structure to arise:

1. The prior distribution of the objects must be independent

2. The objects must evolve according to independent dynamical processes

3. The objects must be observed through independent observation processes

When these three assumptions are met, the mutual information reward of observations of different objects becomes the sum of the individual observation rewards—*i.e.*, submodularity becomes additivity. Accordingly, one may apply a variety of techniques that exploit the structure of integer programming with linear objectives and constraints.

We commence this chapter by developing in Section 4.1 a simple formulation that allows us to select up to one observation for each object. In Section 4.2, we generalize this to our proposed formulation, which finds the optimal plan, permits multiple observations of each object, and can address observations that require different durations to complete. In Section 4.4, we perform experiments which explore the computational efficiency of this formulation on a range of problems. The formulation is generalized to consider resources with arbitrary capacities in Section 4.5; this structure is useful in problems involving time invariant rewards.

■ 4.1 Basic formulation

We commence by presenting a special case of the abstraction discussed in Section 4.2. As per the previous chapter, we assume that the planning horizon is broken into discrete time slots, numbered $\{1, \dots, N\}$, that a sensor can perform at most one task in any time slot, and that each observation action occupies exactly one time slot. We have a number of objects, numbered $\{1, \dots, M\}$, each of which can be observed in any time slot. Our task is to determine which object to observe in each time slot. Initially, we assume that we have only a single mode for the sensor to observe each object in each time slot, such that the only choice to be made in each time slot is which object to observe; this represents the purest form of the beam steering structure discussed in Section 1.1.

To motivate this structure, consider a problem in which we use an airborne sensor to track objects moving on the ground beneath foliage. In some positions, objects will be in clear view and observation will yield accurate position information; in other positions, objects will be obscured by foliage and observations will be essentially uninformative. Within the time scale of a planning horizon, objects will move in and out of obscuration, and it will be preferable to observe objects during the portion of time in which they are expected to be in clear view.

■ 4.1.1 Independent objects, additive rewards

The basis for our formulation is the fact that rewards for observations of independent objects are additive. Denoting by $X^i = \{x^i_1, \dots, x^i_N\}$ the joint state (over the planning horizon) of object i, we define the reward of observation set $\mathcal{A}^i \subseteq \{1, \dots, N\}$ of object i (*i.e.*, $\mathcal{A}^i$ represents the subset of time slots in which we observe object i) to be:

$$r^i_{\mathcal{A}^i} = I(X^i; z^i_{\mathcal{A}^i}) \tag{4.1}$$

where $z^i_{\mathcal{A}^i}$ are the random variables corresponding to the observations of object i in the time slots in $\mathcal{A}^i$. As discussed in the introduction of this chapter, if we assume that the initial states of the objects are independent:

$$p(x^1_1, \dots, x^M_1) = \prod_{i=1}^{M} p(x^i_1) \tag{4.2}$$

and that the dynamical processes are independent:

$$p(x_k^1, \dots, x_k^M | x_{k-1}^1, \dots, x_{k-1}^M) = \prod_{i=1}^M p(x_k^i | x_{k-1}^i) \tag{4.3}$$

and, finally, that observations are independent conditioned on the state, and that each observation relates to a single object:

$$p(z_{\mathcal{A}^1}^1, \dots, z_{\mathcal{A}^M}^M | X^1, \dots, X^M) = \prod_{i=1}^M \prod_{k \in \mathcal{A}^i} p(z_k^i | X^i) \tag{4.4}$$

then the conditional distributions of the states of the objects will be independent conditioned on any set of observations:

$$p(X^1, \dots, X^M | z_{\mathcal{A}^1}^1, \dots, z_{\mathcal{A}^M}^M) = \prod_{i=1}^M p(X^i | z_{\mathcal{A}^i}^i) \tag{4.5}$$

In this case, we can write the reward of choosing observation set $\mathcal{A}^i$ for object $i \in \{1, \dots, M\}$ as:

$$\begin{aligned} I(X^1, \dots, X^M; z_{\mathcal{A}^1}^1, \dots, z_{\mathcal{A}^M}^M) &= H(X^1, \dots, X^M) - H(X^1, \dots, X^M | z_{\mathcal{A}^1}^1, \dots, z_{\mathcal{A}^M}^M) \\ &= \sum_{i=1}^M H(X^i) - \sum_{i=1}^M H(X^i | z_{\mathcal{A}^i}^i) \\ &= \sum_{i=1}^M I(X^i; z_{\mathcal{A}^i}^i) \\ &= \sum_{i=1}^M r_{\mathcal{A}^i}^i \qquad (4.6) \end{aligned}$$

■ 4.1.2 Formulation as an assignment problem

While Eq. (4.6) shows that rewards are additive across objects, the reward for taking several observations of the same object is not additive. In fact, it can be easily shown through submodularity that (see Lemma 4.3)

$$I(X^i; z_{\mathcal{A}^i}^i) \le \sum_{k \in \mathcal{A}^i} I(X^i; z_k^i)$$

As an initial approach, consider the problem structure which results if we restrict ourselves to observing each object *at most once*. For this restriction to be sensible, we must assume that the number of time slots is no greater than the number of objects, so that an observation will be taken in each time slot. In this case, the overall reward is simply the sum of single observation rewards, since each set $\mathcal{A}^i$ has cardinality at most one. Accordingly, the problem of determining which object to observe at each time reduces to an asymmetric assignment problem, assigning time slots to objects. The assignment problem may be written as a linear program in the following form:

$$\max_{\omega_k^i} \sum_{i=1}^{M} \sum_{k=1}^{N} r^i_{\{k\}} \omega_k^i \tag{4.7a}$$

$$\text{s.t.} \sum_{i=1}^{M} \omega_k^i \leq 1 \quad \forall\, k \in \{1, \dots, N\} \tag{4.7b}$$

$$\sum_{k=1}^{N} \omega_k^i \leq 1 \quad \forall\, i \in \{1, \dots, M\} \tag{4.7c}$$

$$\omega_k^i \in \{0, 1\} \quad \forall\, i,\ k \tag{4.7d}$$

The binary indicator variable ω_k^i assumes the value of one if object i is observed in time slot k and zero otherwise. The integer program may be interpreted as follows:

- The objective in Eq. (4.7a) is the sum of the rewards corresponding to each ω_k^i that assumes a non-zero value, *i.e.*, the sum of the rewards of each choice of a particular object to observe in a particular time slot.

- The constraint in Eq. (4.7b) requires that at most one object can be observed in any time slot. This ensures that physical sensing constraints are not exceeded.

- The constraint in Eq. (4.7c) requires that each object can be observed at most once. This ensures that the additive reward objective provides the exact reward value (the reward for selecting two observations of the same object is not the sum of the rewards of each individual observation).

- The integrality constraint in Eq. (4.7d) requires solutions to take on binary values. Because of the structure of the assignment problem, this can be relaxed to allow any $\omega_k^i \in [0, 1]$, and there will still be an integer point which attains the optimal solution.

The assignment problem can be solved efficient using algorithms such as Munkres [72], Jonker-Volgenant-Castañón [24], or Bertsekas' auction [11]. We use the auction algorithm in our experiments.

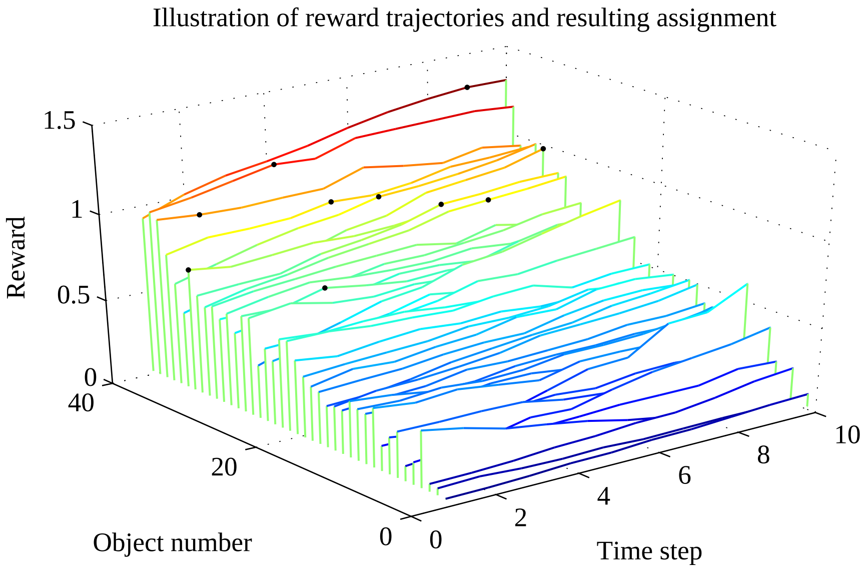


Figure 4.1. Example of operation of assignment formulation. Each "strip" in the diagram corresponds to the reward for observing a particular object at different times over the 10-step planning horizon (assuming that it is only observed once within the horizon). The role of the auction algorithm is to pick one unique object to observe at each time in the planning horizon in order to maximize the sum of the rewards gained. The optimal solution is shown as black dots.

One can gain an intuition for this formulation from the diagram in Fig. 4.1. The rewards correspond to a snapshot of the scenario discussed in Section 4.1.3. The scenario considers the problem of object tracking when probability of detection varies with position. The diagram illustrates the trade-off which the auction algorithm performs: rather than taking the highest reward observation at the first time (as the greedy heuristic would do), the controller defers measurement of that object until a later time when a more valuable observation is available. Instead, it measures at the first time an object with comparatively lower reward value, but one for which the observations at later times are still less valuable.

■ 4.1.3 Example

The approach was tested on a tracking scenario in which a single sensor is used to simultaneously track 20 objects. The state of object i at time k, $\boldsymbol{x}_k^i$, consists of position and velocity in two dimensions. The state evolves according to a linear Gaussian model:

$$\boldsymbol{x}_{k+1}^i = \mathbf{F}\boldsymbol{x}_k^i + \boldsymbol{w}_k^i \tag{4.8}$$

where $\boldsymbol{w}_k^i \sim \mathcal{N}\{\boldsymbol{w}_k^i; \mathbf{0}, \mathbf{Q}\}$ is a white Gaussian noise process. $\mathbf{F}$ and $\mathbf{Q}$ are set as:

$$\mathbf{F} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}; \ \mathbf{Q} = q \begin{bmatrix} \frac{T^3}{3} & \frac{T^2}{2} & 0 & 0 \\ \frac{T^2}{2} & T & 0 & 0 \\ 0 & 0 & \frac{T^3}{3} & \frac{T^2}{2} \\ 0 & 0 & \frac{T^2}{2} & T \end{bmatrix} \tag{4.9}$$

The diffusion strength q is set to 0.01. The sensor can be used to observe any one of the M objects in each time step. The measurement obtained from observing object u_k with the sensor consists of a detection flag $d_k^{u_k} \in \{0, 1\}$ and, if $d_k^{u_k} = 1$, a linear Gaussian measurement of the position, $\boldsymbol{z}_k^{u_k}$:

$$\boldsymbol{z}_k^{u_k} = \mathbf{H}\boldsymbol{x}_k^{u_k} + \boldsymbol{v}_k^{u_k} \tag{4.10}$$

where $\boldsymbol{v}_k^{u_k} \sim \mathcal{N}\{\boldsymbol{v}_k^{u_k}; \mathbf{0}, \mathbf{R}\}$ is a white Gaussian noise process, independent of $\boldsymbol{w}_k^{u_k}$. $\mathbf{H}$ and $\mathbf{R}$ are set as:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}; \ \mathbf{R} = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \tag{4.11}$$

The probability of detection $P_{d_k^{u_k}|\boldsymbol{x}_k^{u_k}}(1|\boldsymbol{x}_k^{u_k})$ is a function of object position. The function is randomly generated for each Monte Carlo simulation; an example of the

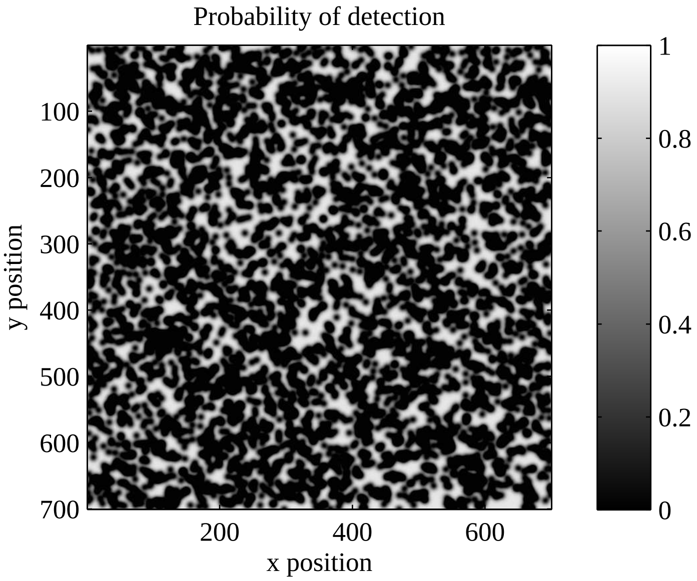


Figure 4.2. Example of randomly generated detection map. The color intensity indicates the probability of detection at each x and y position in the region.

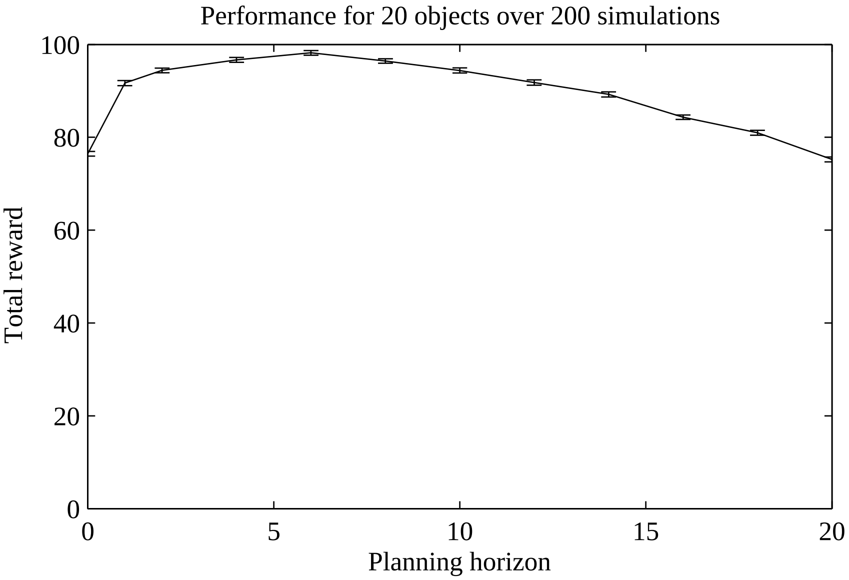


Figure 4.3. Performance tracking $M = 20$ objects. Performance is measured as the average (over the 200 simulations) total change in entropy due to incorporating chosen measurements over all time. The point with a planning horizon of zero corresponds to observing objects sequentially; with a planning horizon of one the auction-based method is equivalent to greedy selection. Error bars indicate 1-σ confidence bounds for the estimate of average total reward.

function is illustrated in Fig. 4.2. The function may be viewed as an obscuration map, *e.g.* due to foliage. Estimation is performed using the Gaussian particle filter [45].

The performance over 200 Monte Carlo runs is illustrated in Fig. 4.3. The point with a planning horizon of zero corresponds to a raster, in which objects are observed sequentially. With a planning horizon of one, the auction-based algorithm corresponds to greedy selection. The performance is measured as the average (over the 200 simulations) total change in entropy due to incorporating chosen measurements over all time. The diagram demonstrates that, with the right choice of planning horizon, the assignment formulation is able to improve performance over the greedy method. The reduction in performance for longer planning horizons is a consequence of the restriction to observe each object at most once in the horizon. If the planning horizon is on the order of the number of objects, we are then, in effect, enforcing that each object *must* be observed once. As illustrated in Fig. 4.1, in this scenario there will often be objects receiving low reward values throughout the planning interval, hence by forcing the controller to observe each object, we are forcing it to (at some stage) take observations of little value. It is not surprising that the increase in performance above the greedy heuristic is small, since the performance guarantees discussed in the previous chapter apply to this scenario.

These limitations motivate the generalization explored in the following section, which allows us to admit multiple observations of each object, as well as observations that require different durations to complete (note that the performance guarantees of Chapter 3 require all observations to consume a single time slot, hence they are not applicable to this wider class of problems).

■ 4.2 Integer programming generalization

An abstraction of the previous analysis replaces the discrete time slots $\{1, \dots, N\}$ with a set of available resources, $\mathcal{R}$ (assumed finite), the elements of which may correspond to the use of a particular sensor over a particular interval of time. As in the previous section, each element of $\mathcal{R}$ can be assigned to at most one task. Unlike the previous section and the previous chapter, the formulation in this section allows us to accommodate observations which consume multiple resources (*e.g.*, multiple time slots on the same sensor, or the same time slot on multiple sensors). We also relax the constraint that each object may be observed at most once, and utilize a more advanced integer programming formulation to find an efficient solution.

4.2.1 Observation sets

Let $\mathcal{U}^i = \{u^i_1, \dots, u^i_{L_i}\}$ be the set of elemental observation actions (assumed finite) that may be used for object i, where each elemental observation u^i_j corresponds to observing object i using a particular mode of a particular sensor within a particular period of time. An elemental action may occupy multiple resources; let $t(u^i_j) \subseteq \mathcal{R}$ be the subset of resource indices consumed by the elemental observation action u^i_j. Let $\mathscr{S}^i \subseteq 2^{\mathcal{U}^i}$ be the collection of observation subsets which we allow for object i. This is assumed to take the form of Eq. (4.12), (4.13) or (4.14); note that in each case it is an independence system, though not necessarily a matroid (as defined in Section 2.4), since observations may consume different resource quantities. If we do not limit the sensing resources that are allowed to be used for object i, the collection will consist of all subsets of $\mathcal{U}^i$ for which no two elements consume the same resource:

$$\mathscr{S}^i = \{\mathcal{A} \subseteq \mathcal{U}^i | t(u_1) \cap t(u_2) = \emptyset \; \forall \, u_1, u_2 \in \mathcal{A}\} \tag{4.12}$$

Alternatively we may limit the total number of elemental observations allowed to be taken for object i to k^i:

$$\mathscr{S}^i = \{\mathcal{A} \subseteq \mathcal{U}^i | t(u_1) \cap t(u_2) = \emptyset \; \forall \, u_1, u_2 \in \mathcal{A}, |\mathcal{A}| \leq k^i\} \tag{4.13}$$

or limit the total quantity of resources allowed to be consumed for object i to R^i:

$$\mathscr{S}^i = \left\{ \mathcal{A} \subseteq \mathcal{U}^i \,\middle|\, t(u_1) \cap t(u_2) = \emptyset \; \forall \, u_1, u_2 \in \mathcal{A}, \sum_{u \in \mathcal{A}} |t(u)| \leq R^i \right\} \tag{4.14}$$

We denote by $t(\mathcal{A}) \subseteq \mathcal{R}$ the set of resources consumed by the actions in set $\mathcal{A}$, *i.e.*,

$$t(\mathcal{A}) = \bigcup_{u \in \mathcal{A}} t(u)$$

The problem that we seek to solve is that of selecting the set of observation actions for each object such that the total reward is maximized subject to the constraint that each resource can be used at most once.

4.2.2 Integer programming formulation

The optimization problem that we seek to solve is reminiscent of the assignment problem in Eq. (4.7), except that now we are assigning to each object a *set* of observations, rather than a single observation:

$$\max_{\omega^i_{\mathcal{A}^i}} \sum_{i=1}^{M} \sum_{\mathcal{A}^i \in \mathscr{S}^i} r^i_{\mathcal{A}^i} \omega^i_{\mathcal{A}^i} \tag{4.15a}$$

$$\text{s.t.} \sum_{i=1}^{M} \sum_{\substack{\mathcal{A}^i \in \mathscr{S}^i \\ t \in t(\mathcal{A}^i)}} \omega^i_{\mathcal{A}^i} \leq 1 \quad \forall\, t \in \mathcal{R} \tag{4.15b}$$

$$\sum_{\mathcal{A}^i \in \mathscr{S}^i} \omega^i_{\mathcal{A}^i} = 1 \quad \forall\, i \in \{1, \dots, M\} \tag{4.15c}$$

$$\omega^i_{\mathcal{A}^i} \in \{0, 1\} \quad \forall\, i,\ \mathcal{A}^i \in \mathscr{S}^i \tag{4.15d}$$

Again, the binary indicator variables $\omega^i_{\mathcal{A}^i}$ are 1 if the observation set $\mathcal{A}^i$ is chosen and 0 otherwise. The interpretation of each line of the integer program follows.

- The objective in Eq. (4.15a) is the sum of the rewards of the subset selected for each object i (*i.e.*, the subsets for which $\omega^i_{\mathcal{A}^i} = 1$).
- The constraints in Eq. (4.15b) ensure that each resource (*e.g.*, sensor time slot) is used at most once.
- The constraints in Eq. (4.15c) ensure that exactly one observation set is chosen for any given object. This is necessary to ensure that the additive objective is the exact reward of corresponding selection (since, in general, $r^i_{\mathcal{A} \cup \mathcal{B}} \neq r^i_{\mathcal{A}} + r^i_{\mathcal{B}}$). Note that the constraint does not force us to take an observation of any object, since the empty observation set is allowed ($\emptyset \in \mathscr{S}^i$) for each object i.
- The integrality constraints in Eq. (4.15d) ensure that the selection variables take on the values zero (not selected) or one (selected).

Unlike the formulation in Eq. (4.7), the integrality constraints in Eq. (4.15d) cannot be relaxed. The problem is not a pure assignment problem, as the observation subsets $\mathcal{A}^i \in \mathscr{S}^i$ consume multiple resources and hence appear in more than one of the constraints defined by Eq. (4.15b). The problem is a bundle assignment problem, and conceptually could be addressed using combinatorial auction methods (*e.g.*, [79]). However, generally this would require computation of $r^i_{\mathcal{A}^i}$ for every subset $\mathcal{A}^i \in \mathscr{S}^i$. If the collections of observation sets $\mathscr{S}^i$, $i \in \{1, \dots, M\}$ allow for several observations to be taken of the same object, the number of subsets may be combinatorially large.

Our approach exploits submodularity to solve a sequence of integer programs, each of which represents the subsets available through a compact representation. The solution of the integer program in each iteration provides an upper bound to the optimal reward, which becomes increasingly tight as iterations progress. The case in which all observations are equivalent, such that the only decision is to determine how many observations to select for each object, could be addressed using [5]. In this thesis, we address the general case which arises when observations are heterogeneous and require different subsets of resources (*e.g.*, different time durations). The complexity associated with evaluating the reward of each of an exponentially large collection of observation sets can be ameliorated using a constraint generation approach, as described in the following section.

■ 4.3 Constraint generation approach

The previous section described a formulation which conceptually could be used to find the optimal observation selection, but the computational complexity of the formulation precludes its utility. This section details an algorithm that can be used to efficiently solve the integer program in many practical situations. The method proceeds by sequentially solving a series of integer programs with progressively greater complexity. In the limit, we arrive at the full complexity of the integer program in Eq. (4.15), but in many practical situations it is possible to terminate much sooner with an optimal (or near-optimal) solution.

The formulation may be conceptually understood as dividing the collection of subsets for each object ($\mathscr{S}^i$) at iteration l into two collections: $\mathscr{T}_l^i \subseteq \mathscr{S}^i$ and the remainder $\mathscr{S}^i \backslash \mathscr{T}_l^i$. The subsets in $\mathscr{T}_l^i$ are those for which the exact reward has been evaluated; we will refer to these as *candidate subsets.*

Definition 4.1 (candidate subset)**.** *The collection of candidate subsets, $\mathscr{T}_l^i \subseteq \mathscr{S}^i$, is the collection subsets of observations for object i for which the exact reward has been evaluated prior to iteration l. New subsets are added to the collection at each iteration (and their rewards calculated), so that $\mathscr{T}_l^i \subseteq \mathscr{T}_{l+1}^i$ for all l. We commence with $\mathscr{T}_l^i = \{\emptyset\}$.*

The reward of each of the remaining subsets (*i.e.*, those in $\mathscr{S}^i \backslash \mathscr{T}_l^i$) has not been evaluated, but an upper bound to each reward is available. In practice, we will not explicitly enumerate the elements in $\mathscr{S}^i \backslash \mathscr{T}_l^i$; rather we use a compact representation which obtains upper bounds through submodularity (details of this will be given later in Lemma 4.3).

In each iteration of the algorithm we solve an integer program, the solution of which selects a subset for each object, ensuring that the resource constraints (*e.g.*, Eq. (4.15b)) are satisfied. If the subset that the integer program selects for each object i is in $\mathscr{T}_l^i$—*i.e.*, it is a subset which had been generated and for which the exact reward had been evaluated in a previous iteration—then we have found an optimal solution to the original problem, *i.e.*, Eq. (4.15). Conversely, if the integer program selects a subset in $\mathscr{S}^i \backslash \mathscr{T}_l^i$ for one or more objects, then we need to tighten the upper bounds on the rewards of those subsets; one way of doing this is to add the newly selected subsets to $\mathscr{T}_l^i$ and evaluate their exact rewards. Each iteration of the optimization reconsiders all decision variables, allowing the solution from the previous iteration to be augmented or reversed in any way.

The compact representation of $\mathscr{S}^i \backslash \mathscr{T}_l^i$ associates with each candidate subset, $\mathcal{A}^i \in \mathscr{T}_l^i$, a subset of observation actions, $\mathcal{B}^i_{l,\mathcal{A}^i}$; $\mathcal{A}^i$ may be augmented with any subset of $\mathcal{B}^i_{l,\mathcal{A}^i}$ to generate new subsets that are not in $\mathscr{T}_l^i$ (but that are in $\mathscr{S}^i$). We refer to $\mathcal{B}^i_{l,\mathcal{A}^i}$ as an *exploration subset*, since it provides a mechanism for discovering promising new subsets that should be incorporated into $\mathscr{T}_{l+1}^i$.

Definition 4.2 (exploration subset)**.** *With each candidate subset $\mathcal{A}^i \in \mathscr{T}_l^i$ we associate an exploration subset $\mathcal{B}^i_{l,\mathcal{A}^i} \subseteq \mathcal{U}^i$. The candidate subset $\mathcal{A}^i$ may be augmented with any subset of elemental observations from $\mathcal{B}^i_{l,\mathcal{A}^i}$ (subject to resource constraints) to generate subsets in $\mathscr{S}^i \backslash \mathscr{T}_l^i$.*

The solution of the integer program at each iteration l is a choice of one candidate subset, $\mathcal{A}^i \in \mathscr{T}_l^i$, for each object i, and a subset of elements of the corresponding exploration subset, $\mathcal{C}^i \subseteq \mathcal{B}^i_{l,\mathcal{A}^i}$. The subset of observations selected by the integer program for object i is the union of these, $\mathcal{A}^i \cup \mathcal{C}^i$.

Definition 4.3 (selection)**.** *The integer program at each iteration l selects a subset of observations, $\mathcal{D}^i \in \mathscr{S}^i$, for each object i. The selected subset, $\mathcal{D}^i$, is indicated indirectly through a choice of one candidate subset, $\mathcal{A}^i$, and a subset of elements from the corresponding exploration subset, $\mathcal{C}^i \subseteq \mathcal{B}^i_{l,\mathcal{A}^i}$ (possibly empty), such that $\mathcal{D}^i = \mathcal{A}^i \cup \mathcal{C}^i$.*

The *update algorithm* (Algorithm 4.1) specifies the way in which the collection of candidate subsets $\mathscr{T}_l^i$ and the exploration subsets $\mathcal{B}^i_{l,\mathcal{A}^i}$ are updated between iterations using the selection results of the integer program. We will prove in Lemma 4.1 that this update procedure ensures that there is exactly one way of selecting each subset

$\mathcal{D}^i \in \mathscr{S}^i$, augmenting a choice of $\mathcal{A}^i \in \mathscr{T}^i_l$ with a subset of elements of the exploration subset $\mathcal{C}^i \subseteq \mathcal{B}^i_{l,\mathcal{A}^i}$.

Definition 4.4 (update algorithm)**.** *The update algorithm takes the result of the integer program at iteration l and determines the changes to make to $\mathscr{T}^i_{l+1}$ (i.e., which new candidate subsets to add), and $\mathcal{B}^i_{l+1,\mathcal{A}^i}$ for each $\mathcal{A}^i \in \mathscr{T}^i_{l+1}$ for iteration $(l+1)$.*

As well as evaluating the reward of each candidate subset $\mathcal{A}^i \in \mathscr{T}^i_l$, we also evaluate the *incremental reward* of each element in the corresponding exploration subset, $\mathcal{B}^i_{l,\mathcal{A}^i}$. The incremental rewards are used to obtain upper bounds on the reward of observation sets in $\mathscr{S}^i \backslash \mathscr{T}^i_l$ generated using candidate subsets and exploration subsets.

Definition 4.5 (incremental reward)**.** *The incremental reward $r_{u|\mathcal{A}^i}$ of an elemental observation $u \in \mathcal{B}^i_{l,\mathcal{A}^i}$ given a candidate subset $\mathcal{A}^i$ is the increase in reward for choosing the single new observation u when the candidate subset $\mathcal{A}^i$ is already chosen:*

$$r^i_{u|\mathcal{A}^i} = r^i_{\mathcal{A}^i \cup \{u\}} - r^i_{\mathcal{A}^i}$$

■ 4.3.1 Example

Consider a scenario involving two objects. There are three observations available for object 1 ($\mathcal{U}^1 = \{a, b, c\}$), and four observations for object 2 ($\mathcal{U}^2 = \{d, e, f, g\}$). There are four resources ($\mathcal{R} = \{\alpha, \beta, \gamma, \delta\}$); observations a, b, and c consume resources α, β and γ respectively, while observations d, e, f and g consume resources α, β, γ and δ respectively. The collection of possible subsets for object i is $\mathscr{S}^i = 2^{\mathcal{U}^i}$.

The subsets involved in iteration l of the algorithm are illustrated in Fig. 4.4. The candidate subsets shown in the circles in the diagram and the corresponding exploration subsets shown in the rectangles attached to the circles are the result of previous iterations of the algorithm. The exact reward of each candidate subset has been evaluated, as has the incremental reward of each element of the corresponding exploration subset. The sets are constructed such that there is a unique way of selecting any subset of observations in $\mathscr{S}^i$.

The integer program at iteration l can choose any candidate subset $\mathcal{A}^i \in \mathscr{T}^i_l$, augmented by any subset of the corresponding exploration subset. For example, for object 1, we could select the subset $\{c\}$ by selecting the candidate subset $\emptyset$ with the exploration subset element $\{c\}$, and for object 2, we could select the subset $\{d, e, g\}$ by choosing the candidate subset $\{g\}$ with the exploration subset elements $\{d, e\}$. The candidate

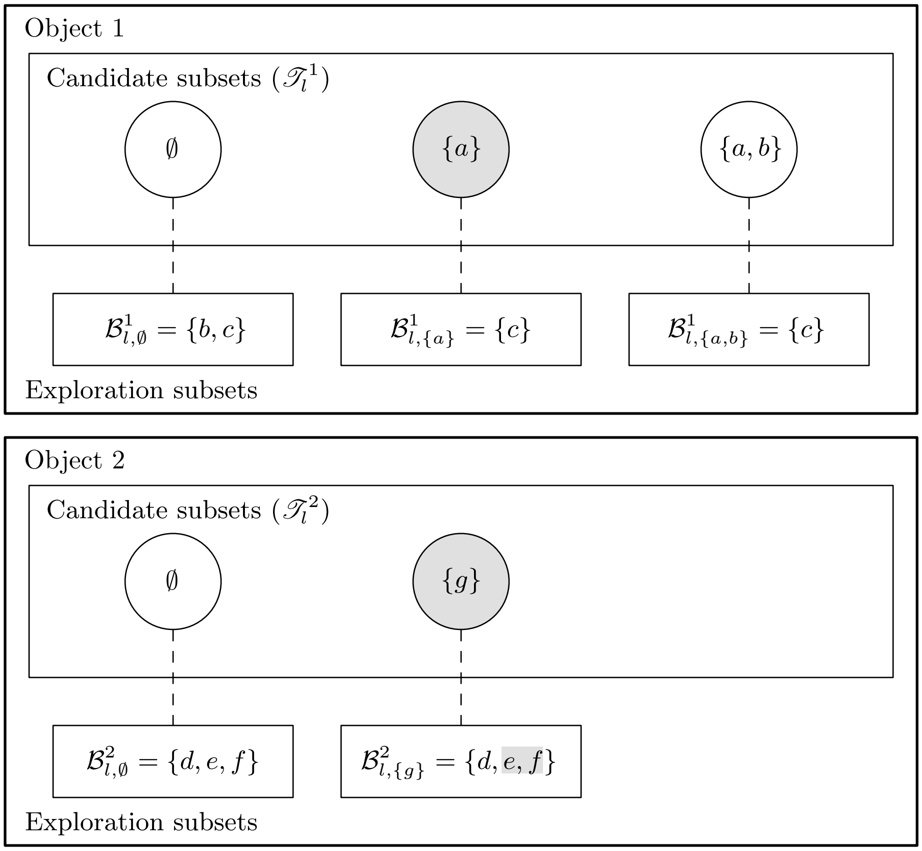

Object 1
Candidate subsets ($\mathscr{T}_l^1$)
$\emptyset$
$\{a\}$
$\{a, b\}$
$\mathcal{B}^1_{l,\emptyset} = \{b, c\}$
$\mathcal{B}^1_{l,\{a\}} = \{c\}$
$\mathcal{B}^1_{l,\{a,b\}} = \{c\}$
Exploration subsets
Object 2
Candidate subsets ($\mathscr{T}_l^2$)
$\emptyset$
$\{g\}$
$\mathcal{B}^2_{l,\emptyset} = \{d, e, f\}$
$\mathcal{B}^2_{l,\{g\}} = \{d, e, f\}$
Exploration subsets

Figure 4.4. Subsets available in iteration l of example scenario. The integer program may select for each object any candidate subset in $\mathscr{T}_l^i$, illustrated by the circles, augmented by any subset of elements from the corresponding exploration subset, illustrated by the rectangle connected to the circle. The sets are constructed such that there is a unique way of selecting any subset of observations in $\mathscr{S}^i$. The subsets selected for each object must collectively satisfy the resource constraints in order to be feasible. The shaded candidate subsets and exploration subset elements denote the solution of the integer program at this iteration.

subsets and exploration subsets are generated using an update algorithm which ensures that there is exactly one way of selecting each subset; *e.g.*, to select the set $\{a, c\}$ for object 1, we must choose candidate subset $\{a\}$ and exploration subset element c; we cannot select candidate subset $\emptyset$ with exploration subset elements $\{a, c\}$ since $a \notin \mathcal{B}^1_{l,\emptyset}$.

We now demonstrate the operation of the update algorithm that is described in detail in Section 4.3.3. Suppose that the solution of the integer program at iteration l selects subset $\{a\}$ for object 1, and subset $\{e, f, g\}$ for object 2 (*i.e.*, the candidate subsets and exploration subset elements that are shaded in Fig. 4.4). Since $\{a\} \in \mathscr{T}^1_l$ the exact reward of the subset selected for object 1 has already been evaluated and we do not need to modify the candidate subsets for object 1, so we simply set $\mathscr{T}^1_{l+1} = \mathscr{T}^1_l$. For object 2, we find that $\{e, f, g\} \notin \mathscr{T}^2_l$, so an update is required. There are many ways that this update could be performed. Our method (Algorithm 4.1) creates a new candidate subset $\tilde{\mathcal{A}}$ consisting of the candidate subset selected for the object ($\{g\}$), augmented by the single element (out of the selected exploration subset elements) with the highest incremental reward. Suppose in our case that the reward of the subset $\{e, g\}$ is greater than the reward of $\{f, g\}$; then $\tilde{\mathcal{A}} = \{e, g\}$.

The subsets in iteration $(l + 1)$ are illustrated in Fig. 4.5. The sets that were modified in the update are shaded in the diagram. There remains a unique way of selecting each subset of observations; *e.g.*, the only way to select elements g and e together (for object 2) is to select the new candidate subset $\{e, g\}$, since element e was removed from the exploration subset for candidate subset $\{g\}$ (*i.e.*, $\mathcal{B}^2_{l+1,\{g\}}$). The procedure that assuring that this is always the case is part of the algorithm which we describe in Section 4.3.3.

■ 4.3.2 Formulation of the integer program in each iteration

The collection of candidate subsets at stage l of the solution is denoted by $\mathscr{T}^i_l \subseteq \mathscr{S}^i$, while the exploration subset corresponding to candidate subset $\mathcal{A}^i \in \mathscr{T}^i_l$ is denoted by $\mathcal{B}^i_{l,\mathcal{A}^i} \subseteq \mathcal{U}^i$. To initialize the problem, we select $\mathscr{T}^i_0 = \{\emptyset\}$, and $\mathcal{B}^i_{0,\emptyset} = \mathcal{U}^i$ for all i. The

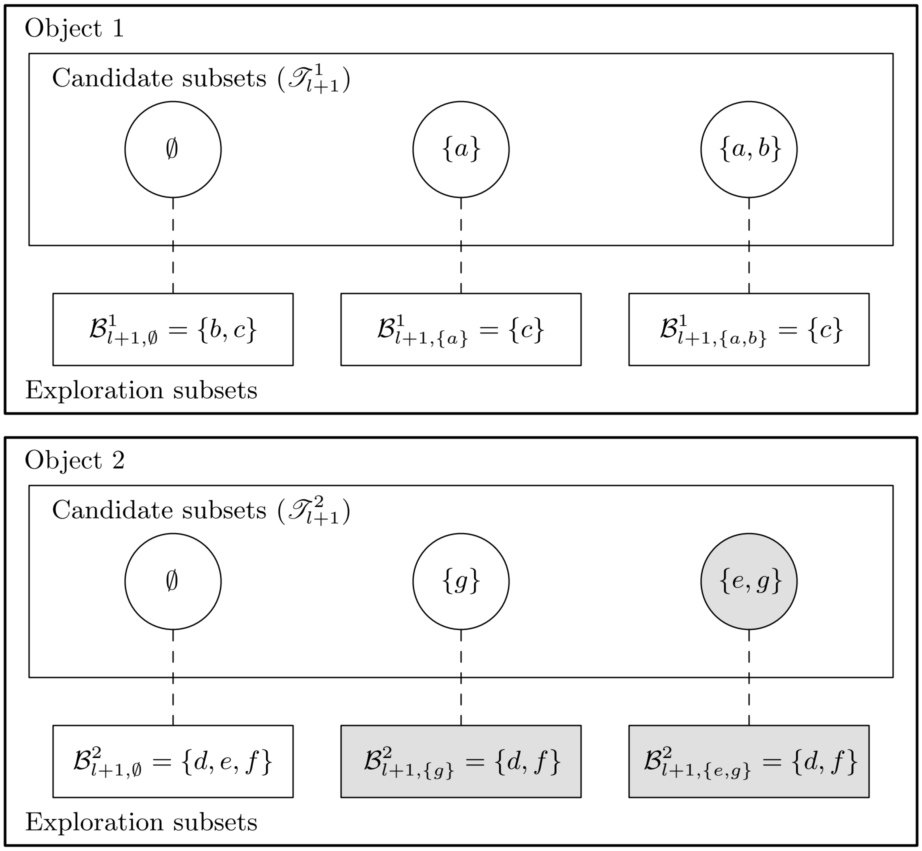

Object 1
Candidate subsets ($\mathscr{T}^1_{l+1}$)
$\emptyset$
$\{a\}$
$\{a,b\}$
$\mathcal{B}^1_{l+1,\emptyset} = \{b,c\}$
$\mathcal{B}^1_{l+1,\{a\}} = \{c\}$
$\mathcal{B}^1_{l+1,\{a,b\}} = \{c\}$
Exploration subsets
Object 2
Candidate subsets ($\mathscr{T}^2_{l+1}$)
$\emptyset$
$\{g\}$
$\{e,g\}$
$\mathcal{B}^2_{l+1,\emptyset} = \{d,e,f\}$
$\mathcal{B}^2_{l+1,\{g\}} = \{d,f\}$
$\mathcal{B}^2_{l+1,\{e,g\}} = \{d,f\}$
Exploration subsets

Figure 4.5. Subsets available in iteration $(l+1)$ of example scenario. The subsets that were modified in the update between iterations l and $(l+1)$ are shaded. There remains a unique way of selecting each subset of observations; *e.g.*, the only way to select elements g and e together (for object 2) is to select the new candidate subset $\{e,g\}$, since element e was removed from the exploration subset for candidate subset $\{g\}$ (*i.e.*, $\mathcal{B}^2_{l+1,\{g\}}$).

integer program that we solve at each stage is:

$$\max_{\omega^i_{\mathcal{A}^i},\,\omega^i_{u|\mathcal{A}^i}} \sum_{i=1}^{M} \sum_{\mathcal{A}^i \in \mathscr{T}^i_l} \left[r^i_{\mathcal{A}^i}\omega^i_{\mathcal{A}^i} + \sum_{u \in \mathcal{B}^i_{l,\mathcal{A}^i}} r^i_{u|\mathcal{A}^i}\omega^i_{u|\mathcal{A}^i} \right] \tag{4.16a}$$

$$\text{s.t.} \sum_{i=1}^{M} \sum_{\substack{\mathcal{A}^i \in \mathscr{T}^i_l \\ t \in t(\mathcal{A}^i)}} \omega^i_{\mathcal{A}^i} + \sum_{i=1}^{M} \sum_{\mathcal{A}^i \in \mathscr{T}^i_l} \sum_{\substack{u \in \mathcal{B}^i_{l,\mathcal{A}^i} \\ t \in t(u)}} \omega^i_{u|\mathcal{A}^i} \leq 1 \quad \forall\, t \in \mathcal{R} \tag{4.16b}$$

$$\sum_{\mathcal{A}^i \in \mathscr{T}^i} \omega^i_{\mathcal{A}^i} = 1 \quad \forall\, i \in \{1, \dots, M\} \tag{4.16c}$$

$$\sum_{u \in \mathcal{B}^i_{l,\mathcal{A}^i}} \omega^i_{u|\mathcal{A}^i} - |\mathcal{B}^i_{l,\mathcal{A}^i}|\omega^i_{\mathcal{A}^i} \leq 0 \quad \forall\, i,\ \mathcal{A}^i \in \mathscr{T}^i_l \tag{4.16d}$$

$$\omega^i_{\mathcal{A}^i} \in \{0, 1\} \quad \forall\, i,\ \mathcal{A}^i \in \mathscr{T}^i \tag{4.16e}$$

$$\omega^i_{u|\mathcal{A}^i} \in \{0, 1\} \quad \forall\, i,\ \mathcal{A}^i \in \mathscr{T}^i,\ u \in \mathcal{B}^i_{l,\mathcal{A}^i} \tag{4.16f}$$

If there is a cardinality constraint of the form of Eq. (4.13) on the maximum number of elemental observations allowed to be used on any given object, we add the constraints:

$$\sum_{\mathcal{A}^i \in \mathscr{T}^i_l} \left[|\mathcal{A}^i|\omega^i_{\mathcal{A}^i} + \sum_{u \in \mathcal{B}^i_{l,\mathcal{A}^i}} \omega^i_{u|\mathcal{A}^i} \right] \leq k^i \quad \forall\, i \in \{1, \dots, M\} \tag{4.16g}$$

Alternatively, if there is a constraint of the form of Eq. (4.14) on the maximum number of elements of the resource set $\mathcal{R}$ allowed to be utilized on any given object, we add the constraints:

$$\sum_{\mathcal{A}^i \in \mathscr{T}^i_l} \left[t(\mathcal{A}^i)\omega^i_{\mathcal{A}^i} + \sum_{u \in \mathcal{B}^i_{l,\mathcal{A}^i}} t(u)\omega^i_{u|\mathcal{A}^i} \right] \leq R^i \quad \forall\, i \in \{1, \dots, M\} \tag{4.16h}$$

The selection variable $\omega^i_{\mathcal{A}^i}$ indicates whether candidate subset $\mathcal{A}^i \in \mathscr{T}^i_l$ is chosen; the constraint in Eq. (4.16c) guarantees that exactly one candidate subset is chosen for each object. The selection variables $\omega^i_{u|\mathcal{A}^i}$ indicate the elements of the exploration subset corresponding to $\mathcal{A}^i$ that are being used to augment the candidate subset. All selection variables are either zero (not selected) or one (selected) due to the integrality constraints in Eq. (4.16e) and Eq. (4.16f). In accordance with Definition 4.3, the solution of the integer program selects for each object i a subset of observations:

Definition 4.6 (selection variables)**.** *The values of the selection variables, $\omega^i_{\mathcal{A}^i}$, $\omega^i_{u|\mathcal{A}^i}$ determine the subsets selected for each object i. The subset selected for object i is $\mathcal{D}^i = \mathcal{A}^i \cup \mathcal{C}^i$ where $\mathcal{A}^i$ is the candidate subset for object i such that $\omega^i_{\mathcal{A}^i} = 1$ and $\mathcal{C}^i = \{u \in \mathcal{B}^i_{l,\mathcal{A}^i} | \omega^i_{u|\mathcal{A}^i} = 1\}$.*

The objective and constraints in Eq. (4.16) may be interpreted as follows:

- The objective in Eq. (4.16a) is the sum of the reward for the candidate subset selected for each object, plus the incremental rewards of any exploration subset elements that are selected. As per Definition 4.5, the reward increment $r^i_{u|\mathcal{A}^i} = (r^i_{\mathcal{A}^i \cup \{u\}} - r^i_{\mathcal{A}^i})$ represents the additional reward for selecting the elemental action u given that the candidate subset $\mathcal{A}^i$ has been selected. We will see in Lemma 4.3 that, due to submodularity, the sum of the reward increments $\sum_u r^i_{u|\mathcal{A}^i}$ is an upper bound for the additional reward obtained for selecting those elements given that the candidate set $\mathcal{A}^i$ has been selected.[1]

- The constraint in Eq. (4.16b) dictates that each resource can only be used once, either by a candidate subset or an exploration subset element; this is analogous with Eq. (4.15b) in the original formulation. The first term includes all candidate subsets that consume resource t, while the second includes all exploration subset elements that consume resource t.

- The constraint in Eq. (4.16c) specifies that exactly one candidate subset should be selected per object. At each solution stage, there will be a candidate subset corresponding to taking no observations ($\mathcal{A}^i = \emptyset$), hence this does not force the system to take an observation of any given object.

- The constraint in Eq. (4.16d) specifies that exploration subset elements which correspond to a given candidate subset can only be chosen if that candidate subset is chosen.

- The integrality constraints in Eq. (4.16e) and Eq. (4.16f) require each variable to be either zero (not selected) or one (selected).

Again, the integrality constraints cannot be relaxed—the problem is not an assignment problem since candidate subsets may consume multiple resources, and there are side constraints (Eq. (4.16d)).

[1] Actually, the reward for selecting a candidate subset and *one* exploration subset variable is a exact reward value; the reward for selecting a candidate subset and two or more exploration subset variables is an upper bound.

4.3.3 Iterative algorithm

Algorithm 4.1 describes the iterative manner in which the integer program in Eq. (4.16) is applied:

```
$\mathscr{T}^i_0 = \emptyset \;\forall\; i$; $\mathcal{B}^i_{0,\emptyset} = \mathcal{U}^i \;\forall\; i$; $l = 0$
evaluate $r^i_{u|\emptyset} \;\forall\; i,\; u \in \mathcal{B}^i_{0,\emptyset}$
solve problem in Eq. (4.16)
while $\exists\; i\; \mathcal{A}^i$ such that $\sum_{u \in \mathcal{B}^i_{l,\mathcal{A}^i}} \omega^i_{u|\mathcal{A}^i} > 1$ do
    for $i \in \{1, \dots, M\}$ do
        let $\mathring{\mathcal{A}}^i_l$ be the unique subset such that $\omega^i_{\mathring{\mathcal{A}}^i_l} = 1$
        if $\sum_{u \in \mathcal{B}^i_{l,\mathring{\mathcal{A}}^i_l}} \omega^i_{u|\mathring{\mathcal{A}}^i_l} \leq 1$ then
            $\mathscr{T}^i_{l+1} = \mathscr{T}^i_l$
            $\mathcal{B}^i_{l+1,\mathcal{A}^i} = \mathcal{B}^i_{l,\mathcal{A}^i} \;\forall\; \mathcal{A}^i \in \mathscr{T}^i_l$
        else
            let $\mathring{u}^i_l = \arg\max_{u \in \mathcal{B}^i_{l,\mathring{\mathcal{A}}^i_l}} r^i_{u|\mathring{\mathcal{A}}^i_l}$
            let $\tilde{\mathcal{A}}^i_l = \mathring{\mathcal{A}}^i_l \cup \{\mathring{u}^i_l\}$
            $\mathscr{T}^i_{l+1} = \mathscr{T}^i_l \cup \{\tilde{\mathcal{A}}^i_l\}$
            $\mathcal{B}^i_{l+1,\tilde{\mathcal{A}}^i_l} = \mathcal{B}^i_{l,\mathring{\mathcal{A}}^i_l} \backslash \{\mathring{u}^i_l\} \backslash \{u \in \mathcal{B}^i_{l,\mathring{\mathcal{A}}^i_l} | \tilde{\mathcal{A}}^i_l \cup \{u\} \notin \mathscr{S}^i\}$
            evaluate $r^i_{\tilde{\mathcal{A}}^i_l} = r^i_{\mathring{\mathcal{A}}^i_l} + r^i_{\mathring{u}|\mathring{\mathcal{A}}^i_l}$
            evaluate $r^i_{u|\tilde{\mathcal{A}}^i_l} \;\forall\; u \in \mathcal{B}^i_{l+1,\tilde{\mathcal{A}}^i_l}$
            $\mathcal{B}^i_{l+1,\mathring{\mathcal{A}}^i_l} = \mathcal{B}^i_{l,\mathring{\mathcal{A}}^i_l} \backslash \{\mathring{u}^i_l\}$
            $\mathcal{B}^i_{l+1,\mathcal{A}^i} = \mathcal{B}^i_{l,\mathcal{A}^i} \;\forall\; \mathcal{A}^i \in \mathscr{T}^i_l,\; \mathcal{A}^i \neq \mathring{\mathcal{A}}^i_l$
        end
    end
    $l = l + 1$
    re-solve problem in Eq. (4.16)
end
```

Algorithm 4.1: Constraint generation algorithm which iteratively utilizes Eq. (4.16) to solve Eq. (4.15).

- In each iteration (l) of the algorithm, the integer program in Eq. (4.16) is re-solved.

If no more than one exploration subset element is chosen for each object, then the rewards of all subsets selected correspond to the exact values (as opposed to upper bounds), the optimal solution has been found (as we will show in Theorem 4.1), and the algorithm terminates; otherwise another iteration of the "while" loop (line 4 of Algorithm 4.1) is executed.

- Each iteration of the "while" loop considers decisions corresponding to each object (i) in turn: (the for loop in line 4)
  - If no more than one exploration subset element is chosen for object i then the reward for that object is an exact value rather than an upper bound, so the collection of candidate subsets ($\mathscr{T}_l^i$) and the exploration subsets ($\mathcal{B}^i_{l,\mathcal{A}^i}$) remain unchanged for that object in the following iteration (lines 8–9).
  - If more than one exploration subset element has been chosen for a given object, then the reward obtained by the integer program for that object is an upper bound to the exact reward (as we show in Lemma 4.3). Thus we generate an additional candidate subset ($\tilde{\mathcal{A}}^i_l$) which augments the previously chosen candidate subset ($\mathring{\mathcal{A}}^i_l$) with the exploration subset element with the highest reward increment ($\mathring{u}^i_l$) (lines 11–13). This greedy exploration is analogous to the greedy heuristic discussed in Chapter 3. Here, rather than using it to make greedy action choices (which would result in loss of optimality), we use it to decide which portion of the action space to explore first. As we will see in Theorem 4.1, this scheme maintains a guarantee of optimality if allowed to run to termination.
  - Exploration subsets allow us to select any subset of observations for which the exact reward has not yet been calculated, using an upper bound to the exact reward. Obviously we want to preclude selection of a candidate subset along with additional exploration subset elements to construct a subset for which the exact reward has already been calculated; the updates of the exploration subsets in lines 14 and 17 achieve this, as shown in Lemma 4.1.

■ 4.3.4 Example

Suppose that there are three objects (numbered $\{1, 2, 3\}$) and three resources ($\mathcal{R} = \{\alpha, \beta, \gamma\}$), and that the reward of the various observation subsets are as shown in Table 4.1. We commence with a single empty candidate subset for each object ($\mathscr{T}_0^i =$

Object	**Subset**	**Resources consumed**	**Reward**
1	$\emptyset$	$\emptyset$	0
1	$\{a\}$	$\{\alpha\}$	2
1	$\{b\}$	$\{\beta\}$	2
1	$\{c\}$	$\{\gamma\}$	2
1	$\{a,b\}$	$\{\alpha,\beta\}$	3
1	$\{a,c\}$	$\{\alpha,\gamma\}$	3
1	$\{b,c\}$	$\{\beta,\gamma\}$	3
1	$\{a,b,c\}$	$\{\alpha,\beta,\gamma\}$	3.5
2	$\emptyset$	$\emptyset$	0
2	$\{d\}$	$\{\beta\}$	0.6
3	$\emptyset$	$\emptyset$	0
3	$\{e\}$	$\{\gamma\}$	0.8

Table 4.1. Observation subsets, resources consumed and rewards for each object in the example shown in Fig. 4.6.

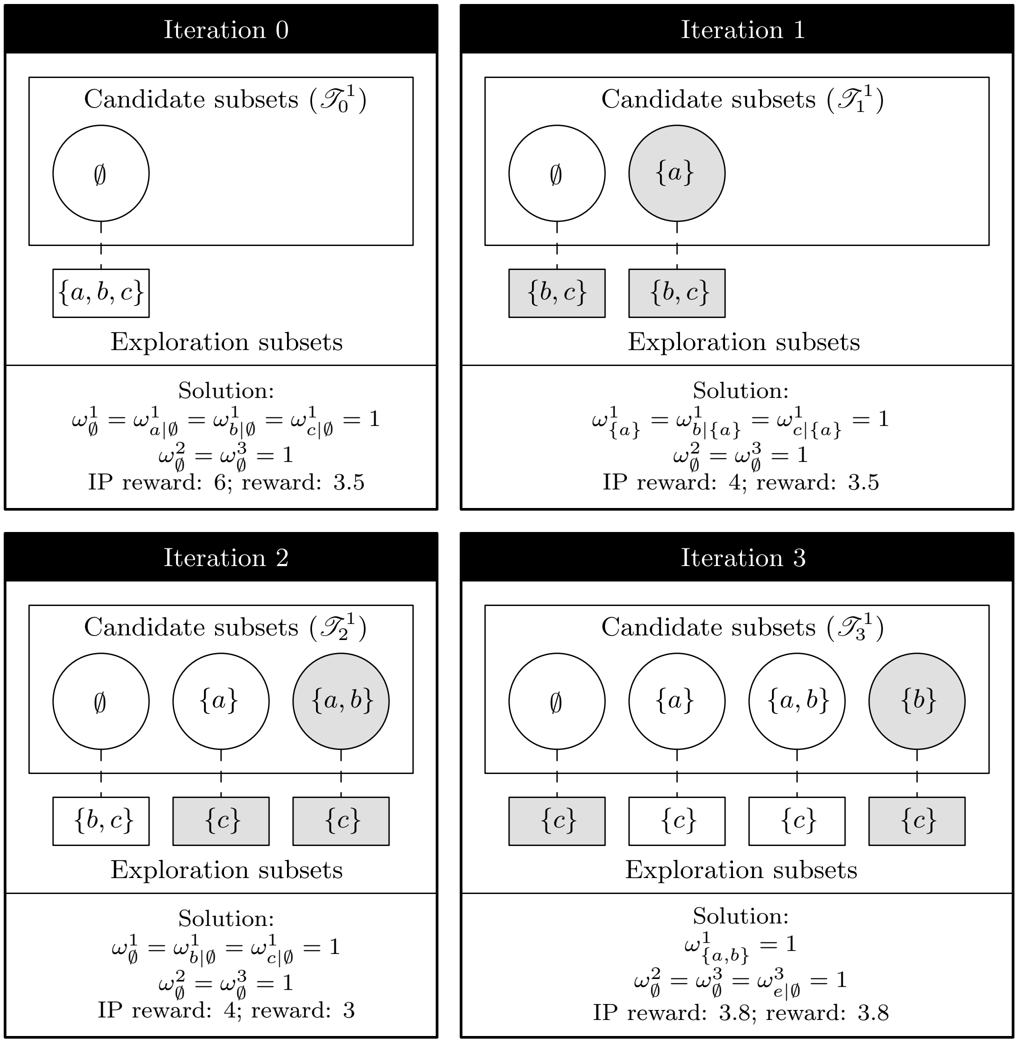

Iteration 0
Candidate subsets ($\mathscr{T}_0^1$)
$\emptyset$
$\{a,b,c\}$
Exploration subsets
Solution:
$\omega^1_\emptyset = \omega^1_{a|\emptyset} = \omega^1_{b|\emptyset} = \omega^1_{c|\emptyset} = 1$
$\omega^2_\emptyset = \omega^3_\emptyset = 1$
IP reward: 6; reward: 3.5
Iteration 1
Candidate subsets ($\mathscr{T}_1^1$)
$\emptyset$
$\{a\}$
$\{b,c\}$
$\{b,c\}$
Exploration subsets
Solution:
$\omega^1_{\{a\}} = \omega^1_{b|\{a\}} = \omega^1_{c|\{a\}} = 1$
$\omega^2_\emptyset = \omega^3_\emptyset = 1$
IP reward: 4; reward: 3.5
Iteration 2
Candidate subsets ($\mathscr{T}_2^1$)
$\emptyset$
$\{a\}$
$\{a,b\}$
$\{b,c\}$
$\{c\}$
$\{c\}$
Exploration subsets
Solution:
$\omega^1_\emptyset = \omega^1_{b|\emptyset} = \omega^1_{c|\emptyset} = 1$
$\omega^2_\emptyset = \omega^3_\emptyset = 1$
IP reward: 4; reward: 3
Iteration 3
Candidate subsets ($\mathscr{T}_3^1$)
$\emptyset$
$\{a\}$
$\{a,b\}$
$\{b\}$
$\{c\}$
$\{c\}$
$\{c\}$
$\{c\}$
Exploration subsets
Solution:
$\omega^1_{\{a,b\}} = 1$
$\omega^2_\emptyset = \omega^3_\emptyset = \omega^3_{e|\emptyset} = 1$
IP reward: 3.8; reward: 3.8

Figure 4.6. Four iterations of operations performed by Algorithm 4.1 on object 1 (arranged in counter-clockwise order, from the top-left). The circles in each iteration show the candidate subsets, while the attached rectangles show the corresponding exploration subsets. The shaded circles and rectangles in iterations 1, 2 and 3 denote the sets that were updated prior to that iteration. The solution to the integer program in each iteration is shown along with the reward in the integer program objective ("IP reward"), which is an upper bound to the exact reward, and the exact reward of the integer program solution ("reward").

$\{\emptyset\}$). The diagram in Fig. 4.6 illustrates the operation of the algorithm in this scenario:

- In iteration $l = 0$, the integer program selects the three observations for object 1 (*i.e.*, choosing the candidate subset $\emptyset$, and the three exploration subset elements $\{a, b, c\}$), and no observations for objects 2 and 3, yielding an upper bound for the reward of 6 (the incremental reward of each exploration subset element from the empty set is 2). The exact reward of this configuration is 3.5. Selection of these exploration subset elements indicates that subsets involving them should be explored further, hence we create a new candidate subset $\tilde{\mathcal{A}}^1_0 = \{a\}$ (the element with the highest reward increment, breaking ties arbitrarily).

- Now, in iteration $l = 1$, the incremental reward of observation b or c conditioned on the candidate subset $\{a\}$ is 1, and the optimal solution to the integer program in iteration is still to select these three observations for object 1, but to do so it is now necessary to select candidate subset $\{a\} \in \mathscr{T}^1_1$ together with the exploration subset elements $\{b, c\}$. No observations are chosen for objects 2 and 3. The upper bound to the reward provided by the integer program is now 4, which is substantially closer to the exact reward of the configuration (3.5). Again we have two exploration subset elements selected (which is why the reward in the integer program is not the exact reward), so we introduce a new candidate subset $\tilde{\mathcal{A}}^1_1 = \{a, b\}$.

- The incremental reward of observation c conditioned on the new candidate subset $\{a, b\}$ is 0.5. The optimal solution to the integer program at iteration $l = 2$ is then to select object 1 candidate subset $\emptyset$ and exploration subset elements $\{b, c\}$, and no observations for objects 2 and 3. The upper bound to the reward provided by the integer program remains 4, but the exact reward of the configuration is reduced to 3 (note that the exact reward of the solution to the integer program at each iteration is not monotonic). Once again there are two exploration subset elements selected, so we introduce a new candidate subset $\tilde{\mathcal{A}}^1_2 = \{b\}$.

- The optimal solution to the integer program in iteration $l = 3$ is then the true optimal configuration, selecting observations a and b for object 1 (*i.e.*, candidate subset $\{a, b\}$ and no exploration subset elements), no observations for object 2, and observation e for object 3 (*i.e.*, candidate subset $\emptyset$ and exploration subset element $\{e\}$). Since no more than one exploration subset element is chosen for each object, the algorithm knows that it has found the optimal solution and thus terminates.

■ 4.3.5 Theoretical characteristics

We are now ready to prove a number of theoretical characteristics of our algorithm. Our goal is to prove that the algorithm terminates in finite time with an optimal solution; Theorem 4.1 establishes this result. Several intermediate results are obtained along the way. Lemma 4.1 proves that there is a unique way of selecting each subset in $\mathscr{S}^i$ in each iteration, while Lemma 4.2 proves that no subset that is not in $\mathscr{S}^i$ can be selected; together, these two results establish that the collection of subsets from which we may select remains the same (*i.e.*, $\mathcal{S}^i$) in every iteration. Lemma 4.3 establishes that the reward in the integer program of any selection is an upper bound to the exact reward of that selection, and that the upper bound is monotonically non-increasing with iteration. Lemma 4.4 establishes that the reward in the integer program of the solution obtained upon termination of Algorithm 4.1 is an exact value (rather than an upper bound). This leads directly to the final result in Theorem 4.1, that the algorithm terminates with an optimal solution.

Before we commence, we prove a simple proposition that allows us to represent selection of a feasible observation subset in two different ways.

Proposition 4.1. *If $\mathcal{C} \in \mathscr{S}^i$, $\mathcal{A}^i \in \mathscr{T}^i_l$ and $\mathcal{A}^i \subseteq \mathcal{C} \subseteq \mathcal{A}^i \cup \mathcal{B}^i_{l,\mathcal{A}^i}$ then the configuration:*

$$\omega^i_{\mathcal{A}^i} = 1$$
$$\omega_{u|\mathcal{A}^i} = \begin{cases} 1, & u \in \mathcal{C} \backslash \mathcal{A}^i \\ 0, & \textit{otherwise} \end{cases}$$

is feasible (provided that the required resources are not consumed on other objects), selects subset $\mathcal{C}$ for object i, and is the unique configuration doing so that has $\omega^i_{\mathcal{A}^i} = 1$. Conversely, if a feasible selection variable configuration for object i selects $\mathcal{C}$, and $\omega^i_{\mathcal{A}^i} = 1$, then $\mathcal{A}^i \subseteq \mathcal{C} \subseteq \mathcal{A}^i \cup \mathcal{B}^i_{\mathcal{A}^i}$.

Proof. Since $\mathcal{C} \in \mathscr{S}^i$, no two resources in $\mathcal{C}$ utilize the same resource. Hence the configuration is feasible, assuming that the required resources are not consumed on other objects. That the configuration selects $\mathcal{C}$ is immediate from Definition 4.6. From Algorithm 4.1, it is clear that $\mathcal{A}^i \cap \mathcal{B}^i_{l,\mathcal{A}^i} = \emptyset \; \forall \; l,\; \mathcal{A}^i \in \mathscr{T}^i_l$ (it is true for $l = 0$, and every time a new candidate subset $\mathcal{A}^i$ is created by augmenting an existing candidate subset with a new element, the corresponding element is removed from $\mathcal{B}^i_{l,\mathcal{A}^i}$). Hence the configuration given is the unique configuration selecting for which $\omega^i_{\mathcal{A}^i} = 1$. The converse results directly from Definition 4.6. In each case, note that all other selection variables for object i must be zero due to the constraints in Eq. (4.16c) and Eq. (4.16d). □

The following lemma establishes that the updates in Algorithm 4.1 maintain a unique way of selecting observation subset in $\mathscr{S}^i$ in each iteration.

Lemma 4.1. *In every iteration l, for any object i and any subset $\mathcal{C} \in \mathscr{S}^i$ there is exactly one configuration of selection variables for the i-th object ($\omega^i_{\mathcal{A}^i}, \omega^i_{u|\mathcal{A}^i}$) that selects observation subset $\mathcal{C}$.*

Proof. In the first iteration ($l = 0$), $\mathscr{T}^i_0 = \{\emptyset\}$, hence the only configuration selecting set $\mathcal{C}$ is that obtained from Proposition 4.1 with $\mathcal{A}^i = \emptyset$.

Now suppose that the lemma holds for some l; we will show that it also holds for $(l+1)$. Given any $\mathcal{C} \in \mathscr{S}^i$, there exists a unique $\mathcal{A}^\dagger \in \mathscr{T}^i_l$ such that $\mathcal{A}^\dagger \subseteq \mathcal{C} \subseteq \mathcal{A}^\dagger \cup \mathcal{B}^i_{l,\mathcal{A}^\dagger}$ by the induction hypothesis and Proposition 4.1. If $\mathcal{A}^\dagger = \mathring{\mathcal{A}}^i_l$ and $\mathring{u}^i_l \in \mathcal{C}$ then $\tilde{\mathcal{A}}^i_l \subseteq \mathcal{C} \subseteq \tilde{\mathcal{A}}^i_l \cup \mathcal{B}^i_{l+1,\tilde{\mathcal{A}}^i_l}$, and it is not the case that $\mathcal{A}^i \subseteq \mathcal{C} \subseteq \mathcal{A}^i \cup \mathcal{B}^i_{l+1,\mathcal{A}^i}$ for any other $\mathcal{A}^i \in \mathscr{T}^i_{l+1}$ (it is not the case for $\mathring{\mathcal{A}}^i_l$ since $\mathring{u}^i_l \notin \mathring{\mathcal{A}}^i_l \cup \mathcal{B}^i_{l+1,\mathring{\mathcal{A}}^i_l}$, and it is not the case for any other $\mathcal{A}^i \in \mathscr{T}^i_l$ since $\mathcal{B}^i_{l+1,\mathcal{A}^i} = \mathcal{B}^i_{l,\mathcal{A}^i}$ and the condition did not hold for iteration l by the induction hypothesis). Hence exactly one selection variable configuration selects $\mathcal{C}$.

If $\mathcal{A}^\dagger = \mathring{\mathcal{A}}^i_l$ and $\mathring{u}^i_l \notin \mathcal{C}$ then $\mathring{\mathcal{A}}^i_l \subseteq \mathcal{C} \subseteq \mathring{\mathcal{A}}^i_l \cup \mathcal{B}^i_{l+1,\mathring{\mathcal{A}}^i_l}$, and it is not the case that $\mathcal{A}^i \subseteq \mathcal{C} \subseteq \mathcal{A}^i \cup \mathcal{B}^i_{l+1,\mathcal{A}^i}$ for any other $\mathcal{A}^i \in \mathscr{T}^i_{l+1}$ (it is not the case for $\tilde{\mathcal{A}}^i_l$ since $\tilde{\mathcal{A}}^i_l \nsubseteq \mathcal{C}$, and it is not the case for any other $\mathcal{A}^i \in \mathscr{T}^i_l$ since $\mathcal{B}^i_{l+1,\mathcal{A}^i} = \mathcal{B}^i_{l,\mathcal{A}^i}$ and the condition did not hold for iteration l by the induction hypothesis). Hence exactly one selection variable configuration selects $\mathcal{C}$.

Finally, if $\mathcal{A}^\dagger \neq \mathring{\mathcal{A}}^i_l$ then $\mathcal{A}^\dagger \in \mathscr{T}^i_{l+1}$ and $\mathcal{A}^\dagger \subseteq \mathcal{C} \subseteq \mathcal{A}^\dagger \cup \mathcal{B}^i_{l+1,\mathcal{A}^\dagger}$. Since it is not the case that $\mathring{\mathcal{A}}^i_l \subseteq \mathcal{C} \subseteq \mathring{\mathcal{A}}^i_l \cup \mathcal{B}^i_{l,\mathring{\mathcal{A}}^i_l}$, it will also not be the case that $\mathring{\mathcal{A}}^i_l \subseteq \mathcal{C} \subseteq \mathring{\mathcal{A}}^i_l \cup \mathcal{B}^i_{l+1,\mathring{\mathcal{A}}^i_l}$ or $\tilde{\mathcal{A}}^i_l \subseteq \mathcal{C} \subseteq \tilde{\mathcal{A}}^i_l \cup \mathcal{B}^i_{l+1,\tilde{\mathcal{A}}^i_l}$, since each of these conditions represents a tightening of the original condition, which was already false. It will also not be the case that $\mathcal{A}^i \subseteq \mathcal{C} \subseteq \mathcal{A}^i \cup \mathcal{B}^i_{l+1,\mathcal{A}^i}$ for any other $\mathcal{A}^i \in \mathscr{T}^i_l$, $\mathcal{A}^i \neq \mathcal{A}^\dagger$ since $\mathcal{B}^i_{l+1,\mathcal{A}^i} = \mathcal{B}^i_{l,\mathcal{A}^i}$.

This shows that the lemma holds for $(l + 1)$. By induction, it must then hold for every iteration. □

The following result establishes that only subsets in $\mathscr{S}^i$ can be generated by the integer program. Combined with the previous result, this establishes that the subsets in $\mathscr{S}^i$ and the configurations of the selection variables in the integer program for object i are in a bijective relationship at every iteration.

Lemma 4.2. *In every iteration l, any feasible selection of a subset for object i, $\mathcal{D}^i$, is in $\mathscr{S}^i$.*

Proof. Suppose, for contradiction, that there is a subset $\mathcal{D}^i$ that is feasible for object i (*i.e.*, there exists a feasible configuration of selection variables with $\omega^i_{\mathcal{A}^i} = 1$ and $\mathcal{C}^i = \{u \in \mathcal{B}^i_{l,\mathcal{A}^i} | \omega^i_{l,\mathcal{A}^i} = 1\}$, such that $\mathcal{D}^i = \mathcal{A}^i \cup \mathcal{C}^i$), but that $\mathcal{D}^i \notin \mathscr{S}^i$. Assume that $\mathscr{S}^i$ is of the form of Eq. (4.12). Since $\mathcal{D}^i \notin \mathscr{S}^i$, there must be at least one resource in $\mathcal{D}^i$ that is used twice. This selection must then be infeasible due to the constraint in Eq. (4.16b), yielding a contradiction. If $\mathscr{S}^i$ is of the form of Eq. (4.13) or (4.14), then a contradiction will be obtained from Eq. (4.16b), (4.16g) or (4.16h). □

The following lemma uses submodularity to show that the reward in the integer program for selecting any subset at any iteration is an upper bound to the exact reward and, furthermore, that the bound tightens as more iterations are performed. This is a key result for proving the optimality of the final result when the algorithm terminates. As with the analysis in Chapter 3, the result is derived in terms of mutual information, but applies to any non-decreasing, submodular reward function.

Lemma 4.3. *The reward associated with selecting observation subset $\mathcal{C}$ for object i in the integer program in Eq. (4.16) is an upper bound to the exact reward of selecting that subset in every iteration. Furthermore, the reward for selecting subset $\mathcal{C}$ for object i in the integer program in iteration l_2 is less than or equal to the reward for selecting $\mathcal{C}$ in l_1 for any $l_1 < l_2$.*

Proof. Suppose that subset $\mathcal{C}$ is selected for object i in iteration l. Let $\mathring{\mathcal{A}}^i_l$ be the subset such that $\omega^i_{\mathring{\mathcal{A}}^i_l} = 1$. Introducing an arbitrary ordering $\{u_1, \dots, u_n\}$ of the elements of $\mathcal{B}^i_{l,\mathring{\mathcal{A}}^i_l}$ for which $\omega^i_{u_i|\mathring{\mathcal{A}}^i_l} = 1$ (*i.e.*, $\mathcal{C} = \mathcal{A}^i_l \cup \{u_1, \dots, u_n\}$), the exact reward for selecting observation subset $\mathcal{C}$ is:

$$\begin{aligned}
r^i_{\mathcal{C}} &\triangleq I(X^i; z^i_{\mathcal{C}}) \\
&\overset{(a)}{=} I(X^i; z^i_{\mathring{\mathcal{A}}^i_l}) + \sum_{j=1}^{n} I(X^i; z^i_{u_j} | \mathring{\mathcal{A}}^i_l, z^i_{u_1}, \dots, z^i_{u_{j-1}}) \\
&\overset{(b)}{\leq} I(X^i; z^i_{\mathring{\mathcal{A}}^i_l}) + \sum_{j=1}^{n} I(X^i; z^i_{u_j} | z^i_{\mathring{\mathcal{A}}^i_l}) \\
&\overset{(c)}{=} r^i_{\mathring{\mathcal{A}}^i_l} + \sum_{j=1}^{n} r^i_{u_j|\mathring{\mathcal{A}}^i_l}
\end{aligned}$$

where (a) is an application of the chain rule, (b) results from submodularity, and (c) results from the definition of $r^i_{i|\mathcal{A}^i}$ and conditional mutual information. This establishes the first result, that the reward for selecting any observation set in any iteration of the integer program is an upper bound to the exact reward of that set.

Now consider the change which occurs between iteration l and iteration $(l+1)$. Suppose that the set updated for object i in iteration l, $\mathring{\mathcal{A}}^i_l$, is the unique set (by Lemma 4.1) for which $\mathring{\mathcal{A}}^i_l \subseteq \mathcal{C} \subseteq \mathring{\mathcal{A}}^i_l \cup \mathcal{B}^i_{l,\mathring{\mathcal{A}}^i_l}$, and that $\mathring{u}^i_l \in \mathcal{C}$. Assume without loss of generality that the ordering $\{u_1, \dots, u_n\}$ from the previous stage was chosen such that $u_n = \mathring{u}^i_l$. Then the reward for selecting subset $\mathcal{C}$ at stage $(l+1)$ will be:

$$\begin{aligned}
r^i_{\tilde{\mathcal{A}}^i_l} + \sum_{j=1}^{n-1} (r^i_{\tilde{\mathcal{A}}^i_l \cup \{u_j\}} - r^i_{\tilde{\mathcal{A}}^i_l}) &\overset{(a)}{=} I(X^i; z^i_{\tilde{\mathcal{A}}^i_l}) + \sum_{j=1}^{n-1} I(X^i; z^i_{u_j} | z^i_{\tilde{\mathcal{A}}^i_l}) \\
&\overset{(b)}{=} I(X^i; z^i_{\mathring{\mathcal{A}}^i_l}) + I(X^i; z^i_{\mathring{u}^i_l} | z^i_{\mathring{\mathcal{A}}^i_l}) \\
&\qquad + \sum_{j=1}^{n-1} I(X^i; z^i_{u_j} | z^i_{\mathring{\mathcal{A}}^i_l}, z^i_{u_n}) \\
&\overset{(c)}{\leq} I(X^i; z^i_{\mathring{\mathcal{A}}^i_l}) + \sum_{j=1}^{n} I(X^i; z^i_{u_j} | z^i_{\mathring{\mathcal{A}}^i_l}) \\
&\overset{(d)}{=} r^i_{\mathring{\mathcal{A}}^i_l} + \sum_{j=1}^{n} r^i_{u_j|\mathring{\mathcal{A}}^i_l}
\end{aligned}$$

where (a) and (d) result from the definition or $r^i_{\mathcal{A}}$ and conditional mutual information, (b) from the chain rule, and (c) from submodularity. The form in (d) is the reward for selecting $\mathcal{C}$ at iteration l.

If it is not true that $\mathring{\mathcal{A}}^i_l \subseteq \mathcal{C} \subseteq \mathring{\mathcal{A}}^i_l \cup \mathcal{B}^i_{l,\mathring{\mathcal{A}}^i_l}$, or if $\mathring{u}^i_l \notin \mathcal{C}$, then the configuration selecting set $\mathcal{C}$ in iteration $(l+1)$ will be identical to the configuration in iteration l, and the reward will be unchanged.

Hence we have found that the reward for selecting subset $\mathcal{C}$ at iteration $(l+1)$ is less than or equal than the reward for selecting $\mathcal{C}$ at iteration l. By induction we then obtain the second result. □

At this point we have established that, at each iteration, we have available to us a unique way of selecting each subset in same collection of subsets, $\mathcal{S}^i$; that in each iteration the reward in the integer program for selecting any subset is an upper bound to the exact reward of that subset; and that the upper bound becomes increasingly tight as iterations proceed. These results directly provide the following corollary.

Corollary 4.1. *The reward achieved in the integer program in Eq. (4.16) at each iteration is an upper bound to the optimal reward, and is monotonically non-increasing with iteration.*

The last result that we need before we prove the final outcome is that the reward in the integer program for each object in the terminal iteration of Algorithm 4.1 is the exact reward of the selection chosen for that object.

Lemma 4.4. *At termination, the reward in the integer program for the subset $\mathcal{D}^i$ selected for object i is the exact reward of that subset.*

Proof. In accordance with Definition 4.6, let $\mathcal{A}^i$ be the subset such that $\omega^i_{\mathcal{A}^i} = 1$, and let $\mathcal{C}^i = \{u \in \mathcal{B}^i_{l,\mathcal{A}^i} | \omega_{u|\mathcal{A}^i} = 1\}$, so that $\mathcal{D}^i = \mathcal{A}^i \cup \mathcal{C}^i$. Since the algorithm terminated at iteration l, we know that (from line 4 of Algorithm 4.1)

$$|\mathcal{C}^i| = \sum_{u \in \mathcal{B}^i_{l,\mathcal{A}^i}} \omega^i_{u|\mathcal{A}^i} \leq 1$$

Hence either no exploration subset element is chosen, or one exploration subset element is chosen. If no exploration subset element is chosen, then $\mathcal{A}^i = \mathcal{D}^i$, and the reward obtained by the integer program is simply $r^i_{\mathcal{A}^i}$, the exact value. If an exploration subset element u is chosen (*i.e.*, $\mathcal{C}^i = \{u\}$), then the reward obtained by the integer program is

$$r^i_{\mathcal{A}^i} + r^i_{u|\mathcal{A}^i} = r^i_{\mathcal{A}^i} + (r^i_{\mathcal{A}^i \cup \{u\}} - r^i_{\mathcal{A}^i}) = r^i_{\mathcal{A}^i \cup \{u\}} = r^i_{\mathcal{D}^i}$$

which, again, is the exact value. Since the objects are independent, the exact overall reward is the sum of the rewards of each object, which is the objective of the integer program. □

We now utilize these outcomes to prove our main result.

Theorem 4.1. *Algorithm 4.1 terminates in finite time with an optimal solution.*

Proof. To establish that the algorithm terminates, note that the number of observation subsets in the collection $\mathscr{S}^i$ is finite for each object i. In every iteration, we add a new subset $\mathcal{A}^i \in \mathscr{S}^i$ into the collection of candidate subsets $\mathscr{T}_l^i$ for some object i. If no such subset exists, the algorithm must terminate, hence finite termination is guaranteed.

Since the reward in the integer program for the subset selected for each object is the exact reward for that set (by Lemma 4.4), and the rewards for all other other subsets are upper bounds to the exact rewards (by Lemma 4.3), the solution upon termination is optimal. □

■ 4.3.6 Early termination

Note that, while the algorithm is guaranteed to terminate finitely, the complexity may be combinatorially large. It may be necessary in some situations to evaluate the reward of every observation subset in $\mathscr{S}^i$ for some objects. At each iteration, the reward obtained by the integer program is an upper bound to the optimal reward, hence if we evaluate the exact reward of the solution obtained at each iteration, we can obtain a guarantee on how far our existing solutions are from optimality, and decide whether to continue processing or terminate with a near-optimal solution. However, while the reward in the integer program is a monotonically non-increasing function of iteration number, the exact reward of the subset selected by the integer program in each iteration may increase or decrease as iterations progress. This was observed in the the example in Section 4.3.4: the exact reward of the optimal solution in iteration $l = 1$ was 3.5, while the exact reward of the optimal solution in iteration $l = 2$ was 3.

It may also be desirable at some stage to terminate the iterative algorithm and select the best observation subset amongst the subsets for which the reward has been evaluated. This can be achieved through a final execution of the integer program in Eq. (4.16), adding the following constraint:

$$\sum_{\mathcal{A}^i \in \mathscr{T}_l^i} \sum_{u \in \mathcal{B}^i_{l,\mathcal{A}^i}} \omega^i_{u|\mathcal{A}^i} \leq 1 \quad \forall\, i \in \{1, \dots, M\} \tag{4.17}$$

Since we can select no more than one exploration subset element for each object, the reward given for any feasible selection will be the exact reward (as shown in Lemma 4.4).

The reward that can be obtained by this augmented integer program is a non-decreasing function of iteration number, since the addition of new candidate subsets yields a wider range of subsets that can be selected without using multiple exploration actions.

4.4 Computational experiments

The experiments in the following sections illustrate the utility of our method. We commence in Section 4.4.1 by describing our implementation of the algorithm. Section 4.4.2 examines a scenario involving surveillance of multiple objects using two moving radar platforms. Section 4.4.3 extends this example to incorporate additional non-stationarity (through observation noise which increases when objects are closely spaced), as well as observations consuming different numbers of time slots. The scenario discussed in Section 4.4.4 was constructed to demonstrate the increase in performance that is possible due to additional planning when observations all consume a single time slot (so that the guarantees of Chapter 3 apply). Finally, the scenario discussed in Section 4.4.5 was constructed to demonstrate the increase in performance which is possible due to additional planning when observations consume different numbers of time slots.

4.4.1 Implementation notes

The implementation utilized in the experiments in this section was written in C++, solving the integer programs using ILOG® CPLEX® 10.1 through the callable library interface. In each iteration of Algorithm 4.1, we solve the integer program, terminating when we find a solution that is guaranteed to be within 98% of optimality, or after 15 seconds of computation time have elapsed. Before commencing the next iteration, we solve the augmented integer program described in Section 4.3.6, again terminating when we find a solution that is guaranteed to be within 98% of optimality or after 15 seconds. Termination occurs when the solution of the augmented integer program is guaranteed to be within 95% of optimality,[2] or after 300 seconds have passed. The present implementation of the planning algorithm is limited to addressing linear Gaussian models. We emphasize that this is not a limitation of the planning algorithm; the assumption merely simplifies the computations involved in reward function evaluations.

Unless otherwise stated, all experiments utilized an open loop feedback control strategy (as discussed in Section 2.2.2). Under this scheme, at each time step a plan was constructed for the next N-step planning horizon, the first step was executed, the re-

[2] The guarantee is obtained by accessing the upper bound found by CPLEX® to the optimal reward of the integer program in Algorithm 4.1.

sulting observations were incorporated, and then a new plan was constructed for the following N-step horizon.

The computation times presented in the following sections include only the computation time involved in the solution of the integer program in Algorithm 4.1. The computation time expended in the augmented integer program of Section 4.3.6 are excluded as the results of these computations are not used as inputs to the following iterations. As such, the augmented integer program could easily be executed on a second processor core in a modern parallel processing architecture. All times were measured using a 3.06 GHz Intel® Xeon® processor.

4.4.2 Waveform selection

Our first example models surveillance of multiple objects by two moving radar platforms. The platforms move along fixed racetrack patterns, as shown in Fig. 4.7. We denote by $\boldsymbol{y}_k^i$ the state (*i.e.*, position and velocity) of platform i at time k. There are M objects under track, the states of which evolve according to the nominally constant velocity model described in Eq. (2.8), with $\Delta t = 0.03$ sec and $q = 1$. The simulation length is 200 steps; the sensor platforms complete 1.7 revolutions of the racetrack pattern in Fig. 4.7 in this time.[3] The initial positions of objects are distributed uniformly in the region $[10, 100] \times [10, 100]$; the initial velocities in each direction are drawn from a Gaussian distribution with mean zero and standard deviation 0.25. The initial estimates are set to the true state, corrupted by additive Gaussian noise with zero mean and standard deviation 1 (in position states) and 0.1 (in velocity states).

In each time slot, each sensor may observe one of the M objects, obtaining either an azimuth and range observation, or an azimuth and range rate observation, each of which occupies a single time slot:

$$\boldsymbol{z}_k^{i,j,r} = \begin{bmatrix} \tan^{-1}\left(\frac{[\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_3}{[\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_1}\right) \\ \sqrt{([\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_1)^2 + ([\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_3)^2} \end{bmatrix} + \begin{bmatrix} b(\boldsymbol{x}_k^i, \boldsymbol{y}_k^j) & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{v}_k^{i,j,r} \tag{4.18}$$

$$\boldsymbol{z}_k^{i,j,d} = \begin{bmatrix} \tan^{-1}\left(\frac{[\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_3}{[\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_1}\right) \\ \frac{[\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_1[\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_2 + [\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_3[\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_4}{\sqrt{([\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_1)^2 + ([\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_3)^2}} \end{bmatrix} + \begin{bmatrix} b(\boldsymbol{x}_k^i, \boldsymbol{y}_k^j) & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{v}_k^{i,j,d} \tag{4.19}$$

where $\boldsymbol{z}_k^{i,j,r}$ denotes the azimuth/range observation for object i using sensor j at time k,

[3] The movement of the sensor is accentuated in order to create some degree of non-stationarity in the sensing model.

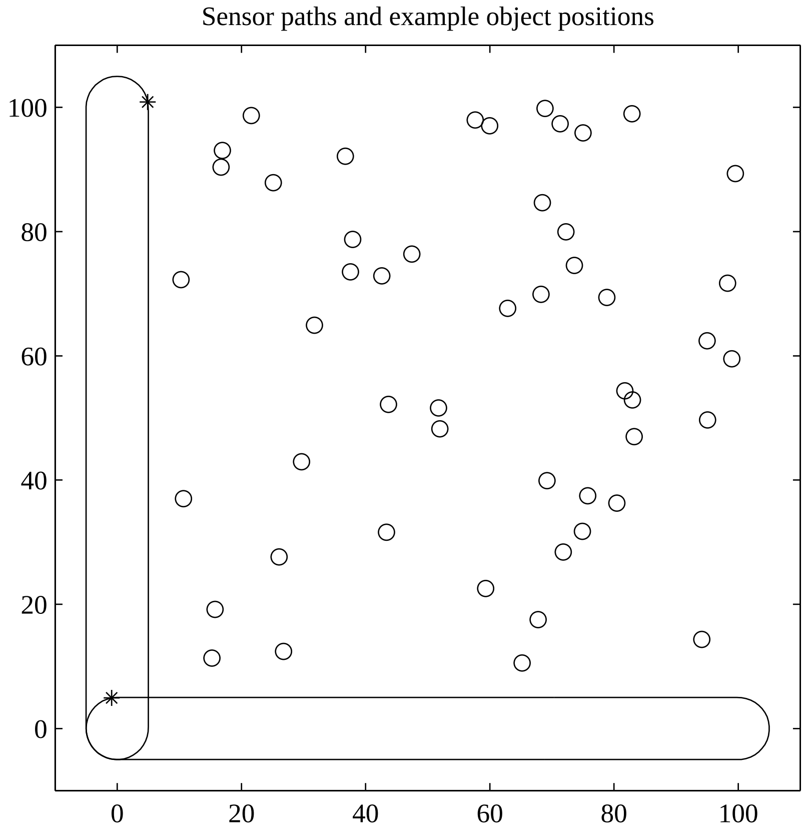


Figure 4.7. The two radar sensor platforms move along the racetrack patterns shown by the solid lines; the position of the two platforms in the tenth time slot is shown by the '*' marks. The sensor platforms complete 1.7 revolutions of the pattern in the 200 time slots in the simulation. M objects are positioned randomly within the $[10, 100] \times [10, 100]$ according to a uniform distribution, as illustrated by the '$\bigcirc$' marks.

and $\boldsymbol{z}_k^{i,j,d}$ denotes the azimuth/range rate (*i.e.*, Doppler) observation. The notation $[\boldsymbol{a}]_l$ denotes the l-th element of the vector $\boldsymbol{a}$; the first and third elements of the object state $\boldsymbol{x}_k^i$ and the sensor state $\boldsymbol{y}_k^j$ contain the position in the x-axis and y-axis respectively, while the second and fourth elements contain the velocity in the x-axis and y-axis respectively.

The observation noise in each observation is independent of all other observations, and distributed according to a Gaussian PDF:

$$\begin{aligned} p(\boldsymbol{v}_k^{i,j,r}) &= \mathcal{N}\left\{\boldsymbol{v}_k^{i,j,r}; \mathbf{0}, \begin{bmatrix} \sigma_\phi^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix}\right\} \\ p(\boldsymbol{v}_k^{i,j,d}) &= \mathcal{N}\left\{\boldsymbol{v}_k^{i,j,d}; \mathbf{0}, \begin{bmatrix} \sigma_\phi^2 & 0 \\ 0 & \sigma_d^2 \end{bmatrix}\right\} \end{aligned}$$

The standard deviation of the noise on the azimuth observations (σ_ϕ) is 3°; the multiplier function $b(\boldsymbol{x}_k^i, \boldsymbol{y}_k^j)$ varies from unity on the broadside (*i.e.*, when the sensor platform heading is perpendicular to the vector from the sensor to the object) to $3\frac{1}{3}$ end-on. The standard deviation of the range observation (σ_r) is 0.05 units, while the standard deviation of the range rate observation (σ_d) is 0.03 units/sec. In practice the tracking accuracy is high enough that the variation in azimuth noise due to the uncertainty in object position is small, so we simply use the object position estimate to calculate an estimate of the noise variance. A linearized Kalman filter is utilized in order to evaluate rewards for the planning algorithm, while an extended Kalman filter is used for estimation.

Results for 16 Monte Carlo simulations with planning horizons of between one and 30 time slots per sensor are shown in Fig. 4.8. The top diagram shows the results for 50 objects, while the middle diagram shows the results for 80 objects. The diagrams show that there is a small increase in performance as the planning horizon increases, although the increase is limited to around 1–3% over that obtained when the planning horizon is limited to a single time slot. With a planning horizon of one, planning is conducted jointly over the two sensors; the computation cost of doing so through brute force enumeration is equivalent to that of planning for a single sensor over two time slots.

The small increase in performance is not unexpected given that there is little non-stationarity in the problem. Intuitively one would expect a relatively short planning horizon to be adequate; Fig. 4.8 provides a confirmation of this intuition using planning horizons that are intractable using previous methods. The example also demonstrates the computational complexity of the method, as shown in the bottom diagram in

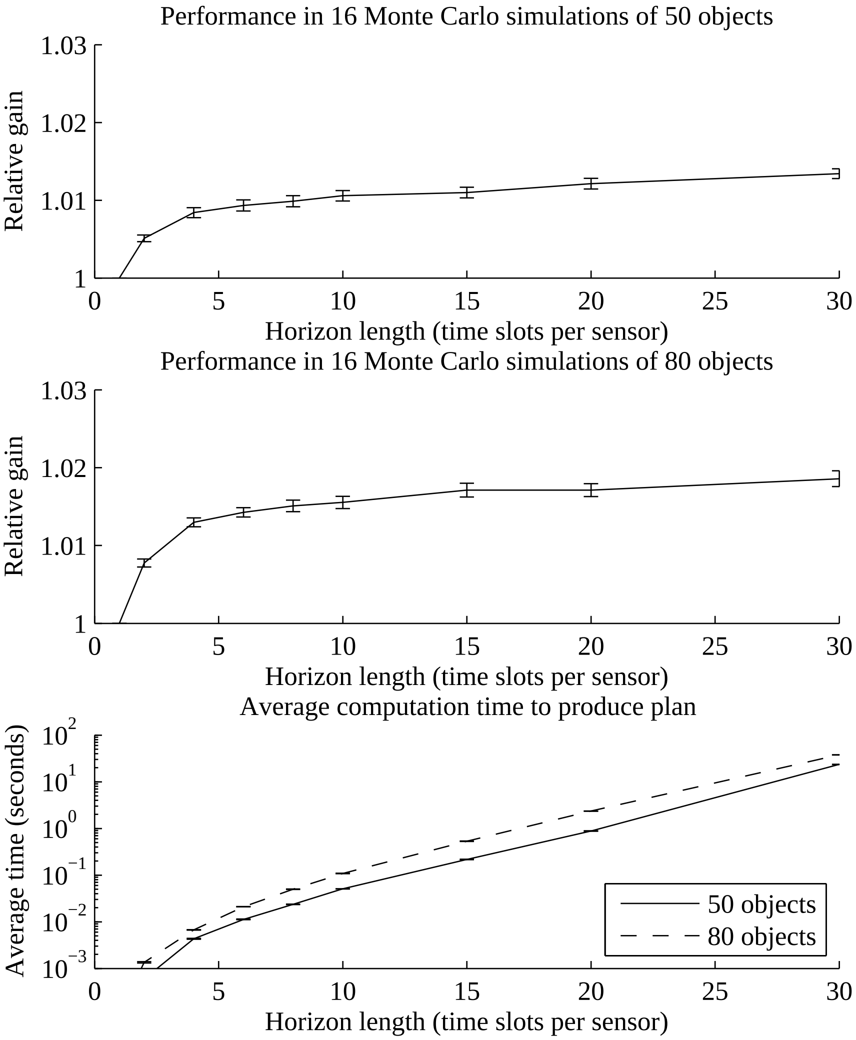


Figure 4.8. Results of Monte Carlo simulations for planning horizons between one and 30 time slots (in each sensor). Top diagram shows results for 50 objects, while middle diagram shows results for 80 objects. Each trace in the plots shows the total reward (*i.e.*, the sum of the MI reductions in each time step) of a single Monte Carlo simulation for different planning horizon lengths divided by the total reward with the planning horizon set to a single time step, giving an indication of the improvement due to additional planning. Bottom diagram shows the computation complexity (measured through the average number of seconds to produce a plan for the planning horizon) versus the planning horizon length.

Fig. 4.8. The diagram shows that, as expected, the complexity increases exponentially with the planning horizon length. However, using the algorithm it is possible to produce a plan for 50 objects over 20 time slots each in two sensors using little over one second in computation time. Performing the same planning through full enumeration would involve evaluation of the reward of 10^{80} different candidate sequences, a computation which is intractable on any foreseeable computational hardware.

The computational complexity for problems involving different numbers of objects for a fixed planning horizon length (10 time slots per sensor) is shown in Fig. 4.9. When the planning horizon is significantly longer than the number of objects, it becomes necessary to construct plans involving several observations of each object. This will generally involve enumeration of an exponentially increasing number of candidate subsets in $\mathscr{S}^i$ for each object, resulting in an increase in computational complexity. As the number of objects increases, it quickly becomes clear that it is better to spread the available resources evenly across objects rather than taking many observations of a small number of objects, so the number of candidate subsets in $\mathscr{S}^i$ requiring consideration is vastly lower. Eventually, as the number of objects increases, the overhead induced by the additional objects again increases the computational complexity.

■ 4.4.3 State dependent observation noise

The second scenario involves a modification of the first in which observation noise increases when objects become close to each other. This is a surrogate for the impact of data association, although we do not model the dependency between objects which generally results. The dynamical model has $\Delta t = 0.01$ sec, and $q = 0.25$; the simulation runs for 100 time slots. As per the previous scenario, the initial positions of the objects are distributed uniformly on the region $[10, 100] \times [10, 100]$; velocity magnitudes are drawn from a Gaussian distribution with mean 30 and standard deviation 0.5, while the velocity directions are distributed uniformly on $[0, 2\pi]$. The initial estimates are set to the true state, corrupted by additive Gaussian noise with zero mean and standard deviation 0.02 (in position states) and 0.1 (in velocity states). The scenario involves a single sensor rather than two sensors; the observation model is essentially the same as the previous case, except that there is a state-dependent scalar multiplier on the

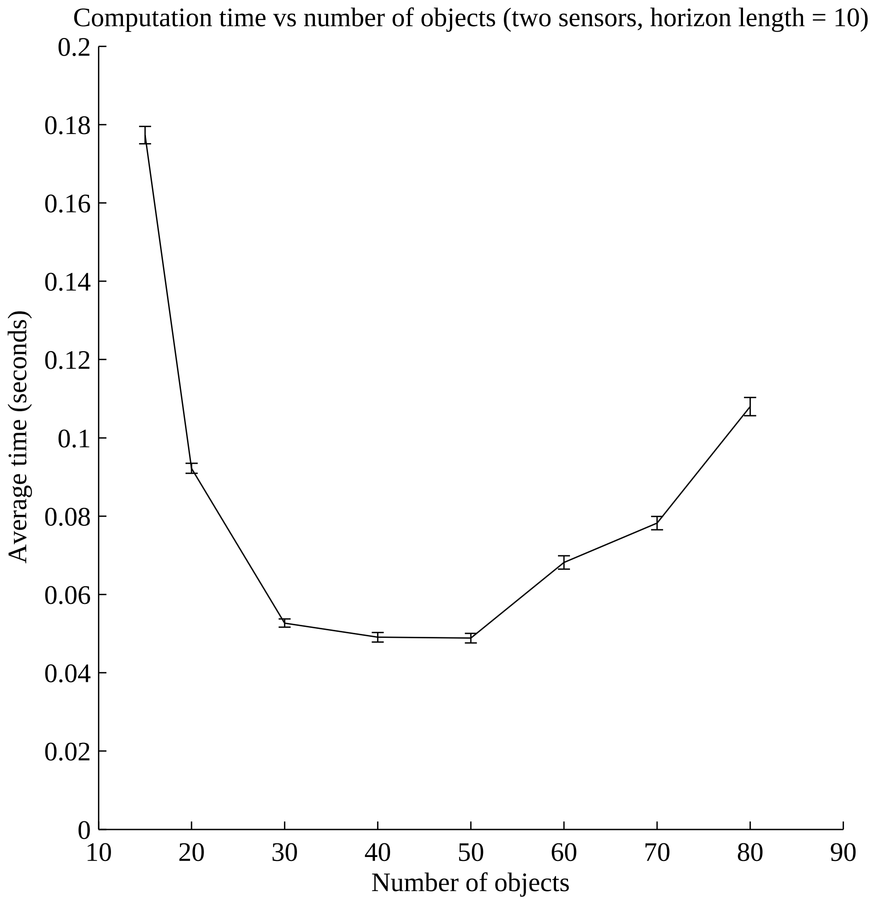


Figure 4.9. Computational complexity (measured as the average number of seconds to produce a plan for the 10-step planning horizon) for different numbers of objects.

observation noise:

$$\boldsymbol{z}_k^{i,j,r} = \begin{bmatrix} \tan^{-1}\left(\frac{[\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_3}{[\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_1}\right) \\ \sqrt{([\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_1)^2 + ([\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_3)^2} \end{bmatrix} + d^i(\boldsymbol{x}_k^1, \dots, \boldsymbol{x}_k^M) \begin{bmatrix} b(\boldsymbol{x}_k^i, \boldsymbol{y}_k^j) & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{v}_k^{i,j,r} \tag{4.20}$$

$$\boldsymbol{z}_k^{i,j,d} = \begin{bmatrix} \tan^{-1}\left(\frac{[\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_3}{[\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_1}\right) \\ \frac{[\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_1 [\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_2 + [\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_3 [\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_4}{\sqrt{([\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_1)^2 + ([\boldsymbol{x}_k^i - \boldsymbol{y}_k^j]_3)^2}} \end{bmatrix} + d^i(\boldsymbol{x}_k^1, \dots, \boldsymbol{x}_k^M) \begin{bmatrix} b(\boldsymbol{x}_k^i, \boldsymbol{y}_k^j) & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{v}_k^{i,j,d} \tag{4.21}$$

The azimuth noise multiplier $b(\boldsymbol{x}_k^i, \boldsymbol{y}_k^j)$ is the same as in Section 4.4.2, as is the azimuth noise standard deviation ($\sigma_\phi = 3°$). The standard deviation of the noise on the range observation is $\sigma_r = 0.1$, and on the range rate observation is $\sigma_d = 0.075$. The function $d(\boldsymbol{x}_k^1, \dots, \boldsymbol{x}_k^M)$ captures the increase in observation noise when objects are close together:

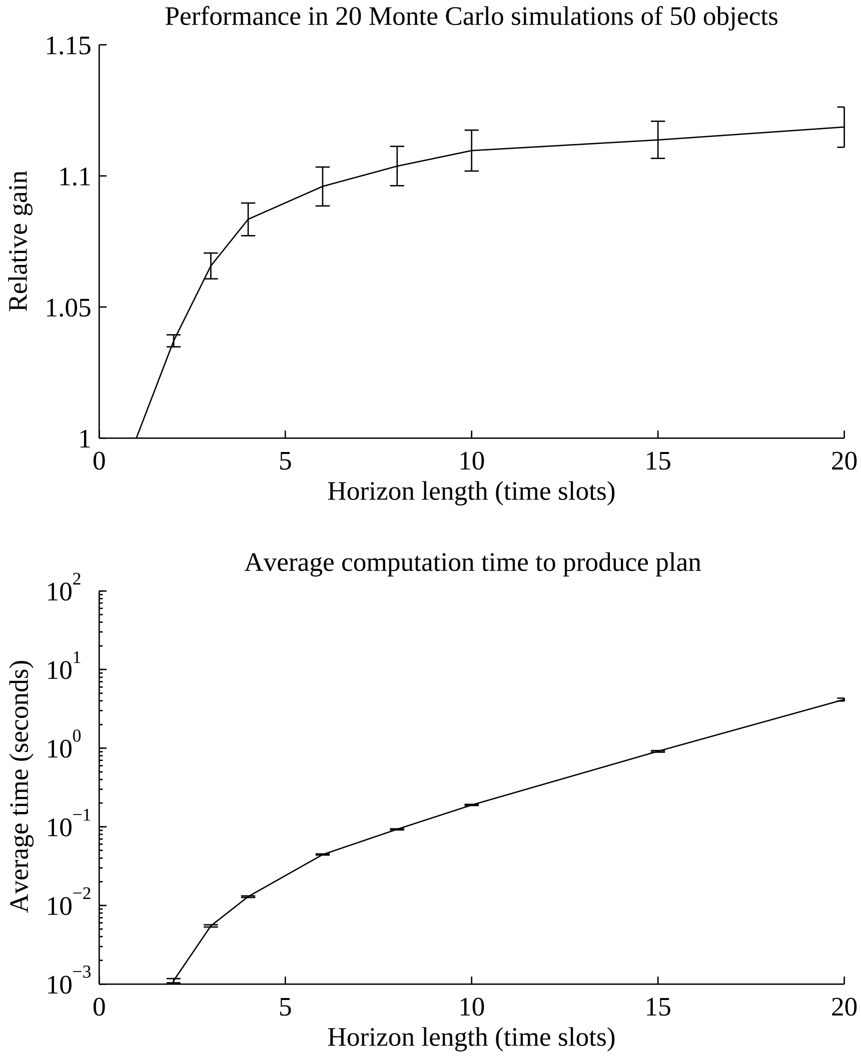
$$d^i(\boldsymbol{x}_k^1, \dots, \boldsymbol{x}_k^M) = \sum_{j \neq i} \delta\left(\sqrt{([\boldsymbol{x}_k^i - \boldsymbol{x}_k^j]_1)^2 + ([\boldsymbol{x}_k^i - \boldsymbol{x}_k^j]_3)^2}\right)$$

where $\delta(x)$ is the piecewise linear function:

$$\delta(x) = \begin{cases} 10 - x, & 0 \leq x < 10 \\ 0, & x \geq 10 \end{cases}$$

The state dependent noise is handled in a manner similar to the optimal linear estimator for bilinear systems, in which we estimate the variance of the observation noise, and then use this in a conventional linearized Kalman filter (for reward evaluations for planning) and extended Kalman filter (for estimation). We draw a number of samples of the joint state of all objects, and evaluate the function $d(\boldsymbol{x}_k^1, \dots, \boldsymbol{x}_k^M)$ for each. We then estimate the noise multiplier as being the 90% percentile point of these evaluations, *i.e.*, the smallest value such that 90% of the samples evaluate to a lower value. This procedure provides a pessimistic estimate of the noise amplitude.

In addition to the option of these two observations, the sensor can also choose a more accurate observation that takes three time slots to complete, and is not subject increased noise when objects become closely spaced. The azimuth noise for these observations in the broadside aspect has $\sigma_\phi = 0.6°$, while the range noise has $\sigma_r = 0.02$ units, and the range rate noise has $\sigma_d = 0.015$ units/sec.



Performance in 20 Monte Carlo simulations of 50 objects

Average computation time to produce plan

Figure 4.10. Top diagram shows the total reward for each planning horizon length divided by the total reward for a single step planning horizon, averaged over 20 Monte Carlo simulations. Error bars show the standard deviation of the mean performance estimate. Lower diagram shows the average time required to produce plan for the different length planning horizon lengths.

The results of the simulation are shown in Fig. 4.10. When the planning horizon is less than three time steps, the controller does not have the option of the three time step observation available to it. A moderate gain in performance is obtained by extending the planning horizon from one time step to three time steps to enable use of the longer observation. The increase is roughly doubled as the planning horizon is increased, allowing the controller to anticipate periods when objects are unobservable. The computational complexity is comparable with that of the previous case; the addition of observations consuming multiple time slots does not increase computation time significantly.

■ 4.4.4 Example of potential benefit: single time slot observations

The third scenario demonstrates the gain in performance which is possible by planning over long horizon length on problems to which the guarantees of Chapter 3 apply (*i.e.*, when all observations occupy a single time slot). The scenario involves $M = 50$ objects being tracked using a single sensor over 50 time slots. The object states evolve according to the nominally constant velocity model described in Eq. (2.8), with $\Delta t = 10^{-4}$ sec and $q = 1$. The initial position and velocity of the objects is identical to the scenario in Section 4.4.3. The initial state estimates of the first 25 objects are corrupted by Gaussian noise with covariance $\mathbf{I}$ (*i.e.*, the 4×4 identity matrix), while the estimates of the remaining objects is corrupted by Gaussian noise with covariance $1.1\mathbf{I}$.

In each time slot, any one of the M objects can be observed; each observation has a linear Gaussian model (*i.e.*, Eq. (2.11)) with $\mathbf{H}^i_k = \mathbf{I}$. The observation noise for the first 25 objects has covariance $\mathbf{R}^i_k = 10^{-6}\mathbf{I}$ in the first 25 time slots, and $\mathbf{R}^i_k = \mathbf{I}$ in the remaining time slots. The observation noise of the remaining objects has covariance $\mathbf{R}^i_k = 10^{-6}\mathbf{I}$ for all k. While this example represents an extreme case, one can see that similar events can commonly occur on a smaller scale in realistic scenarios; *e.g.*, in the problem examined in the previous section, observation noise variances frequently increased as objects became close together.

The controller constructs a plan for each N-step planning horizon, and then executes a single step before re-planning for the following N steps. When the end of the current N-step planning horizon is the end of the scenario (*e.g.*, in the eleventh time step when $N = 40$, and in the first time step when $N = 50$), the entire plan is executed without re-planning.

Intuitively, it is obvious that planning should be helpful in this scenario: half of the objects have significantly degraded observability in the second half of the simulation, and failure to anticipate this will result in a significant performance loss. This intuition is confirmed in the results presented in Fig. 4.11. The top plot shows the increase in relative performance as the planning horizon increases from one time slot to 50 (*i.e.*, the total reward for each planning horizon, divided by the total reward when the planning horizon is unity). Each additional time slot in the planning horizon allows the controller to anticipate the change in observation models sooner, and observe more of the first 25 objects before the increase in observation noise covariance occurs. The performance increases monotonically with planning horizon apart from minor variations. These are due to the stopping criteria in our algorithm: rather than waiting for an optimal solution, we terminate when we obtain a solution that is within 95% of the optimal reward.

With a planning horizon of 50 steps (spanning the entire simulation length), the total reward is 74% greater than the total reward with a single step planning horizon. Since all observations occupy a single time slot, the performance guarantees of Chapter 3, and the maximum gain possible in any scenario of this type is 100% (*i.e.*, the optimal performance can be no more than twice that of the one-step greedy heuristic). Once again, while this example represents an extreme case, one can see that similar events can commonly occur on a smaller scale in realistic scenarios. The smaller change in observation model characteristics and comparative infrequency of these events results the comparatively modest gains found in Sections 4.4.2 and 4.4.3.

■ 4.4.5 Example of potential benefit: multiple time slot observations

The final scenario demonstrates the increase in performance which is possible through long planning horizons when observations occupy different numbers of time slots. In such circumstances, algorithms utilizing short-term planning may make choices that preclude selection of later observations that may be arbitrarily more valuable.

The scenario involves $M = 50$ objects observed using a single sensor. The initial positions and velocities of the objects are the same as in the previous scenario; the initial estimates are corrupted by additive Gaussian noise with zero mean and covariance $\mathbf{I}$.

In each time slot, a single object may be observed through either of two linear Gaussian observations (*i.e.*, of the form in Eq. (2.11)). The first, which occupies a single time slot, has $\mathbf{H}_k^{i,1} = \mathbf{I}$, and $\mathbf{R}_k^{i,1} = 2\mathbf{I}$. The second, which occupies five time slots, has $\mathbf{H}_k^{i,2} = \mathbf{I}$, and $\mathbf{R}_k^{i,2} = r_k\mathbf{I}$. The noise variance of the longer observation, r_k,

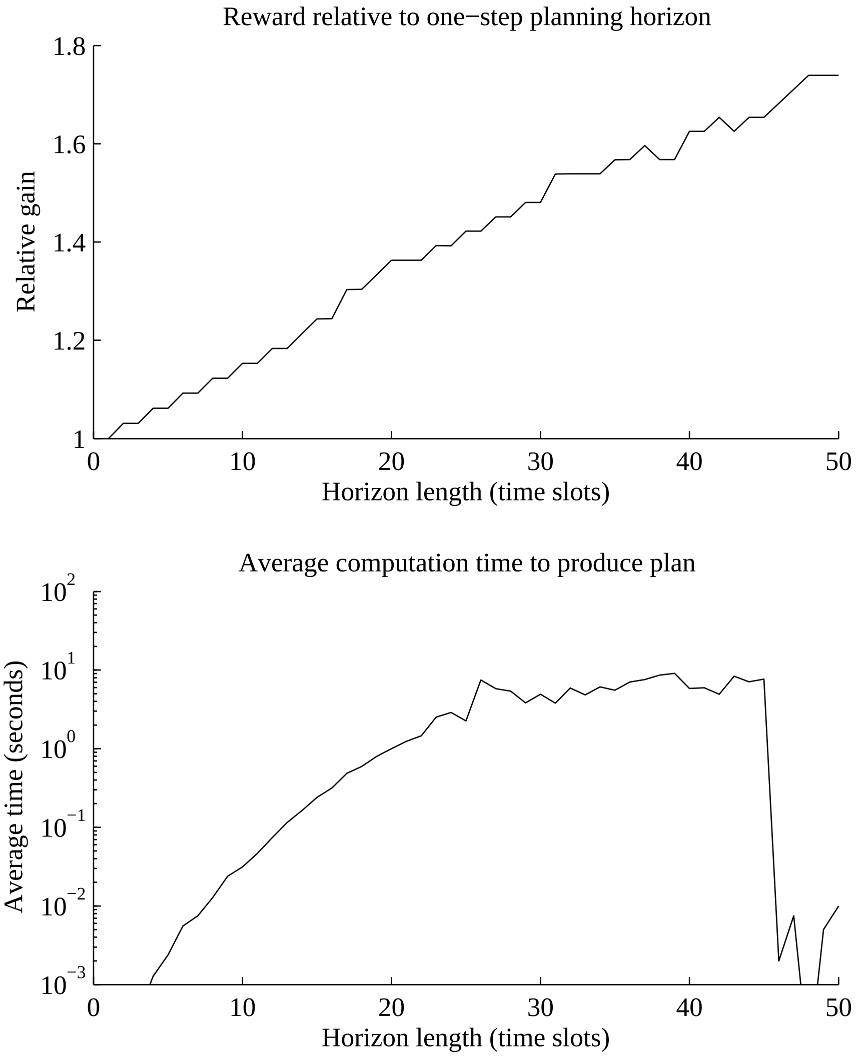


Figure 4.11. Upper diagram shows the total reward obtained in the simulation using different planning horizon lengths, divided by the total reward when the planning horizon is one. Lower diagram shows the average computation time to produce a plan for the following N steps.

varies periodically with time, according to the following values:

$$r_k = \begin{cases} 10^{-1} & \text{mod }(k,5) = 1 \\ 10^{-2} & \text{mod }(k,5) = 2 \\ 10^{-3} & \text{mod }(k,5) = 3 \\ 10^{-4} & \text{mod }(k,5) = 4 \\ 10^{-5} & \text{mod }(k,5) = 0 \end{cases}$$

The time index k commences at $k = 1$. Unless the planning horizon is sufficiently long to anticipate the availability of the observation with variance 10^{-5} several time steps later, the algorithm will select an observation with lower reward, which precludes selection of this later more accurate observation.

The performance of the algorithm in the scenario is shown in Fig. 4.12. The maximum increase in performance over the greedy heuristic is a factor of 4.7×. While this is an extreme example, it illustrates another occasion when additional planning is highly beneficial: when there are observations that occupy several time slots with time varying rewards. In this circumstance, an algorithm utilizing short-term planning may make choices that preclude selection of later observations which may be arbitrarily more valuable.

■ 4.5 Time invariant rewards

In many selection problems where the time duration corresponding to the planning horizon is short, the reward associated with observing the same object using the same sensing mode at different times within the planning horizon is well-approximated as being time invariant. In this case, the complexity of the selection problem can be reduced dramatically by replacing the individual resources associated with each time slot with a single resource, with capacity corresponding to the total number of time units available. In this generalization of Sections 4.2 and 4.3, we associate with each resource $r \in \mathcal{R}$ a capacity $C^r \in \mathbb{R}$, and define $t(u^i_j, r) \in \mathbb{R}$ to be the capacity of resource r consumed by elemental observation $u^i_j \in \mathcal{U}^i$. The collection of subsets from which we may select for object i is changed from Eq. (4.12) to:

$$\mathscr{S}^i = \{\mathcal{A} \subseteq \mathcal{U}^i | t(\mathcal{A}, r) \leq C^r \ \forall \ r \in \mathcal{R}\} \tag{4.22}$$

where

$$t(\mathcal{A}, r) = \sum_{u \in \mathcal{A}} t(u, r)$$

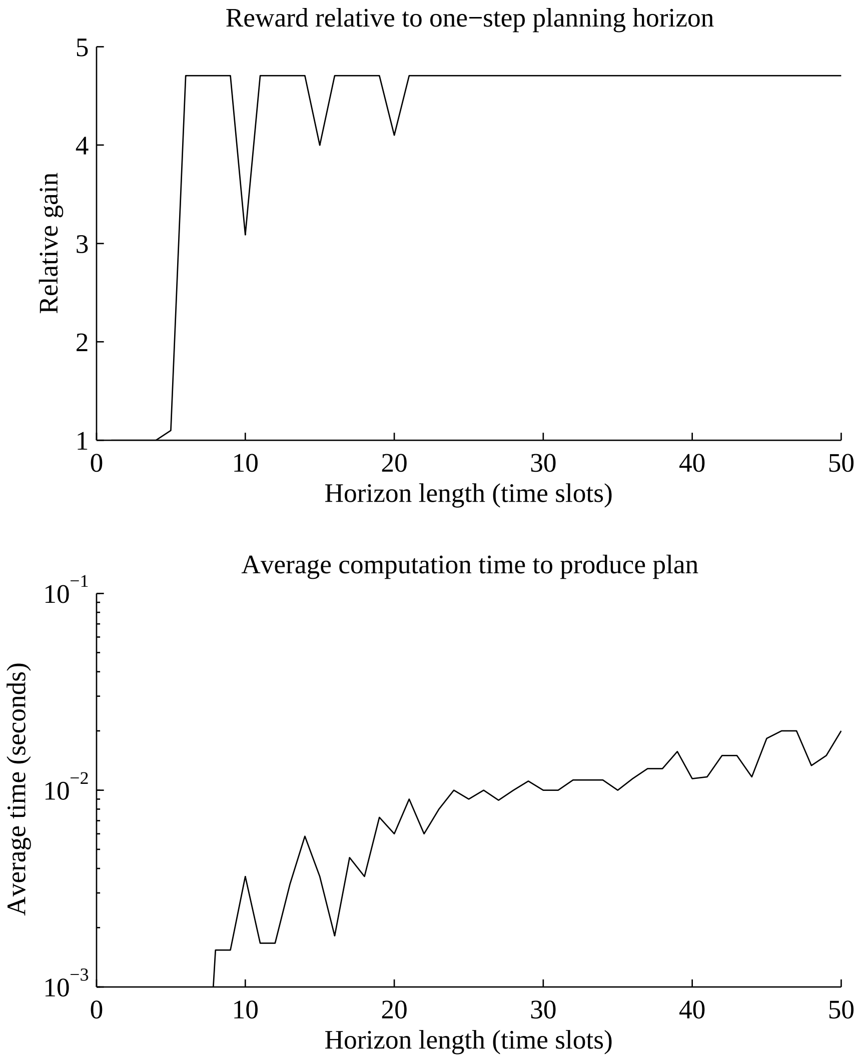


Figure 4.12. Upper diagram shows the total reward obtained in the simulation using different planning horizon lengths, divided by the total reward when the planning horizon is one. Lower diagram shows the average computation time to produce a plan for the following N steps.

The full integer programming formulation of Eq. (4.15) becomes:

$$\max_{\omega^i_{\mathcal{A}^i}} \sum_{i=1}^{M} \sum_{\mathcal{A}^i \in \mathscr{S}^i} r^i_{\mathcal{A}^i} \omega^i_{\mathcal{A}^i} \tag{4.23a}$$

$$\text{s.t.} \sum_{i=1}^{M} \sum_{\mathcal{A}^i \in \mathscr{S}^i} t(\mathcal{A}^i, r) \omega^i_{\mathcal{A}^i} \le C^r \quad \forall\, r \in \mathcal{R} \tag{4.23b}$$

$$\sum_{\mathcal{A}^i \in \mathscr{S}^i} \omega^i_{\mathcal{A}^i} = 1 \quad \forall\, i \in \{1, \dots, M\} \tag{4.23c}$$

$$\omega^i_{\mathcal{A}^i} \in \{0, 1\} \quad \forall\, i,\ \mathcal{A}^i \in \mathscr{S}^i \tag{4.23d}$$

The iterative solution methodology in Section 4.3 may be generalized to this case by replacing Eq. (4.16) with:

$$\max_{\omega^i_{\mathcal{A}^i}, \omega^i_{u|\mathcal{A}^i}} \sum_{i=1}^{M} \sum_{\mathcal{A}^i \in \mathscr{T}^i_l} \left[r^i_{\mathcal{A}^i} \omega^i_{\mathcal{A}^i} + \sum_{u \in \mathcal{B}^i_{l,\mathcal{A}^i}} r^i_{u|\mathcal{A}^i} \omega^i_{u|\mathcal{A}^i} \right] \tag{4.24a}$$

$$\text{s.t.} \sum_{i=1}^{M} \sum_{\mathcal{A}^i \in \mathscr{T}^i_l} t(\mathcal{A}^i, r) \omega^i_{\mathcal{A}^i} + \sum_{i=1}^{M} \sum_{\mathcal{A}^i \in \mathscr{T}^i_l} \sum_{u \in \mathcal{B}^i_{l,\mathcal{A}^i}} t(u, r) \omega^i_{u|\mathcal{A}^i} \le C^r \quad \forall\, r \in \mathcal{R} \tag{4.24b}$$

$$\sum_{\mathcal{A}^i \in \mathscr{T}^i} \omega^i_{\mathcal{A}^i} = 1 \quad \forall\, i \in \{1, \dots, M\} \tag{4.24c}$$

$$\sum_{u \in \mathcal{B}^i_{l,\mathcal{A}^i}} \omega^i_{u|\mathcal{A}^i} - |\mathcal{B}^i_{l,\mathcal{A}^i}| \omega^i_{\mathcal{A}^i} \le 0 \quad \forall\, i,\ \mathcal{A}^i \in \mathscr{T}^i_l \tag{4.24d}$$

$$\omega^i_{\mathcal{A}^i} \in \{0, 1\} \quad \forall\, i,\ \mathcal{A}^i \in \mathscr{T}^i \tag{4.24e}$$

$$\omega^i_{u|\mathcal{A}^i} \in \{0, 1\} \quad \forall\, i,\ \mathcal{A}^i \in \mathscr{T}^i,\ u \in \mathcal{B}^i_{l,\mathcal{A}^i} \tag{4.24f}$$

The only change from Eq. (4.16) is in the form of the resource constraint, Eq. (4.24b). Algorithm 4.1 may be applied without modification using this generalized integer program.

■ 4.5.1 Avoiding redundant observation subsets

Straight-forward implementation of the formulation just described will yield a substantial inefficiency due to redundancy in the observation subset $\mathscr{S}^i$, as illustrated in the following scenario. Suppose we seek to generate a plan for the next N time slots, where there are η total combinations of sensor and mode within each time slot. For each sensor mode[4] u^i_j, $j \in \{1, \dots, \eta\}$, we generate for each duration $d \in \{1, \dots, N\}$

[4] We assume that the index j enumerates all possible combinations of sensor and mode.

(*i.e.*, the duration for which the sensor mode is applied) an elemental observation $u^i_{j,d}$, which corresponds to applying sensor mode u^i_j for d time slots, so that $\mathcal{U}^i = \{u^i_{j,d} | j \in \{1, \dots, \eta\}, d \in \{1, \dots, N\}\}$. With each sensor $s \in \mathcal{S}$ (where $\mathcal{S}$ is the set of sensors) we associate a resource r^s with capacity $C^{r^s} = N$, *i.e.*, each sensor has up to N time slots available for use. Denoting by $s(u^i_j)$ the sensor associated with u^i_j, we set

$$t(u^i_{j,d}, s) = \begin{cases} d, & s = s(u^i_j) \\ 0, & \text{otherwise} \end{cases}$$

We could generate the collection of observation subsets $\mathscr{S}^i$ from Eq. (4.22) using this structure, with $\mathcal{U}^i$ and $t(\cdot, \cdot)$ as described above, and $\mathcal{R} = \{r^s | s \in \mathcal{S}\}$. However, this would introduce a substantial inefficiency which is quite avoidable. Namely, in addition to containing the single element observation subset $\{u^i_{j,d}\}$, the collection of subsets $\mathscr{S}^i$ may also contain several subsets of the form $\{u^i_{j,d_l} | \sum_l d_l = d\}$.[5] We would prefer to avoid this redundancy, ensuring instead that the only way observe object i using sensor mode j for a total duration of d time slots is to choose the elemental observation $u^i_{j,d}$ alone. This can be achieved by introducing one additional resource for each combination of object and sensor mode, $\{r^{i,j} | i \in \{1, \dots, M\}, j \in \{1, \dots, \eta\}\}$, each with capacity $C^{r^{i,j}} = 1$, setting:

$$t(u^i_{j,d}, r^{i',j'}) = \begin{cases} 1, & i = i', \ j = j' \\ 0, & \text{otherwise} \end{cases}$$

The additional constraints generated by these resources ensure that (at most) a single element from the set $\{u^i_{j,d} | d \in \{1, \dots, N\}\}$ can be chosen for each (i, j). As the following experiment demonstrates, this results in a dramatic decrease in the action space for each object, enabling solution of larger problems.

■ 4.5.2 Computational experiment: waveform selection

To demonstrate the reduction in computational complexity that results from this formulation, we apply it to a modification of the example presented in Section 4.4.2. We set $\Delta t = 0.01$ and reduce the sensor platform motion to 0.01 units/step. Fig. 4.13 shows variation in rewards for 50 objects being observed with a single mode of a single sensor over 50 time slots in one realization of the example, confirming that the rewards are reasonably well approximated as being time-invariant.

[5] For example, as well as providing an observation subset for observing object i with sensor mode j

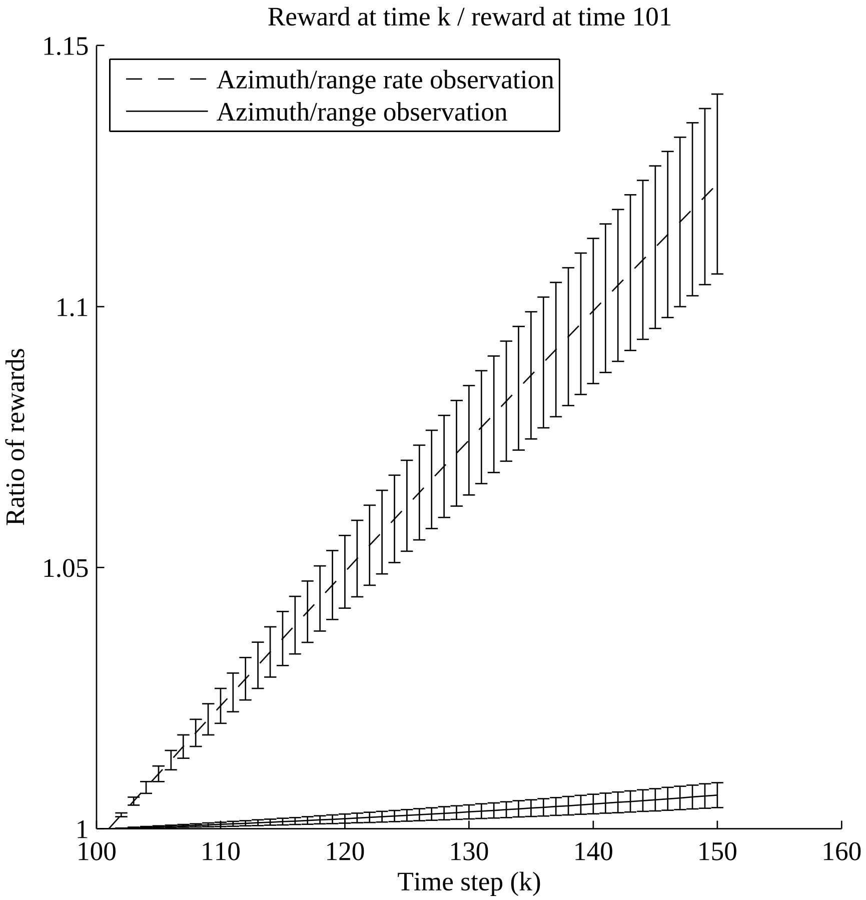


Figure 4.13. Diagram illustrates the variation of rewards over the 50 time step planning horizon commencing from time step $k = 101$. The line plots the ratio between the reward of each observation at time step in the planning horizon and the reward of the same observation at the first time slot in the planning horizon, averaged over 50 objects. The error bars show the standard deviation of the ratio, *i.e.*, the variation between objects.

The simulations run for 200 time slots. The initial position and velocity of the objects is identical to that in Section 4.4.2; the initial state estimates are corrupted by additive Gaussian noise with covariance $(1 + \rho^i)\mathbf{I}$, where ρ^i is randomly drawn, independently for each object, from a uniform distribution on $[0, 1]$ (the values are known to the controller). The observation model is identical to that described in Section 4.4.2. If the controller chooses to observe an object for d time slots, the variance is reduced by a factor of d from that of the same single time slot observation; in the absence of the dynamics process this is equivalent to d single time slot observations.

At each time step, the controller constructs an N-step plan. Since rewards are assumed time invariant, the elements of this plan are not directly attached to time slots. We assign time slots to each task in the plan by processing objects in random order, assigning the first time slots to the observations assigned to the first object in the random order, *etc.* We then execute the action assigned to the first time slot in the plan, before constructing a new plan; this is consistent with the open loop feedback control methodology used in the examples in Section 4.4.

The results of the simulations are shown in Fig. 4.14. There is a very small gain in performance as the planning horizon increases from one time slot to around five time slots. Beyond this limit, the performance drops to be lower than the performance of the greedy heuristic (*i.e.*, using a single step planning horizon). This is due to the mismatch between the assumed model (that rewards are time invariant) and the true model (that rewards are indeed time varying), which worsens as the planning horizon increases. The computational cost in the lower diagram demonstrates the efficiency of this formulation. With a planning horizon of 40, we are taking an average of four observations per object. This would attract a very large computational burden in the original formulation discussed in the experiments of Section 4.4.

■ 4.5.3 Example of potential benefit

To demonstrate the benefit that additional planning can provide in problems involving time invariant rewards, we construct an experiment that is an extension of the example presented in Section 3.4. The scenario involves $M = 50$ objects being observed using a single sensor over 100 time slots. The four-dimensional state of each object is static in

for $d = 10$ steps, there are also subsets for observing object i with sensor mode j multiple times for the same total duration—*e.g.*, for 4 and 6 steps, for 3 and 7 steps, and for 1, 4 and 5 steps.

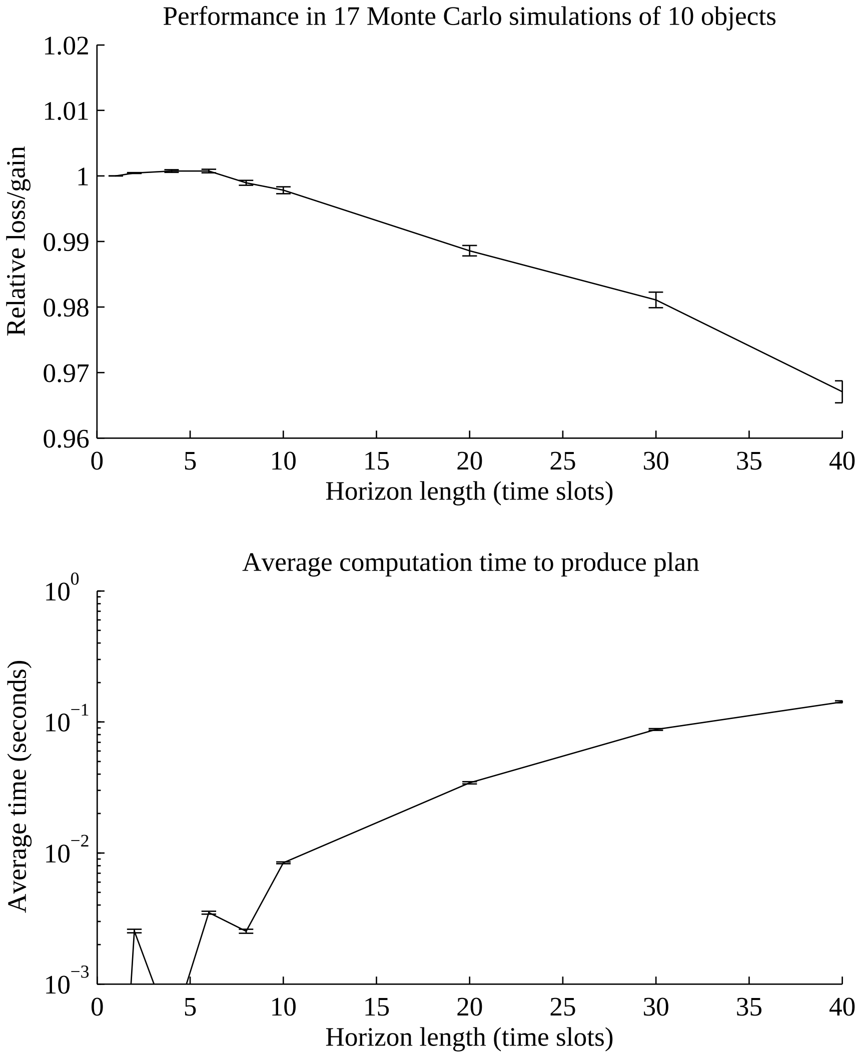


Figure 4.14. Top diagram shows the total reward for each planning horizon length divided by the total reward for a single step planning horizon, averaged over 17 Monte Carlo simulations. Error bars show the standard deviation of the mean performance estimate. Lower diagram shows the average time required to produce plan for the different length planning horizon lengths.

time, and the initial distribution is Gaussian with covariance:

$$\mathbf{P}_0^i = \begin{bmatrix} 1.1 & 0 & 0 & 0 \\ 0 & 1.1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In each time slot, there are three different linear Gaussian observations available for each object. The measurement noise covariance of each is $\mathbf{R}_k^{i,1} = \mathbf{R}_k^{i,2} = \mathbf{R}_k^{i,3} = 10^{-6}\mathbf{I}$, while the forward models are:

$$\mathbf{H}_k^{i,1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}; \quad \mathbf{H}_k^{i,2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}; \quad \mathbf{H}_k^{i,3} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The performance of the algorithm in the scenario is summarized in Fig. 4.15. The results demonstrate that performance increases by 29% as planning increases from a single step to the full simulation length (100 steps). Additional planning allows the controller to anticipate that it will be possible to take a second observation of each object, and hence, rather than utilizing the first observation (which has the highest reward) it should utilize either the second or third observations, which are completely complementary and together provide all of the information found in the first. For longer planning horizons, the computational complexity appears to be roughly linear with planning horizon; the algorithm is able to construct a plan for the entire 100 time slots in five seconds.

■ 4.6 Conclusion

The development in this chapter provides an efficient method of optimal and near-optimal solution for a wide range of beam steering problems involving multiple independent objects. Each iteration of the algorithm in Section 4.3.3 provides an successive reduction of the upper bound to the reward attainable. This may be combined with the augmented problem presented in Section 4.3.6, which provides a series of solutions for which the rewards are successively improved, to yield an algorithm that terminates when a solution has been found that is within the desired tolerance of optimality. The experiments in Section 4.4 demonstrate the computational efficiency of the approach, and the gain in performance that can be obtained through use of longer planning horizons.

In practical scenarios it is commonly the case that, when objects become closely spaced, the only measurements available are joint observations of the two (or more)

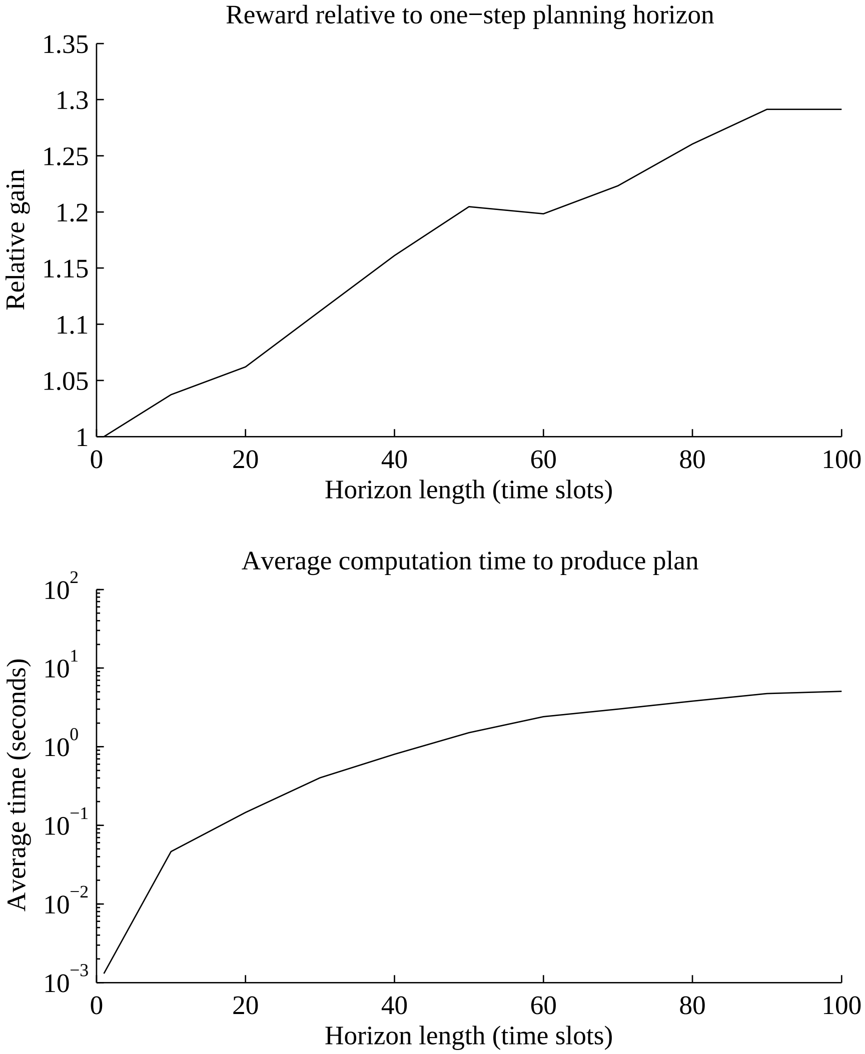


Figure 4.15. Upper diagram shows the total reward obtained in the simulation using different planning horizon lengths, divided by the total reward when the planning horizon is one. Lower diagram shows the average computation time to produce a plan for the following N steps.

objects, rather than observations of the individual objects. This inevitably results in statistical dependency between object states in the conditional distribution; the dependency is commonly represented through association hypotheses (*e.g.*, [14, 19, 58, 71, 82]).

If the objects can be decomposed into many small independent "groups", as in the notion of independent clusters in Multiple Hypothesis Tracking (MHT), then Algorithm 4.1 may be applied to the transformed problem in which each independent group of objects is treated as a single object with state and action space corresponding to the cartesian product of the objects forming the group. This approach may be tractable if the number of objects in each group remains small.

Chapter 5

Sensor management in sensor networks

NETWORKS of intelligent sensors have the potential to provide unique capabilities for monitoring wide geographic areas through the intelligent exploitation of local computation (so called in-network computing) and the judicious use of inter-sensor communication. In many sensor networks energy is a dear resource to be conserved so as to prolong the network's operational lifetime. Additionally, it is typically the case that the energy cost of communications is orders of magnitude greater than the energy cost of local computation [80, 81].

Tracking moving objects is a common application in which the quantities of interest (*i.e.*, kinematic state) are inferred largely from sensor observations which are in proximity to the object (*e.g.*, [62]). Consequently, local fusion of sensor data is sufficient for computing an accurate estimate of object state, and the knowledge used to compute this estimate is summarized by the conditional probability density function (PDF). This property, combined with the need to conserve energy, has led to a variety of approaches (*e.g.*, [37, 64]) which effectively designate the responsibility of computing the conditional PDF to one sensor node (referred to as the leader node) in the network. Over time the leader node changes dynamically as function of the kinematic state of the object. This leads to an inevitable trade-off between the uncertainty in the conditional PDF, the cost of acquiring observations, and the cost of propagating the conditional PDF through the network. In this chapter we examine this trade-off in the context of object tracking in distributed sensor networks.

We consider a sensor network consisting a set of sensors (denoted $\mathcal{S}$, where $|\mathcal{S}| = N_s$), in which the sensing model is assumed to be such that the observation provided by the sensor is highly informative in the region close to the node, and uninformative in regions far from the node. For the purpose of addressing the primary issue, trading off energy consumption for accuracy, we restrict ourselves to sensor resource planning issues associated with tracking a single object. While additional complexities certainly arise in the multi-object case (*e.g.*, data association) they do not change the basic problem formulation or conclusions.

If the energy consumed by sensing and communication were unconstrained, then the optimal solution would be to collect and fuse the observations provided by *all* sensors in the network. We consider a scheme in which, at each time step, a subset of sensors is selected to take an observation and transmit to a sensor referred to as the leader node, which fuses the observations with the prior conditional PDF and tasks sensors at the next time step. The questions which must be answered by the controller are how to select the subset of sensors at each point in time, and how to select the leader node at each point in time.

The approach developed in Section 5.1 allows for optimization of estimation performance subject to a constraint on expected communication cost, or minimization of communication cost subject to a constraint on expected estimation performance. The controller uses a dual problem formulation to adaptively utilize multiple sensors at each time step, incorporating a subgradient update step to adapt the dual variable (Section 5.1.9), and introducing a heuristic cost to go in the terminal cost to avoid anomalous behavior (Section 5.1.10). Our dual problem formulation is closely related to [18], and provides an approximation which extends the Lagrangian relaxation approach to problems involving sequential replanning. Other related work includes [29], which suggests incorporation of sensing costs and estimation performance into a unified objective without adopting the constrained optimization framework that we utilize, and [20], which adopts a constrained optimization framework without incorporating estimation performance and sensing cost into a unified objective, a structure which results in a major computational saving for our approach.

■ 5.1 Constrained Dynamic Programming Formulation

The sensor network object tracking problem involves an inherent trade-off between performance and energy expenditure. One way of incorporating both estimation performance and communication cost into an optimization procedure is to optimize one of the quantities subject to a constraint on the other. In the development which follows, we provide a framework which can be used to either maximize the information obtained from the selected observations subject to a constraint on the expected communication cost, or to minimize the communication cost subject to a constraint on the estimation quality. This can be formulated as a constrained Markov Decision Process (MDP), as discussed in Section 2.2.3.

As discussed in the introduction to this chapter, the tracking problem naturally fits into the Bayesian state estimation formulation, such that the role of the sensor network is to maintain a representation of the conditional PDF of the object state (*i.e.*, position, velocity, *etc*) conditioned on the observations. In our experiments, we utilize a particle filter to maintain this representation (as described in Section 2.1.4), although the planning method that we develop is equally applicable to any state estimation method including the Kalman filter (Section 2.1.2), extended Kalman filter (Section 2.1.3), or unscented Kalman filter [39]. An efficient method of compressing particle representations of PDFs for transmission in sensor networks is studied in [36]; we envisage that any practical implementation of particle filters in sensor networks would use such a scheme.

The estimation objective that we employ in our formulation is discussed in Section 5.1.1, while our communication cost is discussed in Section 5.1.2. These two quantities are utilized differently in dual formulations, the first of which optimizes estimation performance subject to a constraint on communication cost (Section 5.1.3), and the second of which optimizes communication cost subject to a constraint on estimation performance (Section 5.1.4). In either case, the control choice available at each time is $u_k = (l_k, \mathcal{S}_k)$, where $l_k \in \mathcal{S}$ is the leader node at time k and $\mathcal{S}_k \subseteq \mathcal{S}$ is the subset of sensors activated at time k. The decision state of the dynamic program is the combination of conditional PDF of object state, denoted $\mathbb{X}_k \triangleq p(\boldsymbol{x}_k|\check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1})$, and the previous choice of leader node, $l_{k-1} \in \mathcal{S}$.

■ 5.1.1 Estimation objective

The estimation objective that we utilize in our formulation is the joint entropy of the object state over the N steps commencing from the current time k:

$$H(\boldsymbol{x}_k, \dots, \boldsymbol{x}_{k+N-1}|\check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1}, \boldsymbol{z}_k^{\mathcal{S}_k}, \dots, \boldsymbol{z}_{k+N-1}^{\mathcal{S}_{k+N-1}})$$

where $\boldsymbol{z}_l^{\mathcal{S}_l}$ denotes the random variables corresponding to the observations of the sensors in the set $\mathcal{S}_l \subseteq \mathcal{S}$ at time l. As discussed in Section 2.3.6, minimizing this quantity with respect to the observation selections $\mathcal{S}_l$ is equivalent to maximizing the following mutual information expression:

$$\sum_{l=k}^{k+N-1} I(\boldsymbol{x}_l; \boldsymbol{z}_l^{\mathcal{S}_l}|\check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1}, \boldsymbol{z}_k^{\mathcal{S}_k}, \dots, \boldsymbol{z}_{l-1}^{\mathcal{S}_{l-1}})$$

Since we are selecting a subset of sensor observations at each time step, this expression can be further decomposed using with an additional application of the chain rule. To do so, we introduce an arbitrary ordering of the elements of $\mathcal{S}_l$, denoting the j-th element by s_l^j, and the first $(j-1)$ elements by by $\mathcal{S}_l^j \triangleq \{s_l^1, \dots, s_l^{j-1}\}$ (*i.e.*, the selection prior to introduction of the j-th element):

$$\sum_{l=k}^{k+N-1} \sum_{j=1}^{|\mathcal{S}_l|} I(\boldsymbol{x}_l; \boldsymbol{z}_l^{s_l^j} | \check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1}, \boldsymbol{z}_k^{\mathcal{S}_k}, \dots, \boldsymbol{z}_{l-1}^{\mathcal{S}_{l-1}}, \boldsymbol{z}_l^{\mathcal{S}_l^j})$$

Our formulation requires this additivity of estimation objective. The algorithm we develop could be applied to other measures of estimation performance, although the objectives which result may not be as natural.

■ 5.1.2 Communications

We assume that any sensor node can communicate with any other sensor node in the network, and that the cost of these communications is known at every sensor node; in practice this will only be required within a small region around each node. In our simulations, the cost (per bit) of direct communication between two nodes is modelled as being proportional to the square distance between the two sensors:

$$\tilde{C}_{ij} \propto ||\boldsymbol{y}^i - \boldsymbol{y}^j||_2^2 \tag{5.1}$$

where $\boldsymbol{y}^s$ is the location of the s-th sensor (which is assumed to be known, *e.g.*, through the calibration procedure as described in [34]). Communications between distant nodes can be performed more efficiently using a multi-hop scheme, in which several sensors relay the message from source to destination. Hence we model the cost of communicating between nodes i and j, C_{ij}, as the length of the shortest path between i and j, using the distances from Eq. (5.1) as arc lengths:

$$C_{ij} = \sum_{k=1}^{n_{ij}} \tilde{C}_{i_{k-1} i_k} \tag{5.2}$$

where $\{i_0, \dots, i_{n_{ij}}\}$ is the shortest path from node $i = i_0$ to node $j = i_{n_{ij}}$. The shortest path distances can be calculated using any shortest path algorithm, such as deterministic dynamic programming or label correcting methods [9]. We assume that the complexity of the probabilistic model (*i.e.*, the number of bits required for transmission) is fixed at B_p bits, such that the energy required to communicate the model from node i to node j is $B_p C_{ij}$. This value will depend on the estimation scheme used in a given implementation; if a particle filter is utilized then the quantity may adapt as the form of the probabilistic model varies, as detailed in [36]. The number of bits in an observation is denoted as B_m, such that the energy required to transmit an observation from node i to node j is $B_m C_{ij}$. These costs may be amended to incorporate the cost of activating the sensor, taking the measurement, the expected number of retransmissions required, *etc*, without changing the structure of the solution.

■ 5.1.3 Constrained communication formulation

Our first formulation optimizes estimation performance subject to a constraint on communication cost. As discussed in Section 5.1.1, we utilize mutual information as our estimation objective, and define the per-stage cost:

$$\begin{aligned} g(\mathbb{X}_k, l_{k-1}, u_k) &= -I(\boldsymbol{x}_k; \boldsymbol{z}_k^{\mathcal{S}_k} | \check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1}) && (5.3) \\ &= -\sum_{j=1}^{|\mathcal{S}_k|} I(\boldsymbol{x}_k; \boldsymbol{z}_k^{s_k^j} | \check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1}, \boldsymbol{z}_k^{\mathcal{S}_k^j}) && (5.4) \end{aligned}$$

We choose the per-stage constraint contribution $G(\mathbb{X}_k, l_{k-1}, u_k)$ to be such that the the expected communication cost over the next N time steps is constrained:

$$G(\mathbb{X}_k, l_{k-1}, u_k) = \left[B_p C_{l_{k-1} l_k} + \sum_{j \in \mathcal{S}_k} B_m C_{l_k j} \right] \tag{5.5}$$

The first term of this communication cost captures the cost of transmitting the representation of the conditional PDF from the previous leader node to the new leader node, while the second captures the cost of transmitting observations from each active sensor to the new leader node.

Substituting the per-stage cost and constraint function into Eq. (2.53), the unconstrained optimization in the Lagrangian (for a particular value of the Lagrange multiplier λ) can be solved conceptually using the recursive dynamic programming equation:

$$J_i^D(\mathbb{X}_i, l_{i-1}, \lambda) = \min_{u_i} \left\{ \bar{g}(\mathbb{X}_i, l_{i-1}, u_i, \lambda) + \mathop{\mathrm{E}}_{\mathbb{X}_{i+1}|\mathbb{X}_i, u_i} J_{i+1}^D(\mathbb{X}_{i+1}, l_i, \lambda) \right\} \tag{5.6}$$

for time indices $i \in \{k, \dots, k+N-1\}$, terminated by $J_{k+N}^D(\mathbb{X}_{k+N}, l_{k+N-1}, \lambda) = -\lambda M$. The conditional PDF at the next time $\mathbb{X}_{i+1}$ is calculated using the recursive Bayes update described in Section 2.1. The augmented per-stage cost combines the information gain and communication cost in a single quantity, adaptively adjusting the weight applied to each using a Lagrange multiplier:

$$\bar{g}(\mathbb{X}_k, l_{k-1}, u_k, \lambda) = -\sum_{j=1}^{|\mathcal{S}_k|} I(\boldsymbol{x}_k; \boldsymbol{z}_k^{s_k^j} | \check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1}, \boldsymbol{z}_k^{\mathcal{S}_k^j}) + \lambda \left[B_p C_{l_{k-1} l_k} + \sum_{j \in \mathcal{S}_k} B_m C_{l_k j} \right] \tag{5.7}$$

This incorporation of the constraint terms into the per-stage cost is a key step, which allows the greedy approximation described in Sections 5.1.7 and 5.1.8 to capture the trade-off between estimation quality and communication cost.

5.1.4 Constrained entropy formulation

The formulation above provides a means of optimizing the information obtained subject to a constraint on the communication energy expended; there is also a closely-related formulation which optimizes the communication energy subject to a constraint on the entropy of probabilistic model of object state. The cost per stage is set to the communication cost expended by the control decision:

$$g(\mathbb{X}_k, l_{k-1}, u_k) = B_p C_{l_{k-1} l_k} + \sum_{j \in \mathcal{S}_k} B_m C_{l_k j} \tag{5.8}$$

We commence by formulating a constraint function on the joint entropy of the state of the object over each time in the planning horizon:

$$\mathrm{E}\{H(\boldsymbol{x}_k, \dots, \boldsymbol{x}_{k+N-1} | \check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1}, \boldsymbol{z}_k^{\mathcal{S}_k}, \dots, \boldsymbol{z}_{k+N-1}^{\mathcal{S}_{k+N-1}})\} \leq H_{\max} \tag{5.9}$$

Manipulating this expression using Eq. (2.72), we obtain

$$-\mathrm{E}\left\{ \sum_{i=k}^{k+N-1} \sum_{j=1}^{|\mathcal{S}_i|} I(\boldsymbol{x}_i; \boldsymbol{z}_l^{s_l^j} | \check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1}, \boldsymbol{z}_k^{\mathcal{S}_k}, \dots, \boldsymbol{z}_{l-1}^{\mathcal{S}_{l-1}}, \boldsymbol{z}_l^{\mathcal{S}_l^j}) \right\} \\ \leq H_{\max} - H(\boldsymbol{x}_k, \dots, \boldsymbol{x}_{k+N-1} | \check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1}) \tag{5.10}$$

from which we set $M = H_{\max} - H(\boldsymbol{x}_k, \dots, \boldsymbol{x}_{k+N-1} | \check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1})$,[1] and

$$G(\mathbb{X}_k, l_{k-1}, u_k) = -\sum_{j=1}^{|\mathcal{S}_k|} I(\boldsymbol{x}_k; \boldsymbol{z}_k^{s_k^j} | \check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1}, \boldsymbol{z}_k^{\mathcal{S}_k^j}) \tag{5.11}$$

[1] In our implementation, we construct a new control policy at each time step by applying the approximate dynamic programming method described in the following section commencing from the current probabilistic model, $\mathbb{X}_k$. At time step k, $H(\boldsymbol{x}_k, \dots, \boldsymbol{x}_{k+N-1} | \check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1})$ is a known constant (representing the uncertainty prior to receiving any observations in the present planning horizon), hence the dependence on $\mathbb{X}_k$ is immaterial.

Following the same procedure as described previously, the elements of the information constraint in Eq. (5.10) can be integrated into the per-stage cost, resulting in a formulation which is identical to Eq. (5.7), except that the Lagrange multiplier is on the mutual information term, rather than the communication cost terms:

$$\bar{g}(\mathbb{X}_k, l_{k-1}, u_k, \lambda) = B_p C_{l_{k-1} l_k} + \sum_{j\in\mathcal{S}_k} B_m C_{l_k j} - \lambda \left[\sum_{j=1}^{|\mathcal{S}_k|} I(\boldsymbol{x}_k; \boldsymbol{z}_k^{\mathcal{S}_k^j} | \check{\boldsymbol{z}}_0, \ldots, \check{\boldsymbol{z}}_{k-1}, \boldsymbol{z}_k^{\mathcal{S}_k^{1:j-1}}) \right] \tag{5.12}$$

■ 5.1.5 Evaluation through Monte Carlo simulation

The constrained dynamic program described above has an infinite state space (the space of probability distributions over object state), hence it cannot be evaluated exactly. The following sections describe a series of approximations which are applied to obtain a practical implementation.

Conceptually, the dynamic program of Eq. (5.6) could be approximated by simulating sequences of observations for each possible sequence of controls. There are $N_s 2^{N_s}$ possible controls at each time step, corresponding all possible selections of leader node and subsets of sensors to activate. The complexity of the simulation process is formidable: to evaluate $\bar{J}_k^D(\mathbb{X}_k, l_{k-1}, \lambda)$ for a given decision state and control, we draw a set of N_p samples of the set of observations $\boldsymbol{z}_k^{\mathcal{S}_k}$ from the distribution $p(\boldsymbol{z}_k^{S_k}|\check{\boldsymbol{z}}_0, \ldots, \check{\boldsymbol{z}}_{k-1})$ derived from $\mathbb{X}_k$, and evaluate the cost to go one step later $\bar{J}_{k+1}^D(\mathbb{X}_{k+1}, l_k, \lambda)$ corresponding to the decision state resulting from each set of observations. The evaluation of each cost to go one step later will yield the same branching. A tree structure develops, where for each previous leaf of the tree, $N_s 2^{N_s} N_p$ new leaves (samples) are drawn, such that the computational complexity increases as $O({N_s}^N 2^{N_s N} {N_p}^N)$ as the tree depth N (*i.e.*, the planning horizon) increases, as illustrated in Fig. 5.1. Such an approach quickly becomes intractable even for a small number of sensors (N_s) and simulated observation samples (N_p), hence we seek to exploit additional structure in the problem to find a computable approximate solution.

■ 5.1.6 Linearized Gaussian approximation

In Section 2.3.4, we showed that the mutual information of a linear-Gaussian observation of a quantity whose prior distribution is also Gaussian is a function only of the prior covariance and observation model, not of the state estimate. Since the covariance of a Kalman filter is independent of observation values (as seen in Section 2.1.2), this result implies that, in the recursion of Eq. (5.6), future rewards depend only on the control values: they are invariant to the observation values that result. It is well-known that this result implies that open loop policies are optimal: we just need to search for control *values* for each time step, rather than control *policies*. Accordingly, in the linear Gaussian case, the growth of the tree discussed in Section 5.1.5 is reduced to $O({N_s}^N 2^{N_s N})$ with the horizon length N, rather than $O({N_s}^N 2^{N_s N} {N_p}^N)$.

While this is a useful result, its applicability to this problem is not immediately clear, as the observation model of interest generally non-linear (such as the model discussed in Section 5.3). However, let us suppose that the observation model can be approximated by linearizing about a nominal state trajectory. If the initial uncertainty is relatively low, the strength of the dynamics noise is relatively low, and the planning horizon length is relatively short (such that deviation from the nominal trajectory is small), then such a linearization approximation may provide adequate fidelity for *planning* of future actions (this approximation is *not* utilized for inference: in our experiments, the SIS algorithm of Section 2.1.4 is used with the nonlinear observation function to maintain the probabilistic model). To obtain the linearization, we fit a Gaussian distribution to the *a priori* PDF (*e.g.*, using Eq. (2.44)); suppose that the resulting distribution is $\mathcal{N}(\boldsymbol{x}_k; \boldsymbol{\mu}_k, \mathbf{P}_k)$. We then calculate the nominal trajectory by calculating the mean at each of the following N steps. In the case of the stationary linear dynamics model discussed in Section 2.1.1:

$$\boldsymbol{x}_k^0 = \boldsymbol{\mu}_k \tag{5.13}$$

$$\boldsymbol{x}_i^0 = \mathbf{F}\boldsymbol{x}_{i-1}^0, \quad i \in \{k+1, \dots, k+N-1\} \tag{5.14}$$

Subsequently, the observation model is approximated using Eq. (2.19) where the linearization point at time i is $\boldsymbol{x}_i^0$. This is a well-known approximation, referred to as the linearized Kalman filter; it is discussed further in Section 2.1.3; it was previously applied to a sensor scheduling problem in [21]. The controller which results has a structure similar to the open loop feedback controller (Section 2.2.2): at each stage a plan for the next N time steps is generated, the first step of the plan executed, and then a new plan for the following N steps is generated, having relinearized after incorporating the newly received observations.

A significant horizon length is required in order to provide an effective trade-off between communication cost and inference quality, since many time steps are required for the long-term communication cost saved and information gained from a leader node change to outweigh the immediate communication cost incurred. While the linear

Gaussian approximation eliminates the $O({N_p}^N)$ factor in the growth of computational complexity with planning horizon length, the complexity is still exponential in both time and the number of sensors, growing as $O({N_s}^N 2^{N_s N})$. The following two sections describe two tree pruning approximations we introduce to obtain a tractable implementation.

5.1.7 Greedy sensor subset selection

To avoid the combinatorial complexity associated with optimization over subsets of sensors, we decompose each decision stage into a number of substages and apply heuristic approximations in a carefully chosen way. Following the application of the linearized Gaussian approximation (Section 5.1.6), the branching of the computation tree of Fig. 5.1 will be reduced to the structure shown in Fig. 5.2. Each stage of control branching involves selection of a leader node, and a subset of sensors to activate; we can break these two phases apart, as illustrated in Fig. 5.3. Finally, one can decompose the choice of which subset of sensors to activate (given a choice of leader node) into a generalized stopping problem [9] in which, at each substage (indexed by i'), the control choices are to terminate (*i.e.*, move on to portion of the tree corresponding to the following time slot) with the current set of selections, or to select an additional sensor. This is illustrated in Fig. 5.4; the branches labelled 'T' represent the decision to terminate with the currently selected subset.

For the communication constrained formulation, the DP recursion becomes:

$$\bar{J}_i(\mathbb{X}_i, l_{i-1}, \lambda) = \min_{l_i \in \mathcal{S}} \{\lambda B_p C_{l_{i-1} l_i} + \bar{J}_i^0(\mathbb{X}_i, l_i, \{\emptyset\}, \lambda)\} \tag{5.15}$$

for $i \in \{k, \dots, k+N-1\}$, terminated by setting $\bar{J}_N(\mathbb{X}_N, l_{N-1}, \lambda) = -\lambda M$, where

$$\begin{aligned} \bar{J}_i^{i'}(\mathbb{X}_i, l_i, \mathcal{S}_i^{i'}, \lambda) = \min \Bigg\{ & \operatorname*{E}_{\mathbb{X}_{i+1} | \mathbb{X}_i, \mathcal{S}_i^{i'}} \bar{J}_{i+1}(\mathbb{X}_{i+1}, l_i, \lambda), \\ & \min_{s_i^{i'} \in \mathcal{S} \setminus \mathcal{S}_i^{i'}} \{\bar{g}(\mathbb{X}_i, l_i, \mathcal{S}_i^{i'}, s_i^{i'}, \lambda) + \bar{J}_i^{i'+1}(\mathbb{X}_i, l_i, \mathcal{S}_i^{i'} \cup \{s_i^{i'}\}, \lambda)\} \Bigg\} \end{aligned} \tag{5.16}$$

$\mathcal{S}_i^{i'}$ is the set of sensors chosen in stage i prior to substage i', and the substage cost $\bar{g}(\mathbb{X}_i, l_i, \mathcal{S}_i^{i'}, s_i^{i'}, \lambda)$ is

$$\bar{g}(\mathbb{X}_i, l_i, \mathcal{S}_i^{i'}, s_i^{i'}, \lambda) = \lambda B_m C_{l_i s_i^{i'}} - I(\boldsymbol{x}_i; \boldsymbol{z}_i^{s_i^{i'}} | \check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{i-1}, \boldsymbol{z}_i^{\mathcal{S}_i^{i'}}) \tag{5.17}$$

The cost to go $\bar{J}_i(\mathbb{X}_i, l_{i-1}, \lambda)$ represents the expected cost to the end of the problem (*i.e.*, the bottom of the computation tree) commencing from the beginning of time slot

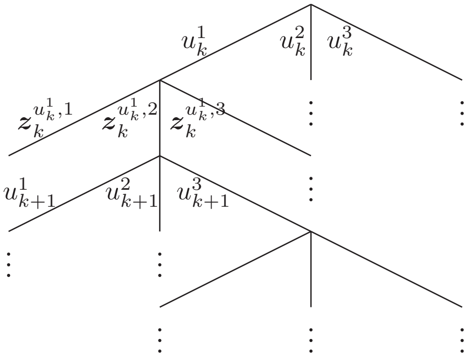

u_k^1
u_k^2
u_k^3
$\boldsymbol{z}_k^{u_k^1,1}$
$\boldsymbol{z}_k^{u_k^1,2}$
$\boldsymbol{z}_k^{u_k^1,3}$
u_{k+1}^1
u_{k+1}^2
u_{k+1}^3

Figure 5.1. Tree structure for evaluation of the dynamic program through simulation. At each stage, a tail sub-problem is required to be evaluated each new control, and a set of simulated values of the resulting observations.

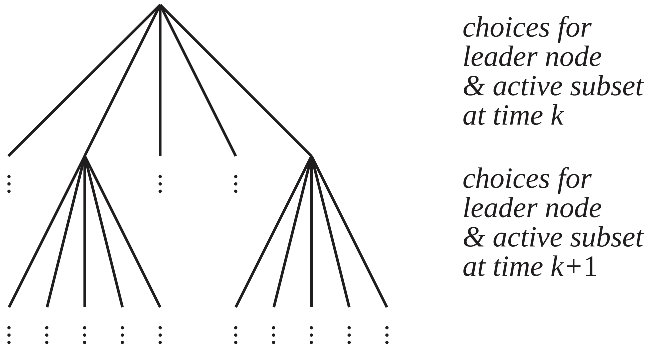

choices for leader node & active subset at time k
choices for leader node & active subset at time k+1

Figure 5.2. Computation tree after applying the linearized Gaussian approximation of Section 5.1.6.

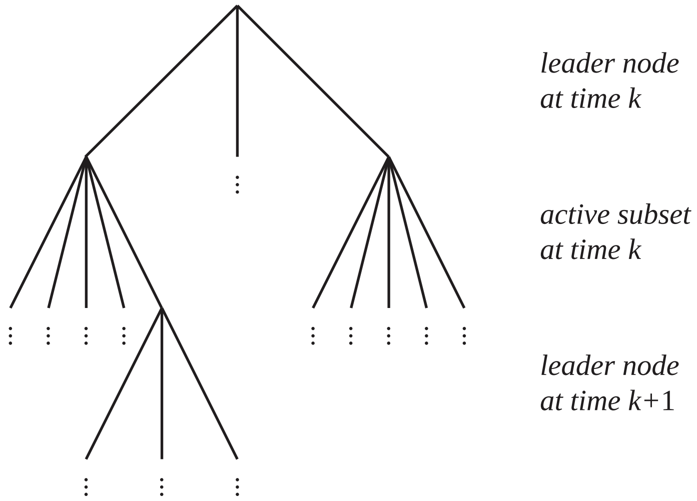


Figure 5.3. Computation tree equivalent to Fig. 5.2, resulting from decomposition of control choices into distinct stages, selecting leader node for each stage and then selecting the subset of sensors to activate.

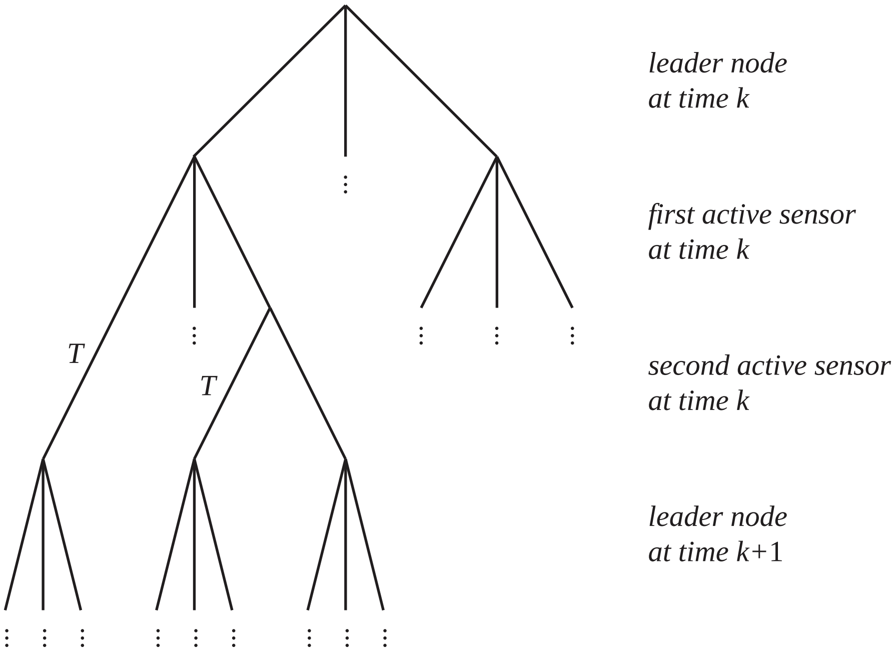


Figure 5.4. Computation tree equivalent to Fig. 5.2 and Fig. 5.3, resulting from further decomposing sensor subset selection problem into a generalized stopping problem, in which each substage allows one to terminate and move onto the next time slot with the current set of selected sensors, or to add an additional sensor.

i (*i.e.*, the position of the tree in Fig. 5.4 where branching occurs over choices of leader node for that time slot). The function $\bar{J}_i^{i'}(\mathbb{X}_i, l_i, \mathcal{S}_i^{i'}, \lambda)$, represents the cost to go from substage i' of stage i to the end of the problem, *i.e.*, the expected cost to go to the bottom of the tree commencing from a partial selection of which sensors to activate at time i. The first choice in the outer minimization in Eq. (5.16) represents the choice to terminate (*i.e.*, move on to the next time slot) with the currently selected subset of sensors, while the second represents the choices of additional sensors to select.

While this formulation is algebraically equivalent to the original problem, it is in a form which is more suited to approximation. Namely, the substages which form a generalized stopping problem may be performed using a greedy method, in which, at each stage, if there is no sensor $s_i^{i'}$ for which the substage cost $\bar{g}(\mathbb{X}_i, l_i, \mathcal{S}_i^{i'}, s_i^{i'}, \lambda) \leq 0$ (*i.e.*, for which the cost of transmitting the observation is not outweighed by the expected information it will provide), then we progress to the next stage; otherwise the sensor $s_i^{i'}$ with the lowest substage cost is added. The fact that the constraint terms of the Lagrangian were distributed into the per-stage and per-substage cost allows the greedy approximation to be used in a way which trades off estimation quality and communication cost.

While worst-case complexity of this algorithm is $O(N_s^2)$, careful analysis of the sensor model can yield substantial practical reductions. One quite general simplification can be made: assuming that sensor measurements are independent conditioned on the state, one can show that, for the substage cost in Eq. (5.17) since the first term is constant with respect to $\mathcal{S}_i^{i'}$ and the second is submodular: (assuming that observations are independent conditioned on the state)

$$\bar{g}(\mathbb{X}_i, l_i, \mathcal{S}_i^{i'}, s, \lambda) \leq \bar{g}(\mathbb{X}_i, l_i, \mathcal{S}_i^{i''}, s, \lambda) \quad \forall\, i' < i'' \tag{5.18}$$

Using this result, if at any substage of stage i we find that the substage cost of adding a particular sensor is greater then zero (so that the augmented cost of activating the sensor is higher than the augmented cost of terminating), then that sensor will not be selected in any later substages of stage i (as the cost cannot decrease as we add more sensors), hence it can be excluded from consideration. In practice this will limit the sensors requiring consideration to those in a small neighborhood around the current leader node and object, reducing computational complexity when dealing with large networks.

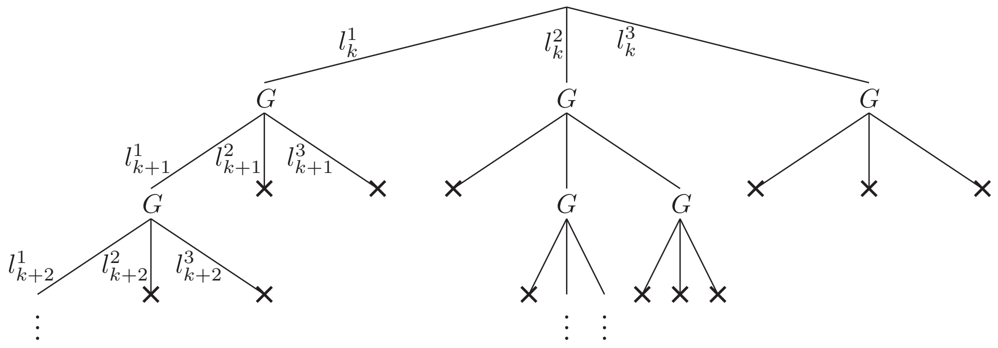


Figure 5.5. Tree structure for n-scan pruning algorithm with $n = 1$. At each stage new leaves are generated extending each remaining sequence with using each new leader node. Subsequently, all but the best sequence ending with each leader node is discarded (marked with '×'), and the remaining sequences are extended using greedy sensor subset selection (marked with 'G').

5.1.8 n-Scan pruning

The algorithm described above is embedded within a slightly less coarse approximation for leader node selection, which incorporates costs over multiple time stages. This approximation operates similarly to the n-scan pruning algorithm, which is commonly used to control computational complexity in the Multiple Hypothesis Tracker [58]. Setting $n = 1$, the algorithm is illustrated in Fig. 5.5. We commence by considering each possible choice of leader node[2] for the next time step and calculating the greedy sensor subset selection from Section 5.1.7 for each leader node choice (the decisions made in each of these branches will differ since the sensors must transmit their observations to a different leader node, incurring a different communication cost). Then, for each leaf node, we consider the candidate leader nodes at the following time step. All sequences ending with the same candidate leader node are compared, the one with the lowest cost value is kept, and the other sequences are discarded. Thus, at each stage, we keep some approximation of the best control trajectory which ends with each sensor as leader node.

Using such an algorithm, the tree width is constrained to the number of sensors, and the overall worst case computational complexity is $O(N{N_s}^3)$ (in practice, at each stage we only consider candidate sensors in some neighborhood of the estimated object location, and the complexity will be substantially lower). This compares to the simulation-based evaluation of the full dynamic programming recursion which, as discussed in Section 5.1.5, has a computation complexity of the order $O({N_s}^N 2^{N_s N} {N_p}^N)$. The difference in complexity is striking: even for a problem with $N_s = 20$ sensors, a planning horizon of $N = 10$ and simulating $N_p = 50$ values of observations at each stage, the complexity is reduced from 1.6×10^{90} to (at worst case) 8×10^5.

Because the communication cost structure is Markovian with respect to the leader node (*i.e.*, the communication cost of a particular future control trajectory is unaffected by the control history given the current leader node), it is captured perfectly by this model. The information reward structure, which is not Markovian with respect to the leader node, is approximated using the greedy method.

5.1.9 Sequential subgradient update

The previous two sections provide an efficient algorithm for generating a plan for the next N steps given a particular value of the dual variable λ. Substituting the resulting

[2] The set of candidate leader nodes would, in practice, be limited to sensors close to the object, similar to the sensor subset selection.

plan into Eq. (2.56) yields a subgradient which can be used to update the dual variables (under the linear Gaussian approximation, feedback policies correspond to open loop plans, hence the argument of the expectation of $\text{E}[\sum_i G(\mathbb{X}_i, l_{i-1}, \mu_i(\mathbb{X}_i, l_{i-1})) - M]$ is deterministic). A full subgradient implementation would require evaluation for many different values of the dual variable each time re-planning is performed, which is undesirable since each evaluation incurs a substantial computational cost.[3] Since the planning is over many time steps, in practice the level of the constraint (*i.e.*, the value of $\text{E}[\sum_i G(\mathbb{X}_i, l_{i-1}, \mu_i(\mathbb{X}_i, l_{i-1})) - M]$) will vary little between time steps, hence the slow adaptation of the dual variable provided by a single subgradient step in each iteration may provide an adequate approximation.

In the experiments which follow, at each time step we plan using a single value of the dual variable, and then update it for the next time step utilizing either an additive update:

$$\lambda_{k+1} = \begin{cases} \min\{\lambda_k + \gamma^+, \lambda^{\max}\}, & \text{E}[\sum_i G(\mathbb{X}_i, l_{i-1}, \mu_i(\mathbb{X}_i, l_{i-1}))] > M \\ \max\{\lambda_k - \gamma^-, \lambda^{\min}\}, & \text{E}[\sum_i G(\mathbb{X}_i, l_{i-1}, \mu_i(\mathbb{X}_i, l_{i-1}))] \leq M \end{cases} \tag{5.19}$$

or a multiplicative update:

$$\lambda_{k+1} = \begin{cases} \min\{\lambda_k \beta^+, \lambda^{\max}\}, & \text{E}[\sum_i G(\mathbb{X}_i, l_{i-1}, \mu_i(\mathbb{X}_i, l_{i-1}))] > M \\ \max\{\lambda_k / \beta^-, \lambda^{\min}\}, & \text{E}[\sum_i G(\mathbb{X}_i, l_{i-1}, \mu_i(\mathbb{X}_i, l_{i-1}))] \leq M \end{cases} \tag{5.20}$$

where γ^+ and γ^- are the increment and decrement sizes, β^+ and β^- are the increment and decrement factors, and $\lambda^{\max}$ and $\lambda^{\min}$ are the maximum and minimum values of the dual variable. It is necessary to limit the values of the dual variable since the constrained problem may not be feasible. If the variable is not constrained, undesirable behavior can result such as utilizing every sensor in a sensor network in order to meet an information constraint which cannot be met in any case, or because the dual variable in the communication constraint was adapted such that it became too low, effectively implying that communications are cost-free.

The dual variables may be initialized using several subgradient iterations or some form of line search when the algorithm is first executed in order to commence with a value in the right range.

[3] The rolling horizon formulation necessitates re-optimization of the dual variable at every time step, as opposed to [18].

5.1.10 Roll-out

If the horizon length is set to be too small in the communications constrained formulation, then the resulting solution will be to hold the leader node fixed, and take progressively fewer observations. To prevent this degenerate behavior, we use a roll-out approach (a commonly used suboptimal control methodology), in which we add to the terminal cost in the DP recursion (Eq. (5.6)) the cost of transmitting the probabilistic model to the sensor with the smallest expected distance to the object at the final stage. Denoting by $\tilde{\mu}(\mathbb{X}_k) \in \mathcal{S}$ the policy which selects as leader node the sensor with the smallest expected distance to the object, the terminal cost is:

$$J_{k+N}(\mathbb{X}_{k+N}, l_{k+N-1}) = \lambda B_p C_{l_{k+N-1}\tilde{\mu}(\mathbb{X}_{k+N})} \tag{5.21}$$

where the Lagrange multiplier λ is included only in the communication-constrained case. This effectively acts as the cost of the base policy in a roll-out [9]. The resulting algorithm constructs a plan which assumes that, at the final stage, the leader node will have to be transferred to the closest sensor, hence there is no benefit in holding it at its existing location indefinitely. In the communication-constrained case, this modification can make the problem infeasible for short planning horizons, but the limiting of the dual variables discussed in Section 5.1.9 can avoid anomalous behavior.

5.1.11 Surrogate constraints

A form of information constraint which is often more desirable is one which captures the notion that it is acceptable for the uncertainty in object state to increase for short periods of time if informative observations are likely to become available later. The minimum entropy constraint is such an example:

$$\mathrm{E}\left\{ \min_{i \in \{k, \dots, k+N-1\}} H(\boldsymbol{x}_i | \check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{i-1}) - H_{\max} \right\} \leq 0 \tag{5.22}$$

The constraint in Eq. (5.22) does not have an additive decomposition (*cf* Eq. (5.10)), as required by the approximations in Sections 5.1.7 and 5.1.8. However, we can use the constraint in Eq. (5.10) to generate plans for a given value of the dual variable λ using the approximations, and then perform the dual variable update of Section 5.1.9 using the desired constraint, Eq. (5.22). This simple approximation effectively uses the additive constraint in Eq. (5.10) as a surrogate for the desired constraint in Eq. (5.22), allowing us to use the computationally convenient method described above with a more meaningful criterion.

5.2 Decoupled Leader Node Selection

Most of the sensor management strategies proposed for object localization in existing literature seek to optimize the estimation performance of the system, incorporating communication cost indirectly, such as by limiting the maximum number of sensors utilized. These methods typically do not consider the leader node selection problem directly, although the communication cost consumed in implementing them will vary depending on the leader node since communications costs are dependent on the transmission distance. In order to compare the performance of the algorithm developed in Section 5.1 with these methods, we develop an approach which, conditioned on a particular sensor management strategy (that is insensitive to the choice of leader node), seeks to dynamically select the leader node to minimize the communications energy consumed due to activation, deactivation and querying of sensors by the leader node, and transmission of observations from sensors to the leader node. This involves a trade-off between two different forms of communication: the large, infrequent step increments produced when the probability distribution is transferred from sensor to sensor during leader node hand-off, and the small, frequent increments produced by activating, deactivating and querying sensors. The approach is fundamentally different from that in Section 5.1 as we are optimizing the leader node selection conditioned on a fixed sensor management strategy, rather than jointly optimizing sensor management and leader node selection.

5.2.1 Formulation

The objective which we seek to minimize is the expected communications cost over an N-step rolling horizon. We require the sensor management algorithm to provide predictions of the communications performed by each sensor at each time in the future. As in Section 5.1, the problem corresponds to a dynamic program in which the decision state at time k is the combination of the conditional PDF of object state, $\mathbb{X}_k \triangleq p(\boldsymbol{x}_k|\check{\boldsymbol{z}}_0, \dots, \check{\boldsymbol{z}}_{k-1})$, and the previous leader node, l_{k-1}. The control which we may choose is the leader node at each time, $u_k = l_k \in \mathcal{S}$. Denoting the expected cost of communications expended by the sensor management algorithm (due to sensor activation and deactivation, querying and transmission of obervations) at time k if the leader node is l_k as $g_c(\mathbb{X}_k, l_k)$, the dynamic program for selecting the leader node at time k can be written as the following recursive equation:

$$J_i(\mathbb{X}_i, l_{i-1}) = \min_{l_i \in \mathcal{S}} \left\{ g_c(\mathbb{X}_i, l_i) + B_p C_{l_{i-1} l_i} + \mathop{\mathrm{E}}_{\mathbb{X}_{i+1}|\mathbb{X}_i, l_i} J_{i+1}(\mathbb{X}_{i+1}, l_i) \right\} \tag{5.23}$$

for $i \in \{k, \dots, k+N-1\}$. In the same way as discussed in Section 5.1.10, we set the terminal cost to the cost of transmitting the probabilistic model from the current leader node to the node with the smallest expected distance to the object, $\tilde{\mu}(\mathbb{X}_{k+N})$:

$$J_{k+N}(\mathbb{X}_{k+N}, l_{k+N-1}) = B_p C_{l_{k+N-1}\tilde{\mu}(\mathbb{X}_{k+N})} \tag{5.24}$$

In Section 5.3 we apply this method using a single look-ahead step ($N = 1$) with a greedy sensor management strategy selecting firstly the most informative observation, and then secondly the two most informative observations.

5.3 Simulation results

As an example of the employment of our algorithm, we simulate a scenario involving an object moving through a network of sensors. The state of the object is position and velocity in two dimensions ($\boldsymbol{x}_k = [p_x \; v_x \; p_y \; v_y]^T$); the state evolves according to the nominally constant velocity model described in Eq. (2.8), with $\Delta t = 0.25$ and $q = 10^{-2}$.

The simulation involves $N_s = 20$ sensors positioned randomly according to a uniform distribution inside a 100×100 unit region; each trial used a different sensor layout and object trajectory. Denoting the measurement taken by sensor $s \in \mathcal{S} = \{1 : N_s\}$ (where N_s is the number of sensors) at time k as z^s_k, a nonlinear observation model is assumed:

$$z^s_k = h(\boldsymbol{x}_k, s) + v^s_k \tag{5.25}$$

where $v^s_k \sim \mathcal{N}\{v^s_k; 0, 1\}$ is a white Gaussian noise process, independent of $\boldsymbol{w}_k \; \forall \; k$ and of $v^j_k, j \neq s \; \forall \; k$. The observation function $h(\cdot, s)$ is a quasi-range measurement, *e.g.*, resulting from measuring the intensity of an acoustic emission of known amplitude:

$$h(\boldsymbol{x}_k, s) = \frac{a}{||\mathbf{L}\boldsymbol{x}_k - \boldsymbol{y}^s||_2^2 + b} \tag{5.26}$$

where $\mathbf{L}$ is the matrix which extracts the position of the object from the object state (such that $\mathbf{L}\boldsymbol{x}_k$ is the location of the object), and $\boldsymbol{y}^s$ is the location of the s-th sensor. The constants a and b can be tuned to model the signal-to-noise ratio of the sensor, and the rate at which the signal-to-noise ratio decreases as distance increases; we use $a = 2000$ and $b = 100$. The information provided by the observation reduces as the range increases due to the nonlinearity.

As described in Section 2.1.3, the measurement function $h(\cdot, s)$ can be approximated as a first-order Taylor series truncation in a small vicinity around a nominal point $\boldsymbol{x}^0$:

$$z^s_k \approx h(\boldsymbol{x}^0, s) + \mathbf{H}^s(\boldsymbol{x}^0)(\boldsymbol{x}_k - \boldsymbol{x}^0) + \boldsymbol{v}^s_k$$
$$\mathbf{H}^s(\boldsymbol{x}^0) = \boldsymbol{\nabla}_{\boldsymbol{x}} h(\boldsymbol{x}, s)|_{\boldsymbol{x}=\boldsymbol{x}^0}$$

where:

$$\mathbf{H}^s(\boldsymbol{x}^0) = \frac{-2a}{(||\mathbf{L}\boldsymbol{x}^0 - \boldsymbol{y}^s||_2^2 + b)^2}(\mathbf{L}\boldsymbol{x}^0 - \boldsymbol{y}^s)^T\mathbf{L} \tag{5.27}$$

This approximation is used for planning as discussed in Section 5.1.6; the particle filter described in Section 2.1.4 is used for inference.

The model was simulated for 100 Monte Carlo trials. The initial position of the object is in one corner of the region, and the initial velocity is 2 units per second in each dimension, moving into the region. The simulation ends when the object leaves the 100×100 region or after 200 time steps, which ever occurs sooner (the average length is around 180 steps). The communication costs were $B_p = 64$ and $B_m = 1$, so that the cost of transmitting the probabilistic model is 64× the cost of transmitting an observation. For the communication-constrained problem, a multiplicative update was used for the subgradient method, with $\beta^+ = \beta^- = 1.2$, $\lambda^{\min} = 10^{-5}$, $\lambda^{\max} = 5\times10^{-3}$, and $C_{\max} = 10N$ where N is the planning horizon length. For the information-constrained problem, an additive update was used for the subgradient method, with $\gamma^+ = 50$, $\gamma^- = 250$, $\lambda^{\min} = 10^{-8}$, $\lambda^{\max} = 500$ and $H_{\max} = 2$ (these parameters were determined experimentally).

The simulation results are summarized in Fig. 5.6. The top diagram demonstrates that the communication-constrained formulation provides a way of controlling sensor selection and leader node which reduces the communication cost and improves estimation performance substantially over the myopic single-sensor methods, which at each time activate and select as leader node the sensor with the observation producing the largest expected reduction in entropy. The information-constrained formulation allows for an additional saving in communication cost while meeting an estimation criterion wherever possible.

The top diagram in Fig. 5.6 also illustrates the improvement which results from utilizing a longer planning horizon. The constraint level in the communication-constrained case is 10 cost units per time step; since the average simulation length is 180 steps, the average communication cost if the constraint were always met with equality would be 1800. However, because this cost tends to occur in bursts (due to the irregular hand-off of leader node from sensor to sensor as the object moves), the practical behavior of the system is to reduce the dual variable when there is no hand-off in the planning horizon (allowing more sensor observations to be utilized), and increase it when there is a hand-off in the planning horizon (to come closer to meeting the constraint). A longer planning horizon reduces this undesirable behavior by anticipating upcoming leader node hand-off events earlier, and tempering spending of communication resources sooner. This is

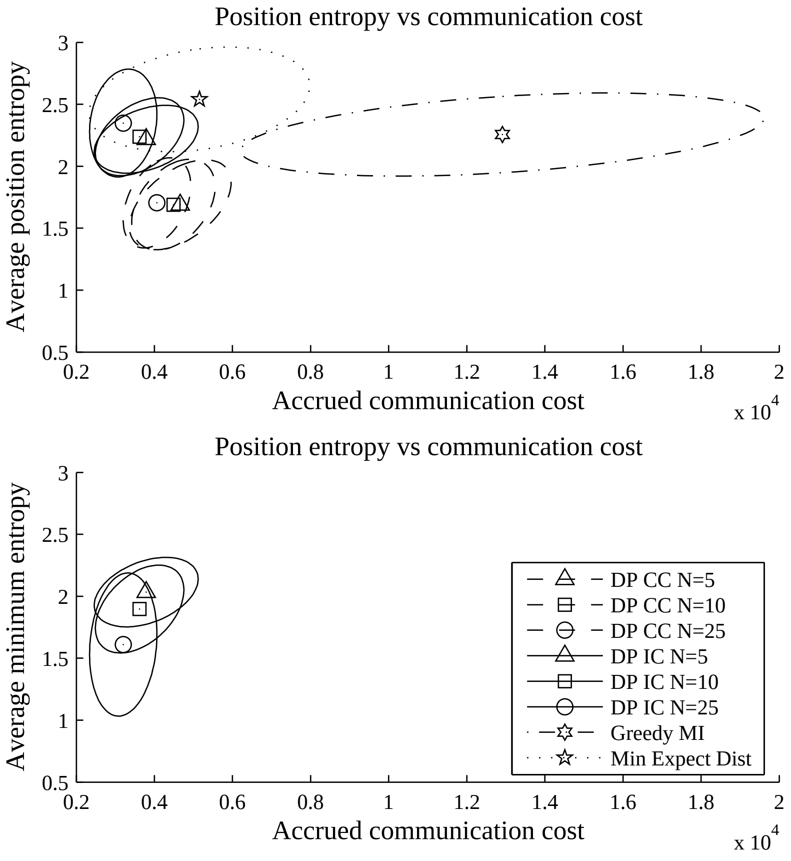


Figure 5.6. Position entropy and communication cost for dynamic programming method with communication constraint (DP CC) and information constraint (DP IC) with different planning horizon lengths (N), compared to the methods selecting as leader node and activating the sensor with the largest mutual information (greedy MI), and the sensor with the smallest expected square distance to the object (min expect dist). Ellipse centers show the mean in each axis over 100 Monte Carlo runs; ellipses illustrate covariance, providing an indication of the variability across simulations. Upper figure compares average position entropy to communication cost, while lower figure compares average of the minimum entropy over blocks of the same length as the planning horizon (*i.e.*, the quantity to which the constraint is applied) to communication cost.

demonstrated in Fig. 5.7, which shows the adaptation of the dual variable for a single Monte Carlo run.

In the information-constrained case, increasing the planning horizon relaxes the constraint, since it requires the *minimum* entropy within the planning horizon to be less than a given value. Accordingly, using a longer planning horizon, the average minimum entropy is reduced, and additional communication energy is saved. The lower diagram in Fig. 5.6 shows the average minimum entropy in blocks of the same length as the planning horizon, demonstrating that the information constraint is met more often with a longer planning horizon (as well as resulting in a larger communication saving).

Fig. 5.8 compares the adaptive Lagrangian relaxation method discussed in Section 5.1 with the decoupled scheme discussed in Section 5.2, which adaptively selects the leader node to minimize the expected communication cost expended in implementing the decision of the fixed sensor management method. The fixed sensor management scheme activates the sensor or two sensors with the observation or observations producing the largest expected reduction in entropy. The results demonstrate that for this case the decoupled method using a single sensor at each time step results in similar estimation performance and communication cost to the Lagrangian relaxation method using an information constraint with the given level. Similarly, the decoupled method using two sensors at each time step results in similar estimation performance and communication cost to the Lagrangian relaxation method using a communication constraint with the given level. The additional flexibility of the Lagrangian relaxation method allows one to select the constraint level to achieve various points on the estimation performance/communication cost trade-off, rather than being restricted to particular points corresponding to different numbers of sensors.

■ 5.4 Conclusion and future work

This paper has demonstrated how an adaptive Lagrangian relaxation can be utilized for sensor management in an energy-constrained sensor network. The introduction of secondary objectives as constraints provides a natural methodology to address the trade-off between estimation performance and communication cost.

The planning algorithm may be applied alongside a wide range of estimation methods, ranging from the Kalman filter to the particle filter. The algorithm is also applicable to a wide range of sensor models. The linearized Gaussian approximation in Section 5.1.6 results in a structure identical to the OLFC. The remainder of our algorithm (removing the linearized Gaussian approximation) may be applied to find

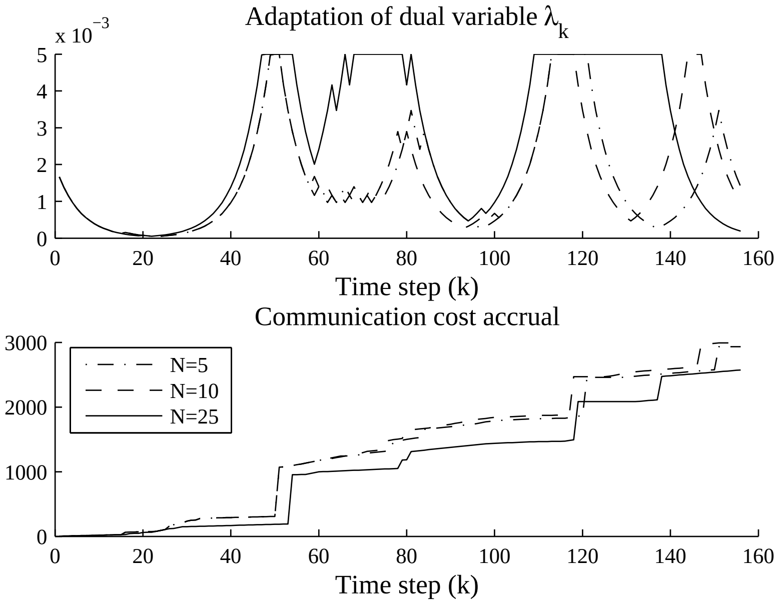


Figure 5.7. Adaptation of communication constraint dual variable λ_k for different horizon lengths for a single Monte Carlo run, and corresponding cumulative communication costs.

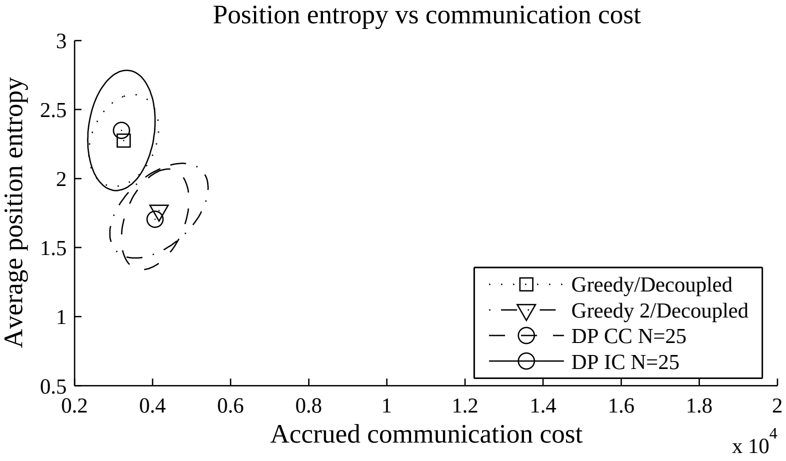


Figure 5.8. Position entropy and communication cost for dynamic programming method with communication constraint (DP CC) and information constraint (DP IC), compared to the method which dynamically selects the leader node to minimize the expected communication cost consumed in implementing a fixed sensor management scheme. The fixed sensor management scheme activates the sensor ('greedy') or two sensors ('greedy 2') with the observation or observations producing the largest expected reduction in entropy. Ellipse centers show the mean in each axis over 100 Monte Carlo runs; ellipses illustrate covariance, providing an indication of the variability across simulations.

an efficient approximation of the OLFC as long as an efficient estimate of the reward function (mutual information in our case) is available.

The simulation results in Section 5.3 demonstrate that approximations based on dynamic programming are able to provide similar estimation performance (as measured by entropy), for a fraction of the communication cost in comparison to simple heuristics which consider estimation performance alone and utilize a single sensor. The discussion in Section 5.1.7 provides a guide for efficient implementation strategies that can enable implementation on the latest generation wireless sensor networks. Future work includes incorporation of the impact on planning caused by the interaction between objects when multiple objects are observed by a single sensor, and developing approximations which are less coarse than the linearized Gaussian model.

Chapter 6

Contributions and future directions

THE preceding chapters have extended existing sensor management methods in three ways: firstly, obtaining performance guarantees for sequential sensor management problems; secondly, finding an efficient integer programming solution that exploits the structure of beam steering; and finally, finding an efficient heuristic sensor management method for object tracking in sensor networks. This chapter briefly summarizes these contributions, before outlining suggestions of areas for further investigation.

■ 6.1 Summary of contributions

The following sections outline the contributions made in this thesis.

■ 6.1.1 Performance guarantees for greedy heuristics

The analysis in Chapter 3 extends the recent work in [46] to the sequential problem structures that commonly arise in waveform selection and beam steering. The extension is quite general in that it applies to arbitrary, time varying observation and dynamical models. Extensions include tighter bounds that exploit either process diffusiveness or objectives involving discount factors, and applicability to closed loop problems. The results apply to objectives including mutual information; the log-determinant of the Fisher information matrix was also shown to be submodular, yielding a guarantee on the posterior Cramér-Rao bound. Examples demonstrate that the bounds are tight, and counterexamples illuminate larger classes of problems to which they do not apply.

The results are the first of their type for sequential problems, and effectively justify the use of the greedy heuristic in certain contexts, delineating problems in which additional open loop planning can be beneficial from those in which it cannot. For example, if a factor of 0.5 of the optimal performance is adequate, then additional planning is unnecessary for any problem that fits within the structure. The online guarantees confirm cases in which the greedy heuristic is even closer to optimality.

6.1.2 Efficient solution for beam steering problems

The analysis in Chapter 4 exploits the special structure in problems involving large numbers of independent objects to find an efficient solution of the beam steering problem. The analysis from Chapter 3 was utilized to obtain an upper bound on the objective function. Solutions with guaranteed near-optimality were found by simultaneously reducing the upper bound and raising a matching lower bound.

The algorithm has quite general applicability, admitting time varying observation and dynamical models, and observations requiring different time durations to complete. An alternative formulation that was specialized to time invariant rewards provided a further computational saving. The methods are applicable to a wide range of objectives, including mutual information and the posterior Cramér-Rao bound.

Computational experiments demonstrated application to problems involving 50–80 objects planning over horizons up to 60 time slots. Performing planning of this type through full enumeration would require evaluation of the reward of more than 10^{100} different observation sequences. As well as demonstrating that the algorithm is suitable for online application, these experiments also illustrate a new capability for exploring the benefit that is possible through utilizing longer planning horizons. For example, we have quantified the small benefit of additional open loop planning in problems where models exhibit low degrees of non-stationarity.

6.1.3 Sensor network management

In Chapter 5, we presented a method trading off estimation performance and energy consumed in an object tracking problem. The trade off between these two quantities was formulated by maximizing estimation performance subject to a constraint on energy cost, or the dual of this, *i.e.*, minimizing energy cost subject to a constraint on estimation performance. Our analysis has proposed a planning method that is both computable and scalable, yet still captures the essential structure of the underlying trade off. The simplifications enable computation over much longer planning horizons: *e.g.*, in a problem involving $N_s = 20$ sensors, closed loop planning over a horizon of $N = 20$ time steps using $N_p = 50$ simulated values of observations at each stage would involve complexity of the order $O([N_s 2^{N_s}]^N N_p^N) \approx 10^{180}$; the simplifications yield worst-case computation of the order $O(NN_s^3) = 1.6 \times 10^5$. Simulation results demonstrate the dramatic reduction in the communication cost required to achieve a given estimation performance level as compared to previously proposed algorithms. The approximations are applicable to a wide range of problems; *e.g.*, even if the linearized Gaussian assumption is relaxed, the remaining approximations may be applied to find an efficient approximation of the open loop feedback controller as long as an efficient estimate of the reward function (mutual information in our case) is available.

■ 6.2 Recommendations for future work

The following sections describe some promising areas for future investigation.

■ 6.2.1 Performance guarantees

Chapter 3 has explored several performance guarantees that are possible through exploitation of submodular objectives, as well as some of the boundaries preventing wider application. Directions in which this analysis may be extended include those outlined in the following paragraphs.

Guarantees for longer look-ahead lengths

It is easy to show that no stronger guarantees exist for heuristics using longer look-ahead lengths for general models; *e.g.*, if we introduce additional time slots, in which all observations are uninformative, in between the two original time slots in Example 3.1, we can obtain the same factor of 0.5 for any look-ahead length. However, under diffusive assumptions, one would expect that additional look-ahead steps would yield an algorithm that is closer to optimal.

Observations consuming different resources

Our analysis inherently assumes that all observations utilize the same resources: the same options available to us in later stages regardless of the choice we make in the current stage. In [46], a guarantee is obtained for the subset selection problem in which each observation j has a resource consumption c_j, and we seek the most informative subset $\mathcal{A}$ of observations for which $\sum_{j\in\mathcal{A}} c_j \leq C$. Expanding this analysis to sequential selection problems involving either a single resource constraint encompassing all time slots, or separate resource constraints for each time slot, would be an interesting extension. A guarantee with a factor of $(e-1)/(2e-1) \approx 0.387$ can be obtained quite easily in the latter case, but it may be possible to obtain tighter guarantees (*i.e.*, 0.5).

Closed loop guarantees

Example 3.3 establishes that there is no guarantee on the ratio of the performance of the greedy heuristic operating in closed loop to the performance of the optimal closed loop controller. However, it may be possible to introduce additional structure (*e.g.*, diffusiveness and/or limited bandwidth observations) to obtain some form of weakened guarantee.

Stronger guarantees exploiting additional structure

Finally, while the guarantees in Chapter 3 have been shown to be tight within the level of generality to which they apply, it may be possible to obtain stronger guarantees for problems with specific structure, *e.g.*, linear Gaussian problems with dynamics and observation models satisfying particular properties. An example of this is the result that greedy heuristics are *optimal* for beam steering of one-dimensional linear Gaussian systems [33].

■ 6.2.2 Integer programming formulation of beam steering

Chapter 4 proposed a new method for efficient solution of beam steering problems, and explored its performance and computation complexity. There are several areas in which this development may be extended, as outlined in the following sections.

Alternative update algorithms

The algorithm presented in Section 4.3.3 represents one of many ways in which the update between iterations could be performed. It remains to explore the relative benefits of other update algorithms; *e.g.*, generating candidate subsets for each of the chosen exploration subset elements, rather than just the one with the highest reward increment.

Deferred reward calculation

It may be beneficial to defer calculation of some of the incremental rewards, *e.g.*, the incremental reward of an exploration subset element conditioned on a given candidate subset is low enough that it is unlikely to be chosen, it would seem unnecessary to recalculate the incremental reward when the candidate subset is extended.

Accelerated search for lower bounds

The lower bound in Section 4.3.6 utilizes the results of the Algorithm 4.1 to find the best solution amongst those explored so far (*i.e.*, those for which the exact reward has been evaluated). However, the decisions made by Algorithm 4.1 tend to focus on reducing the upper bound to the reward rather than on finding solutions for which the reward is high. It may be beneficial to incorporate heuristic searches that introduce additional candidate subsets that appear promising in order to raise the lower bound quickly as well as decreasing the upper bound. One example of this would be to include candidate subsets corresponding to the decisions made by the greedy heuristic—this would ensure that a solution at least as good as that of the greedy heuristic will be obtained regardless of when the algorithm is terminated.

Integration into branch and bound procedure

In the existing implementation, changes made between iterations of the integer program force the solver to restart the optimization. A major computational saving may result if a method is found that allows the solution of the previous iteration to be applied to the new solution. This is easily performed in linear programming problems, but is more difficult in integer programming problems since the bounds previously evaluated in the branch and bound process are (in general) invalidated. A further extension along the same line would be to integrate the algorithm for generating new candidate sets with the branch and bound procedure for the integer program.

■ 6.2.3 Sensor network management

Chapter 5 provides a computable method for tracking an object using a sensor network, with demonstrated empirical performance. Possible extensions include multiple objects and performance guarantees.

Problems involving multiple objects

While the discussion in Chapter 5 focused on the case of a single object, the concept may be easily extended to multiple objects. When objects are well-separated in space, one may utilize a parallel instance of the algorithm from Chapter 5 for each object. When objects become close together and observations induce conditional dependency in their states, one may either store the joint conditional distribution of the object group at one sensor, or utilize a distributed representation across two or more sensors. In the former case, there will be a control choice corresponding to breaking the joint distribution into its marginals after the objects separate again. This will result in a loss of information and a saving in communication cost, both of which could be incorporated into the trade-off performed by the constrained optimization. In the case of a distributed representation of the joint conditional distribution, it will be necessary to quantify both the benefit (in terms of estimation performance) and cost of each of the communications involved in manipulating the distributed representation.

Performance guarantees

The algorithm presented in Chapter 5 does not possess any performance guarantee; Section 3.9 provides an example of a situation in which one element of the approximate algorithm performs poorly. An extension of both Chapters 3 and 5 is to exploit additional problem structure and amend the algorithm to guarantee performance.

Bibliography

[1] Karim M. Abadir and Jan R. Magnus. *Matrix Algebra.* Cambridge University Press, 2005.

[2] Eitan Altman. *Constrained Markov decision processes.* Chapman and Hall, London, UK, 1999.

[3] Brian D. O. Anderson and John B. Moore. *Optimal filtering.* Prentice-Hall, Englewood Cliffs, NJ, 1979.

[4] M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[5] Lawrence M. Ausubel. An efficient ascending-bid auction for multiple objects. *American Economic Review*, 94(5):1452–1475, December 2004.

[6] Yaakov Bar-Shalom and Xiao-Rong Li. *Estimation and Tracking: Principles, Techniques and Software.* Artech House, Norwood, MA, 1993.

[7] M Behara. *Additive and Nonadditive Measures of Entropy.* Wiley Eastern Ltd, New Delhi, India, 1990.

[8] P.E. Berry and D.A.B. Fogg. On the use of entropy for optimal radar resource management and control. In *Radar Conference, 2003. Proceedings of the International*, pages 572–577, 2003.

[9] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, Belmont, MA, second edition, 2000.

[10] Dimitri P. Bertsekas. *Nonlinear Programming.* Athena Scientific, Belmont, MA, second edition, 1999.

[11] D.P. Bertsekas. Auction algorithms for network flow problems: A tutorial introduction. *Computational Optimization and Applications*, 1:7–66, 1992.

[12] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, Belmont, MA, 1997.

[13] Frederick J. Beutler and Keith W. Ross. Optimal policies for controlled Markov chains with a constraint. *Journal of Mathematical Analysis and Applications*, 112 (1):236–252, November 1985.

[14] Samuel S. Blackman and Robert Popoli. *Design and Analysis of Modern Tracking Systems.* Artech House, Norwood, MA, 1999.

[15] V.D. Blondel and John N. Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274, September 2000.

[16] David A. Castañón. Stochastic control bounds on sensor network performance. In *IEEE Conference on Decision and Control*, pages 4939–4944, 2005.

[17] David A. Castañón. Optimal search strategies in dynamic hypothesis testing. *Systems, Man and Cybernetics, IEEE Transactions on*, 25(7):1130–1138, July 1995.

[18] David A. Castañón. Approximate dynamic programming for sensor management. In *Proc 36th Conference on Decision and Control*, pages 1202–1207. IEEE, December 1997.

[19] Lei Chen, Martin J. Wainwright, Müjdat Çetin, and Alan S. Willsky. Data association based on optimization in graphical models with application to sensor networks. *Mathematical and Computer Modelling*, 43(9-10):1114–1135, May 2006.

[20] Amit S. Chhetri, Darryl Morrell, and Antonia Papandreou-Suppappola. Energy efficient target tracking in a sensor network using non-myopic sensor scheduling. In *Proc. Eighth International Conference of Information Fusion*, July 2005.

[21] A.S. Chhetri, D. Morrell, and A. Papandreou-Suppappola. Scheduling multiple sensors using particle filters in target tracking. In *IEEE Workshop on Statistical Signal Processing*, pages 549–552, September/October 2003.

[22] A.S. Chhetri, D. Morrell, and A. Papandreou-Suppappola. Sensor scheduling using a 0-1 mixed integer programming framework. In *Fourth IEEE Workshop on Sensor Array and Multi-channel Processing*, 2006.

[23] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* John Wiley and Sons, New York, NY, 1991.

[24] O.E. Drummond, David A. Castañón, and M.S. Bellovin. Comparison of 2-D assignment algorithms for sparse, rectangular, floating point, cost matrices. *Journal of the SDI Panels on Tracking*, (4):81–97, December 1990.

[25] Emre Ertin, John W. Fisher, and Lee C. Potter. Maximum mutual information principle for dynamic sensor query problems. In *Proc IPSN 2003*, pages 405–416. Springer-Verlag, April 2003.

[26] Satoru Fujishige. *Submodular functions and optimization*, volume 58 of *Annals of discrete mathematics.* Elsevier, Boston, MA, second edition, 2005.

[27] Arthur Gelb. *Applied optimal estimation.* MIT Press, Cambridge, MA, 1974.

[28] Neil Gordon, David J. Salmond, and A.F.M. Smith. Novel approach to non-linear and non-Gaussian Bayesian state estimation. *IEE Proceedings F: Radar and Signal Processing*, 140:107–113, 1993.

[29] Ying He and Edwin K. P. Chong. Sensor scheduling for target tracking: A Monte Carlo sampling approach. *Digital Signal Processing.* to appear.

[30] M.L. Hernandez, T. Kirubarajan, and Y. Bar-Shalom. Multisensor resource deployment using posterior cramer-rao bounds. *IEEE Transactions on Aerospace and Electronic Systems*, 40(2):399–416, 2004.

[31] Kenneth J. Hintz and Gregory A. McIntyre. Goal lattices for sensor management. In *Signal Processing, Sensor Fusion, and Target Recognition VIII*, volume 3720, pages 249–255. SPIE, 1999.

[32] K.J. Hintz and E.S. McVey. Multi-process constrained estimation. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(1):237–244, 1991.

[33] Stephen Howard, Sofia Suvorova, and Bill Moran. Optimal policy for scheduling of Gauss-Markov systems. In *Proceedings of the Seventh International Conference on Information Fusion*, 2004.

[34] A. T. Ihler, J. W. Fisher III, R. L. Moses, and A. S. Willsky. Nonparametric belief propagation for self-calibration in sensor networks. *IEEE Journal of Selected Areas in Communication*, 2005.

[35] A.T. Ihler, E.B. Sudderth, W.T. Freeman, and A.S. Willsky. Efficient multiscale sampling from products of Gaussian mixtures. In *Neural Information Processing Systems 17*, 2003.

[36] A.T. Ihler, J.W. Fisher III, and A.S. Willsky. Communications-constrained inference. Technical Report 2601, Massachusetts Institute of Technology Laboratory for Information and Decision Systems, 2004.

[37] Mark Jones, Shashank Mehrotra, and Jae Hong Park. Tasking distributed sensor networks. *International Journal of High Performance Computing Applications*, 16 (3):243–257, 2002.

[38] Michael I. Jordan. Graphical models. *Statistical Science*, 19(1):140–155, 2004.

[39] S.J. Julier and J.K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, March 2004.

[40] M. Kalandros and L.Y. Pao. Covariance control for multisensor systems. *IEEE Transactions on Aerospace and Electronic Systems*, 38(4):1138–1157, 2002.

[41] Keith D. Kastella. Discrimination gain to optimize detection and classification. *SPIE Signal and Data Processing of Small Targets*, 2561(1):66–70, 1995.

[42] D.J. Kershaw and R.J. Evans. Optimal waveform selection for tracking systems. *IEEE Transactions on Information Theory*, 40(5):1536–1550, September 1994.

[43] D.J. Kershaw and R.J. Evans. Waveform selective probabilistic data association. *IEEE Transactions on Aerospace and Electronic Systems*, 33(4):1180–1188, October 1997.

[44] Mark P. Kolba, Peter A. Torrione, and Leslie M. Collins. Information-based sensor management for landmine detection using multimodal sensors. In *Detection and Remediation Technologies for Mines and Minelike Targets X*, volume 5794, pages 1098–1107. SPIE, 2005.

[45] J.H. Kotecha and P.M. Djuric. Gaussian particle filtering. *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, 51(10):2592–2601, October 2003.

[46] Andreas Krause and Carlos Guestrin. Near-optimal nonmyopic value of information in graphical models. In *UAI 2005*, July 2005.

[47] Andreas Krause, Carlos Guestrin, and Ajit Paul Singh. Near-optimal sensor placements in Gaussian processes. In *International Conference on Machine Learning*, August 2005.

[48] Andreas Krause, Carlos Guestrin, Anupam Gupta, and John Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Fifth International Conference on Information Processing in Sensor Networks*, April 2006.

[49] Chris Kreucher, Keith Kastella, and Alfred O. Hero III. Information-based sensor management for multitarget tracking. In *SPIE Signal and Data Processing of Small Targets*, volume 5204, pages 480–489. The International Society for Optical Engineering, 2003.

[50] Chris Kreucher, Alfred O. Hero III, and Keith Kastella. A comparison of task driven and information driven sensor management for target tracking. In *IEEE Conference on Decision and Control*, December 2005.

[51] Chris Kreucher, Keith Kastella, and Alfred O. Hero III. Sensor management using an active sensing approach. *Signal Processing*, 85(3):607–624, March 2005.

[52] Chris M. Kreucher, Keith Kastella, and Alfred O. Hero III. A bayesian method for integrated multitarget tracking and sensor management. In *International Conference on Information Fusion*, volume 1, pages 704–711, 2003.

[53] Chris M. Kreucher, Alfred O. Hero III, Keith Kastella, and Daniel Chang. Efficient methods of non-myopic sensor management for multitarget tracking. In *43rd IEEE Conference on Decision and Control*, December 2004.

[54] Christopher M. Kreucher, Alfred O. Hero III, Keith D. Kastella, and Ben Shapo. Information-based sensor management for simultaneous multitarget tracking and identification. In *Proceedings of The Thirteenth Annual Conference on Adaptive Sensor Array Processing (ASAP)*, June 2005.

[55] V. Krishnamurthy. Algorithms for optimal scheduling and management of hidden Markov model sensors. *Signal Processing, IEEE Transactions on*, 50(6):1382–1397, 2002.

[56] V. Krishnamurthy and R.J. Evans. Hidden Markov model multiarm bandits: a methodology for beam scheduling in multitarget tracking. *Signal Processing, IEEE Transactions on*, 49(12):2893–2908, December 2001.

[57] V. Krishnamurthy and R.J. Evans. Correction to 'Hidden Markov model multiarm bandits: a methodology for beam scheduling in multitarget tracking'. *Signal Processing, IEEE Transactions on*, 51(6):1662–1663, June 2003.

[58] Thomas Kurien. Issues in the design of practical multitarget tracking algorithms. In *Multitarget-Multisensor Tracking: Advanced Applications*, pages 43–83, Norwood, MA, 1990. Artech-House.

[59] B. La Scala, M. Rezaeian, and B. Moran. Optimal adaptive waveform selection for target tracking. In *Proceedings of the Eighth International Conference on Information Fusion*, volume 1, pages 552–557, 2005.

[60] B.F. La Scala, W. Moran, and R.J. Evans. Optimal adaptive waveform selection for target detection. In *Proceedings of the International Radar Conference*, pages 492–496, September 2003.

[61] II Lewis, P. The characteristic selection problem in recognition systems. *IEEE Transactions on Information Theory*, 8(2):171–178, February 1962. ISSN 0018-9448.

[62] Dan Li, Kerry D. Wong, Yu Hen Hu, and Akbar M. Sayeed. Detection, classification, and tracking of targets. *IEEE Signal Processing Magazine*, 19(2):17–29, March 2002.

[63] Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Efficient dynamic-programming updates in partially observable Markov decision processes. Technical Report CS-95-19, Brown University, 1995.

[64] Juan Liu, James Reich, and Feng Zhao. Collaborative in-network processing for target tracking. *EURASIP Journal on Applied Signal Processing*, (4):378–391, 2003.

[65] A. Logothetis and A. Isaksson. On sensor scheduling via information theoretic criteria. In *Proceedings of the American Control Conference*, volume 4, pages 2402–2406, San Diego, CA, June 1999.

[66] Ronald P. S. Mahler. Global posterior densities for sensor management. In *Acquisition, Tracking, and Pointing XII*, volume 3365, pages 252–263. SPIE, 1998.

[67] Peter S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 1. Navtech, Arlington, VA, 1994.

[68] Peter S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 2. Navtech, Arlington, VA, 1994.

[69] Gregory A. McIntyre and Kenneth J. Hintz. Information theoretic approach to sensor scheduling. In *Signal Processing, Sensor Fusion, and Target Recognition V*, volume 2755, pages 304–312. SPIE, 1996.

[70] Bill Moran, Sofia Suvorova, and Stephen Howard. *Advances in Sensing with Security Applications*, chapter Sensor management for radar: a tutorial. Springer-Verlag, 2006.

[71] S. Mori, Chee-Yee Chong, E. Tse, and R. Wishner. Tracking and classifying multiple targets without a priori identification. *IEEE Transactions on Automatic Control*, 31(5):401–409, May 1986. ISSN 0018-9286.

[72] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, March 1957.

[73] Kevin P. Murphy. *Dynamic Bayesian networks: representation, inference and learning*. PhD thesis, University of California, Berkeley, 2002.

[74] A. Nedich, M.K. Schneider, and R.B. Washburn. Farsighted sensor management strategies for move/stop tracking. In *Proceedings of the Eighth International Conference on Information Fusion*, volume 1, pages 566–573, 2005.

[75] George L. Nemhauser and Laurence A. Wolsey. *Integer and combinatorial optimization*. Wiley, 1988.

[76] G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functions–I. *Mathematical Programming*, 14(1): 265–294, December 1978.

[77] G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functions–II. In M.L. Balinski and A.J. Hoffman,

editors, *Polyhedral combinatorics*, volume 8 of *Mathematical programming study*, pages 73–87. Elsevier, 1978.

[78] C.H. Papadimitrou and John N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, August 1987.

[79] David C. Parkes and Lyle H. Ungar. Iterative combinatorial auctions: Theory and practice. In *Proc 17th National Conference on Artificial Intelligence (AAAI)*, pages 74–81, 2000.

[80] Kris Pister. Smart dust (keynote address). In *IPSN '03*, April 2003.

[81] G.J. Pottie and W.J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, May 2000.

[82] Donald B. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, AC-24(6):843–854, December 1979.

[83] Branko Ristic, Sanjeev Arulampalam, and Neil Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications.* Artech House, 2004.

[84] Louis L. Scharf. *Statistical Signal Processing: Detection, Estimation and Time Series Analysis.* Addison-Wesley, Reading, MA, 1991.

[85] M.K. Schneider, G.L. Mealy, and F.M. Pait. Closing the loop in sensor fusion systems: stochastic dynamic programming approaches. In *Proceedings of the American Control Conference*, volume 5, pages 4752–4757, 2004.

[86] Sumeetpal Singh, Ba-Ngu Vo, Robin J. Evans, and Arnaud Doucet. Variance reduction for monte carlo implementation of adaptive sensor management. In *Proc. Seventh International Conference of Information Fusion*, pages 901–908, 2004.

[87] Richard D. Smallwood and Edward J. Sondik. The optimal control of partially observable Markov decision processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973.

[88] E. Sudderth, A.T. Ihler, W. Freeman, and A.S. Willsky. Nonparametric belief propagation. Technical Report LIDS-TR-2551, Massachusetts Institute of Technology, 2002.

[89] E.B. Sudderth, A.T. Ihler, W.T. Freeman, and A.S. Willsky. Nonparametric belief propagation. In *Computer Vision and Pattern Recognition*, 2003.

[90] P. Tichavsky, C.H. Muravchik, and A. Nehorai. Posterior Cramér-Rao bounds for discrete-time nonlinear filtering. *IEEE Transactions on Signal Processing*, 46(5): 1386–1396, 1998.

[91] Harry L. Van Trees. *Detection, Estimation, and Modulation Theory.* Wiley-Interscience, 2001.

[92] Martin J. Wainwright. *Stochastic Processes on Graphs: Geometric and Variational Approaches.* PhD thesis, Massachusetts Institute of Technology, 2002.

[93] R.B. Washburn, M.K. Schneider, and J.J. Fox. Stochastic dynamic programming based approaches to sensor resource management. In *Proceedings of the Fifth International Conference on Information Fusion*, volume 1, pages 608–615, 2002.

[94] Kirk A. Yost and Alan R. Washburn. The LP/POMDP marriage: Optimization with imperfect information. *Naval Research Logistics*, 47(8):607–619, October 2000.

[95] Feng Zhao, Jaewon Shin, and James Reich. Information-driven dynamic sensor collaboration. *IEEE Signal Processing Magazine*, 19(2):61–72, March 2002.

[96] J.H. Zwaga and H. Driessen. Tracking performance constrained MFR parameter control: applying constraints on prediction accuracy. In *International Conference on Information Fusion*, volume 1, July 2005.

[97] J.H. Zwaga, Y. Boers, and H. Driessen. On tracking performance constrained MFR parameter control. In *International Conference on Information Fusion*, volume 1, pages 712–718, 2003.