Models for the Management of Satellite-Based Sensors

by

James Thomas Walton

S.B., Massachusetts Institute of Technology, 1977
S.M., Massachusetts Institute of Technology, 1977
M.B.A., University of California, Berkeley, 1985

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1993

Author .....
Sloan School of Management
October 20, 1992

Certified by .....
James B. Orlin
Professor
Thesis Supervisor

Accepted by .....
James B. Orlin
Chairman, Ph.D. Program Committee

Models for the Management of Satellite-Based Sensors

by

James Thomas Walton

Submitted to the Sloan School of Management
on October 20, 1992, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

In this thesis, we develop and evaluate several models for the management of satellite-based sensors. Motivating applications include both strategic surveillance and environmental sensing. In the former case, the focus is on footprint placement for the coverage of point targets, where an individual sensor is presumed capable of including several targets in a single scan. To capture the notion of coverage, we use a variant of the maximum coverage location model in which we allow sensor footprints to vary between both targeted locations and sensors. Solution algorithms are developed and computationally tested against representative problem instances. The approaches considered include two marginal return algorithms, a Lagrangian relaxation with subgradient optimization, and decompositon. Finally, it is shown how the coverage model can be extended to a dynamic scenario through the use of a rolling horizon.

In the case of environmental sensing, the issue is that of scan sequencing. Here the objective is to first determine how to cover or partition a targeted region with scans, and then to optimally sequence them so as to complete the scans as quickly as possible. The scan sequencing portion of this problem can be posed as a structured version of a well known combinatorial paradigm: the Wandering Salesman Problem (WSP). For the most general regions to be imaged, scan sequencing is shown to be $\mathcal{NP}$-complete. Optimal strategies are developed for regions whose shape is subject to certain restrictions, while heuristics are specified for less constrained areas. We consider a specific sensor proposed for use in the Earth Observing System (Eos), the high resolution imaging spectrometer (HIRIS). Problems representative of the different science objectives for which HIRIS is intended are used to test the heuristics developed. Classes of strategies that are wider than those that technical limitations impose on HIRIS are evaluated. Techniques for constructing feasible shortest scan paths include nearest neighbor and multiple fragment heuristics followed by local

optimization. Lower bounds are provided by a tree relaxation with a Lagrangian objective function.

Scan sequencing can be regarded as a subproblem, the solution of which is then fed to a higher level management problem: project selection and scheduling. As a direction for further research, we show how previous work in space mission scheduling can be extended to encompass the selection and scheduling of imaging projects for a sensor such as HIRIS. Key constraining resources include time, data bandwidth, power, and thermal dissipation.

Thesis Supervisor: James B. Orlin
Title: Professor

Acknowledgements

I would like to thank my thesis supervisor, Professor James Orlin. His guidance and support played a major role in the development of the research presented in this thesis. He has also had a direct influence on my professional development. In addition, I would like to thank the other members of my thesis committee, Professor Thomas Magnanti and Professor Jeremy Shapiro, for their helpful comments and advice.

Throughout my years at MIT I have had many friends from the Sloan School, the Operations Research Center, the Laboratory for Information and Decision Systems, and the Dupont Gymnasium who have made my stay enjoyable. They are almost too numerous to mention, but include Arnout Eikeboom, David Gebala, Steve Gilbert, Ram Gopalan, Patrick Hosein, Prakash Mirchandani, Suguna Pappu, Mike Peterson, Trilochan Sastry, and Robert Zak. I also wish to thank the other faculty and staff of the Management Sciences Area at Sloan, to include Professor Robert Freund, Dain Waters, and Joan Wingo.

Lastly, I would like to thank my parents and my sister, Chris, for their support and encouragement during my years at MIT.

This research was conducted at the MIT Alfred P. Sloan School of Management with partial support provided by the Hughes Aircraft Company under a Howard Hughes Doctoral Fellowship.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

While single satellites were once sufficient for a variety of objectives, many missions today call for multiple satellites working together—even though this presents substantive tasking issues. Examples include communications applications such as Motorola's proposed network of 66 satellites for a worldwide cellular phone system, the Global Positioning System (GPS), the Earth Observing System (Eos), and space-based surveillance and tracking systems. Multiple satellites are necessary to achieve either frequent or continuous observation of specified ground locations, especially if low-earth orbits are required to provide sufficient resolution for sensors or a short range for low power transmitters. Some missions, such as either GPS or target tracking, depend upon more than single-satellite viewing of a site: tracking may need up to three satellites and GPS requires four. The Eos satellite constellation is envisioned to have four levels of simultaneity in its observations, ranging from measurements that must be made at the same time to those needed within every 1 to 3 days of one another. The principal applications that we model are surveillance and environmental observation.

The number of satellites necessary to provide continuous coverage of all ground

sites is typically prohibitively high. With fewer satellites than this available, real-time decisions must be made as to which regions will be covered, and when. In the case of communications satellites, a schedule must be determined that establishes when contact is made with specific ground stations. An algorithm that does so must be sufficiently responsive to allow for the failure of either a satellite or ground station, as well as the need for unplanned contacts. Similarly, a surveillance satellite must be oriented so as to capture as many of the objects that it is desired be covered, but must have the flexibility to handle changing priorities among ground sites. And while most Eos instruments will have planned observation strategies, the high resolution optical surface imagers will be programmable so that targets of opportunity (volcanos, fires) can be scanned.

The problems of greatest interest are inherently dynamic and real time—a sequence of assignments must be scheduled for each satellite as they move in their orbits. A low-earth orbit satellite can circle the earth (actually the earth spins beneath it) in approximately 90 minutes. Due to this movement, there will be restrictions on the surface sites that can be selected for a particular satellite as a function of time since the sensor-target geometry will change. In addition, there is no longer a single, unambiguous objective to optimize when considering the dynamic satellite assignment problem. Besides selecting scans so as to maximize coverage, another objective is to sequence them in an order that permits efficient repositioning of the sensors between scans. This avoids the need for erratic sensor movement and also allows for the scheduling of as many scans as possible within a given time span. In most scenarios, the time since each object was last visited is relevant—the less recently observed, the more desirable it is to view an object again soon.

Furthermore, solving a multiperiod decision problem requires estimates of target values for several periods ahead. Since forecasts for periods farther into the future are likely to be less certain than those made now, a common practice for solving such problems is to use a "rolling horizon" [23], [69]. This entails first solving a finite

horizon multiperiod problem and then implementing only the first period's decisions. One period later, the multiperiod problem is updated as better information becomes available, and then the procedure is repeated. For a satellite, a time period might be no more than a few minutes. This approach is well suited to re-tasking surveillance or observation satellites in future periods when significant events are observed in the current period.

There are other tasking issues of concern in the operation of satellite based sensors. For an application such as environmental observation, the focus is less on footprint placement for the the coverage of point targets than on regional mapping or the implementation of a sampling strategy. The selection of scans to be made, along with the order in which they are sequenced, must be determined. Remote sensing satellites, which have been circling the earth since Landsat was launched in 1972, routinely pose this problem. Remote sensing detects the reflected electromagnetic energy from an object, the pattern of which produces a unique signature. This can indicate an object's density, surface texture, moisture, and other physical and chemical properties. Virtually every quarter-acre of the earth's surface has been imaged by Landsat sensors in the last 20 years, with over 2.5 million pictures. The current generation of Landsats, 4 and 5, are in near-polar orbits with a 16 day repeat cycle. They are equipped with two sensors: a multispectral scanner and a thematic mapper.

The thematic mapper records an image by sweeping a mirror from side to side seven times a second. Light is deflected into detectors, scanning the scene below in a series of parallel swaths. The mirror itself is pointable to different regions of the earth. Seven separate spectral bands are viewed simultaneously, covering the visible, near infrared, shortwave infrared, and thermal infrared bands of the spectrum. The first band is used for coastal water mapping, differentiating between soil and vegetation, and identifying trees. Band 2 detects green reflectance by healthy vegetation. Infrared sensing in band 3 detects the chlorophyll absorption of different plant species. Band 4 is near-infrared and is used for biomass surveys and water body characterization.

Bands 5 and 7 are shortwave-infrared, with band 5 being used to measure vegetation moisture and to separate snow from clouds. Band 7 has the best penetration of haze. Band 6 is far-infrared, or thermal, and is used to distinguish between different types of vegetation, measure vegetation heat stress, and map coastal boundaries. The measurements taken from a particular scene can be used to identify and characterize water pollution, crops, forest, landforms, cities, and patterns of land use.

About 273 million pixels are contained in a 115-mile by 110-mile thematic mapper scene. Its resolution is 30 yards; an improved thematic mapper scheduled to be launched on Landsat 6 later this year will have an additional black and white band with about 15 yard resolution. While the Landsat thematic mapper currently has the best combination of spectral and spatial resolution with wide area coverage, an even more sophisticated imaging spectrometer with 192 spectral bands is envisioned for Eos; we describe this in detail as an application of our work. As is implied by Landsat's mapping characteristics, both it and the proposed Eos sensor pose substantive problems regarding how to select and sequence scans so as to accomplish any of a variety of prospective science objectives.

The solution to the scan sequencing problem is useful in a broader context: the selection and scheduling of which imaging projects to perform with a specific instrument, and when. Development of observation schedules is typically a large and complex task. From all the requests submitted in a period of time, some subset of them must be selected and scheduled. Decisions are subject to many constraints, to include orbit characteristics, power and thermal balance requirements for the entire platform, instrument capabilities, viewing conditions, guidance requirements, overall resource allocation objectives such as data bandwidth, and image specific restrictions and preferences. Constraints can emanate from the activities that are necessary to physically perform an observation, such as moving the sensor so that it points at the correct spot, as well from mission objectives. An example of the latter are scheduling constraints imposed by science goals. Numerous relationships among observa-

tions may be specified, including partial orderings over sets of observations, temporal separation constraints between ordered observations, temporal grouping constraints over unordered observations, coordinated parallel observations with different viewing instruments, same sensor orientation constraints for repeat observations, and conditional execution of dependent observations [82]. And within a given project, scans can have differing priorities.

It will often not be possible to satisfy all problem constraints. In such situations, it may be necessary to selectively relax requirements that are not absolute. In general, the scheduling problem is one of seeking to maximize the amount of "science" performed, subject to insuring feasibility with respect to a complex set of constraints involved with instrument operation, image execution, and overall resource allocation objectives.

Specific instrument scheduling is similar to the larger domain of space mission scheduling [4], [17], [18], [38], [62] in which the same basic decisions of selection and scheduling of science projects must be made. The primary means currently used by the National Aeronautics and Space Administration (NASA) is a knowledge-based system called *Plan-It*. The claim is made that classical optimization techniques are not able to handle scheduling in this large, complex domain [18]. An alternative approach [57] is to consider only a subset of the constraints involved, which would certainly seem suitable for the smaller scale problem of scheduling a single instrument. A model that can quickly generate high quality solutions with respect to changing resource constraints could be quite useful as part of a comprehensive scheduling system.

1.2 Applications

Applications for satellite-based sensors in surveillance and environmental sensing motivated our study of satellite coverage and scheduling problems. Before mathemati-

cally formulating models for these problems, however, we describe these applications in some detail, as well as the additional example of communications.

1.2.1 Strategic Surveillance

As part of the Strategic Defense Initiative, "Brilliant Eyes" is proposed as a constellation of some 40-60 satellites that would track nuclear warheads from intercontinental ballistic missiles in midcourse. Current proposals favor large numbers of small-aperture satellites, since proliferated systems of spacecraft enhance their ability to survive an adversary's antisatellite attack. Brilliant Eyes would use sensors to track objects, discriminate reentry vehicles from decoys, and cue interceptors. The sensor satellites themselves would typically have to be cued to look for targets by other warning satellites, but could also be told to look in an area the size of Iraq. At a minimum, several tens of satellites would be needed to provide continuous coverage of priority launch areas, which would also ensure adequate coverage around the globe for submarine-launched missiles. Redundancy would be required for stereo viewing of targets as well as survivability. Such a satellite sensor constellation is also envisioned to play a role in theater missile defense, in addition to strategic defense.

Each satellite would be about the size of a desk and weigh less than half a ton. The most recent concept calls for two telescopes for target discrimination. An on-axis telescope would have sensors in the visible and mid-wavelength infrared range for the acquisition and tracking of warm objects. Long-wavelength infrared sensors would be used for tracking somewhat warm and cold objects. Such sensors could not detect targets by looking straight down against the relatively warm earth background. Instead, they would only look above the horizon in a pattern that provides the necessary cold space background for infrared detectors. A laser radar would probably be included to permit precise range determination, allowing a single satellite to determine a target's state vector. The exact selection of sensors will depend on both engineering constraints and the missions for which the constellation is to be intended. While the

orbits of the satellites have yet to be determined, they will likely fall within the range of 700-1600 km.

The assignment of satellite-based sensors for strategic surveillance entails considering the coverage problems already described. All potential launch sites can not be covered with high resolution sensors at a given instant, and thus the best sensor footprint placement must be determined. A sequence of assignments that creates a smooth scan sequence and captures the most high-priority locations is sought. The scheduling process must account for the recency with which an object was last viewed, as well as be responsive to adaptive rescheduling if the cueing satellites identify a possible missile launch.

1.2.2 Earth Observing System (Eos)

The traditional approach to earth science has been to study individual components—the atmosphere, oceans and inland bodies of water, alpine snow and ice, soils, vegetation, and geology. Models of these separate components have not been designed to interact with one another, but rather were developed independently and focused only on their specific discipline. Some models, such as ecological and hydrological models, require information and provide results only over very small areas, while global climate models characterize the entire planet. Thus the outputs of the former cannot easily be incorporated into the coarse spatial grid of the latter.

The current trend is to consider the various interactions between different components at all temporal and spatial scales. The satellite-based Earth Observing System (Eos) is intended to provide the sensing capability necessary to collect data in support of such an integrated model of the earth's environment. It will contain instruments that sample various phenomena throughout the electromagnetic spectrum. One objective is to understand the scale dependencies that are involved among the different terrestrial components. This has the potential to allow the outputs of models for small scale events to be used as inputs for larger scale models, which can enhance

our ability to predict environmental change.

An instrument that will make measurements at the finest temporal and spatial scales is the high resolution imaging spectrometer (HIRIS). HIRIS will operate at an intermediate level in a multistage sampling program between *in situ* human-acquired field data and global mapping instruments. The sensor employs a pointable imaging spectrometer with area array detectors that can sample any point on the surface of the earth at least once every 2 days. The instrument obtains images in the 0.4- to 2.45-μm wavelength region in 192 spectral bands with 10-nm spectral sampling and a 30 meter pixel size. The swath width is 24 km (800 pixels). It is a targeting (rather than continuous acquisition) instrument that images areas of interest by pointing line-of-sight from +60° to -30° along-track (positive is the velocity direction) and ±45° cross-track. Cross-track pointing permits frequent repeat sampling. Along-track pointing will be used to estimate the reflectance distribution of surfaces, to remove atmospheric attenuation, and to implement image-motion compensation that increase the signal-to-noise ratio for dark targets. Contiguous spectral coverage will be used to perform spectral analyses. Relevant technical parameters for HIRIS are discussed more fully in section 3.1 of Chapter 3.

The planned spectral resolution of HIRIS (10-nm) is sufficient to reproduce essentially all diagnostic features in the spectra of solids, and thus is a major step toward direct identification of minerals and soils. It will also allow for the examination of suspended sediments and phytoplankton in coastal and inland waters; estimation of the grain size of snow and its contamination by absorbing impurities; and study of biochemical processes in vegetation canopies, such as chlorophyll concentration, leaf area index, leaf tissue water content, canopy composition and geometry, vegetation indices, and the distribution of dead and green biomass [36]. With present broadband sensors, none of these attributes have been measured.

High spatial resolution for HIRIS is important because various processes that occur on a global scale have surface features with dimensions on the order of tens

of meters. Examples include upwelling and mixing along fronts in coastal waters; damage to vegetation such as tree death, windfall, and other disturbances which can occur in patches; forest clearing; land-use change; mineral outcrops that may only have a few square meters of surface exposure; and alpine snow and ice [53]. Greater detail regarding the science objectives and uses of HIRIS is provided in Appendix A.

HIRIS is tentative for the Eos-A2 and -A3 platforms, which will be launched around the turn of the century. If included, they will obtain at least 1 decade of overlapping, calibrated observations. HIRIS will allow scientists to observe and analyze specific terrestrial phenomena in a manner that is not currently possible with existing ground-based or spaceborne instruments. Imaging requests from prospective users are likely to be numerous, and efficient management of the instrument so as to maximize scientific use is thus a critical concern. Since adaptive scheduling of HIRIS will be necessary to image targets of opportunity such as volcanos or fires, the ability to quickly generate high quality solutions with the respect to changing resource constraints for the project scheduling problem is potentially quite useful.

Because the Eos platforms will be in low-earth, sun-synchronous orbits (705 km), the various projects can only be performed during specific time windows. Depending upon the length of the planning horizon for which a schedule is being constructed, there may be multiple opportunities during which specific imaging requests can be accomplished. Different time windows for the same project may have differing levels of desirability, to reflect preferences between viewing angles. Viewing vegetation from an off-nadir angle, for instance, can provide a better estimate of the total albedo of the canopy, since the angular distribution of reflectance from vegetation is sensitive to leaf density, leaf orientation, and the distance between plants [51]. It may also be the case that projects will consist of linked or repeat observations. The frequency requirements of coverage flow directly from the nature of the science area to be studied. Water and snow/ice studies need frequent measurements, while vegetation studies require less frequent ones. Geological studies are a distant third, but must acquire data under

optimal lighting and visibility conditions and when the area of interest is neither covered by vegetation nor obscured by clouds.

There are constraints for other shared resources relevant to the operation of HIRIS. The sensor is capable of producing data at a prodigious rate—on the order of terabytes a day. This vastly exceeds the capacity of the Eos Data Information System, which is the system that will manage the collection, processing, and communication of data produced by the entire suite of instruments. HIRIS tentatively has a long term data rate "budget" of 10 mbps, as well as a short term data rate limit of 100 mbps that can be passed across the platform interface from the instrument to the data information system. Since the data that many measurements produce will easily exceed 100 mbps, a tape recorder will be used to buffer data. The buffer size provides a storage budget that must be allocated between projects and over time (stored data is downlinked to the ground, freeing up storage capacity). Power consumption and thermal dissipation are other operating constraints.

In addition to macro-level project scheduling, in which projects from various areas of earth science are to be selected and scheduled, there are issues related to the sequencing of scans within an individual imaging project. The processing time for a given project is taken as an input to the larger scale scheduling problem. Many projects, however, are likely to require far more than a single 24 km by 30 m scan. Thus the following problem is posed:

> Given a region that is to be imaged, what selection of scans and in what sequence will minimize the amount of time necessary to cover the desired area?

Note that in its most general form, the region to be imaged for a particular project need be neither contiguous nor convex. This problem really has two coupled parts: determining which scans produce the optimal partition or cover of the targeted area, and then sequencing them. The two parts are coupled in that the scan selection that ultimately minimizes the processing time might not be a minimum cardinality

partition or cover. If it is assumed that the optimal scan cover is already known, then the remaining problem of sequencing the scans has the flavor of a geometric wandering salesman problem, but with an underlying scan structure.

1.2.3 Satellite-Linked Mobile Phones

Several competing proposals for satellite-linked mobile phone systems are currently under consideration by a combination of investors and policy makers. All of them envision a system that would let users make calls to and from places that are not currently served by either traditional or cellular telephone services, such as the forests of Canada or the outback of Australia. In addition, the satellites could also serve as relays for computers or paging devices. The proposed designs and their costs vary significantly, depending on how ambitious they are. A fundamental question yet to be resolved, is exactly what service should be provided?

One version, Motorola's Iridium plan, calls for a constellation of 66 (recently scaled down from 77) satellites in low polar orbit, each weighing about 680 kilograms and carrying complex switching equipment. They would pick up mobile phone calls from any location on Earth and be capable of handling 200 simultaneous calls. The system would be able to relay calls on its own, from satellite to satellite, bypassing the present land-based long-distance carriers. These features would require the satellite array to provide continuous coverage of the entire earth's surface.

Several rivals to Motorola's plan propose systems employing between 12 and 48 satellites that would generally be lighter, cheaper, and less capable. In particular, there would be no provision for routing calls. Instead, the satellites' only role would be to link mobile phones to regional ground stations. All switching and routing between ground stations would be done over lines provided by the current long distance carriers. Far fewer satellites are needed for such an architecture, since unpopulated land areas and the oceans would not be covered. In addition, their design can be much simpler since they do no switching.

All of the competing systems would use low-earth orbit satellites. Unlike previous generations of geostationary communications satellites that orbit at 22,300 miles above the earth, these would circle the earth at distances of a few hundred to several thousand miles. Since they move across the horizon, the only to way to provide adequate coverage is to launch a dozen or more satellites, and then optimize the assignment of antennas to gateways on the ground. Such a system poses challenging scheduling issues related to insuring acceptable antenna coverage. The advantage of low-earth orbit satellites for communications, however, is that they need much less power, are cheaper to launch, and are close enough to acquire signals from very weak transmitters.

Regardless of the proposal, there are profound issues concerning the coverage of ground sites that is to be provided by the antennas that will be used, some of which are directional. Moreover, this is not a problem that needs be solved only once. Failures of various system components (transmitters, antennas, receivers, power units) will result in the need to re-solve a dynamically evolving coverage problem, and possibly to do so very quickly. Thus our analysis of these issues is a potential contribution to development of the technology.

1.3 Contributions of Thesis

The satellite coverage and scheduling problems considered are significant because their optimal or near optimal solution will allow for the effective utilization of costly space resources—both from economic and operational perspectives. If exercised to the extent of their capabilities, fewer such resources will ultimately need to be deployed. And once deployed, the level of service provided will be sufficient to meet most mission objectives by means of adaptive rescheduling. These perspectives correspond to optimizing strategic (economic) and tactical (operational) decisions. In this research we develop, implement and computationally test algorithms for the schedul-

ing and tasking of satellite based sensors. We are primarily interested in the physical paradigms of either surveillance or environmental observation.

We first consider a static version of satellite coverage. In a fashion similar to problems in the location theory literature, we formulate the static version as a maximal covering location problem model in which the objective is to cover as many of a discrete set of objects as is possible with a given number of satellites. In comparison to traditional specifications, ours does not assume that different sensors (satellites) will cover an identical region when oriented toward the same area; i.e., we allow for different "footprints." Various algorithms are developed and several of them implemented and tested in C. Implementation and testing entails generating data for realistic problem instances with which to test the various algorithms specified. For the static problem we produce surface objects whose values and sensors whose positions are both time-invariant. We then show how the static model can be extended to consider a dynamic version of the coverage problem, in which the additional issues of time windows, construction of smooth scan sequences, and time-dependence of target values are addressed.

We also study sequencing and scheduling problems of interest in the management of space-based sensors. First we examine a micro-level sequencing problem, in which we evaluate the selection of scans and their subsequent ordering for actual imaging. The basic problem has the flavor of a wandering salesman (or shortest Hamiltonian path) problem, in which a salesman seeks to minimize the distance necessary to visit each in a set of cities exactly once, starting and stopping at the cities of his choosing. In the problem of interest to us, the "cities", or scan points, have an underlying scan structure. We establish upper and lower bounds on the number of scans that will need to be made to cover a region, as well as for the amount of time that it will take to perform the scans. We consider a special case of the scan sequencing problem for which we can specify a polynomial algorithm that will provide an optimal solution. The general optimal scan sequencing problem (OSSP) is shown to be $\mathcal{NP}$-complete,

and thus we consider practical heuristics. A number of heuristics that exploit the structure of the OSSP are specified and computationally tested.

The scan sequencing problems are useful in a broader context, that of project selection and scheduling. The outputs of numerous different OSSP problems are input to this higher level problem. Whatever the scheduling horizon may be, there will far more projects to be performed than there are resources to conduct them. The selection of projects is based on their relative importance, as well as the availability of a number of constraining resources such as time and data bandwidth. Rather than constructing a large, complex model that attempts to solve all scheduling problems at one time, a potential approach is a relatively simple formulation that is capable of quickly generating very good solutions to the base resource allocation problem. It could then be adapted to other emerging requirements, or re-solved altogether if circumstances should change sufficiently. As a future research direction, we discuss this wider setting in which the scan sequencing algorithms would be used as subroutines.

Chapter 2

Sensor Coverage Models

2.1 Introduction

In this chapter we present coverage models for satellite based sensors. Although the applications of greatest interest include environmental observation as well as surveillance, it is the latter for which the fit is best for the models we will be describing. The sensors used in a surveillance context would likely be sensitive to mid- and long-wavelength infrared, in addition to visible wavelengths, and thus a single satellite may well have several sensors. The underlying optimization problem is relatively simple to state: we seek the best placement of sensor footprints for the coverage of point targets, where different sensors may have different footprints based upon satellite-target geometry. The "footprint" of a sensor is the pattern that its instantaneous coverage region takes, where "coverage" refers to the area or volume within which the sensor will respond to a stimulus.

An example of such a sensor is a camera mounted on a satellite. The stimulus to which it responds can be visible, infrared, or other wavelengths of electromagnetic radiation, depending on the sensitivity of the particular detector. If it is visible wavelengths, then the total terrestrial region within which the sensor can acquire targets is limited to those surface areas with which straight-line line-of-sight can be

established. The camera's footprint is determined by its technical characteristics, such as the focal length of the lens, its aperture setting, the size of the focal plane, and other attributes. The image recorded by the sensor is of a specified resolution, which is the size of the smallest area for which the sensor can measure a response. Each of these minimally distinguishable elements of the coverage region is called a pixel, or picture element.

It is assumed that a sensor's footprint, which may consist of many pixels, is capable of covering multiple objects. The size of an individual pixel will increase as a sensor is pointed off-nadir (nadir being the point on earth closest to the satellite—its projection to the earth below). Although this implies that the sensor will be covering more area, that is not always desirable. In imaging spectrometry, for example, the spectral response within a pixel is a mixture of signals from a diversity of surfaces. Thus increased pixel size serves to decrease spatial resolution, degrading the ability of scientists to investigate reflectance characteristics with a precision that allows recognition of features and analysis of the mixing process. Similarly in surveillance, off-nadir imagery may be accompanied by a loss in resolution.

A static (snapshot) version of such a coverage problem can be posed as locating the sensors under deterministic conditions, i.e., when there are specific sets of sensors and of objects to be scanned or "covered." Figure 2-1 shows the basic model. The three different shapes (squares, triangles, and circles) represent the possible footprint placements of three different sensors. We do not mean to imply that this these patterns are the *actual* shapes that a sensor footprint might take on the surface, which are likely to be far more complex, but simply seek to distinguish between sensors. As shown, the coverage capabilities of a particular sensor will vary depending on where it is pointed. And different sensors, even if oriented toward the same location, may have different footprints. In Figure 2-1 we seek the circle, triangle, and square that together cover the most valuable collection of objects; the relative importance of different objects is reflected by associating a non-zero value with each one. Although

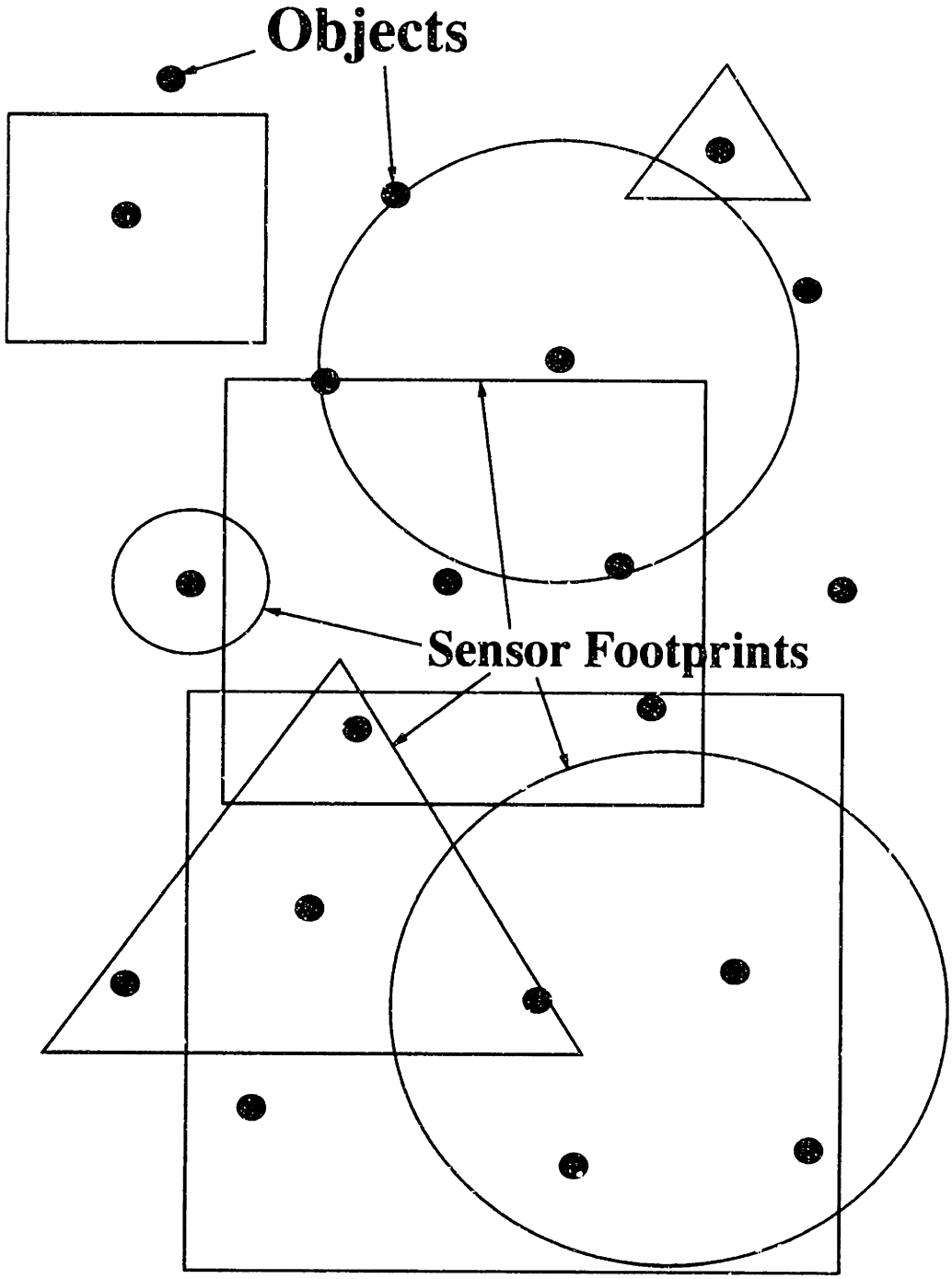

Figure 2-1: Coverage Model.

the upper left square covers only one object while the bottom square covers eight, it may well be that the single object is far more significant than the other eight. Since the small square is the only sensor shown to be capable of capturing that object, that position may represent the sensor's optimal placement at the particular time that this diagram shows the alternatives available.

There are several classic operations research problems in discrete location theory that are explicitly concerned with the notion of coverage, one of which we use as the departure point for our models. The most well known of them is the set covering problem (SCP), which can readily be formulated as a 0-1 integer program. Let A be an $m \times n$ binary incidence matrix. Each $i \in M$, where M is the set of rows, represents an element in a ground set. The columns, M_j for $j \in N$, correspond to a set of subsets defined on the ground set; that is, for $i \in M$,

$$a_{ij} = \begin{cases} 1 & \text{if } i \in M_j \\ 0 & \text{if } i \notin M_j \, . \end{cases}$$

Also define

$$x_j = \begin{cases} 1 & \text{if } j \in F \\ 0 & \text{if } j \notin F \, . \end{cases}$$

Then F is a *cover* of M if and only if $x \in \{0,1\}^n$ satisfies

$$Ax \geq 1,$$

where 1 is an m-vector all of whose components equal 1. If, instead, we have $Ax \leq 1$ or $Ax = 1$, then this problem becomes, respectively, the set packing and set partitioning problems. The optimization problem defined over the set covering constraints consists of finding a cover x whose cost cx is minimum; c is an n-vector and its components, c_j, are the costs of the M_j's.

Many practical problems have been formulated using the set covering model

[34], [40], [77], [79], [107]. A representative application is the location of an emergency services facility [93], [108], [112]. Assume we are given a set of possible sites $N = \{1, \ldots, n\}$ for the location of police stations in a county. A police station at site j costs c_j. Suppose we have a set of towns $M = \{1, \ldots, m\}$ that have to be covered. The subset of towns that can be patrolled from a station at j is M_j, which could correspond to the towns that could be reached within some time limit, say, 15 minutes. Then the optimization problem of selecting the minimum-cost set of police stations that is capable of reaching every community within 15 minutes is a set covering problem. Other applications include airline flight and crew scheduling, assigning customers to vehicle routes, and assigning workers to cover shifts.

There are a number of similarities between the set-covering problem and the sensor management problem that we seek to model. They include the basic notion of spatial coverage, in which the ability of some sort of "server" to satisfy "demand" occurring at geographically distributed points is a function of distance from wherever the server is located. The potential objectives of either minimizing cost or maximizing coverage are similar also. Nonetheless, there is a significant difference between the two problems: in our case, an arbitrary number of sensors cannot be selected. We assume that the number of sensors and their locations are given. The constellation design problem [9], [35], which is not addressed here, would consider how many sensors/satellites to launch and into what orbits (e.g., orbital altitude and inclination). The goal of our modeling is to optimize the use of a specified set of resources; as observed earlier, this corresponds to tactical rather than strategic decision making.

The second location theory problem that deals with coverage, however, comes closer to capturing the physical paradigm we are examining. The maximal covering location problem (MCLP) adds an additional constraint to the SCP:

$$\sum_{j \in N} x_j = p \,,$$

where p is a limit on the number of subsets (columns in the integer program) that can be selected. The weights are now defined over the elements of the ground set (rows in the integer program), and the optimization problem is to select the set of columns that covers the maximum weight. The MCLP can also be formulated as a generalized assignment problem (GAP) [96] or as a p-median problem with a distance matrix of special structure [29]. A typical application of the maximal covering location problem is to maximize the population within a service distance S of a fixed number of facilities p, which could be on a transportation network. The population at a node is said to be covered if there is a facility located within a distance or time S of the demand node.

The difference between the traditional MCLP formulation and our model is that our "servers" (satellite-based sensors) are distinguishable. Typically, all service facilities in an MCLP are assumed to have the same service distance regardless of their location, e.g., a warehouse in Houston has the same service capabilities as does one in Cleveland. If service distances differ, it is on the basis of either the service level in a hierarchy of service categories [80], or on some attribute of the customer/demand [76]. An example of the first case is the hierarchical covering problem [80] in which there are N types of facilities that provide different levels of service. A facility of type i provides a radius r_i of coverage, and the objective is to locate n_i facilities of type j to maximize the total demand that has access to all levels of service. The coverage footprint (which is the analog of service distance) of a space based sensor, however, can vary with the satellite-target geometry. While there is no dependence on the object being imaged, different types of sensors will have different footprints, and the same sensor's footprint can change depending on the scan geometry. Thus the "servers" in this context are distinguishable. This is in contrast to the MCLP and set covering models that will be described with greater detail in the literature survey.

After considering the static version of such a coverage model, we will discuss the salient issues that one must address in a dynamic version, and indicate how the static

problem could be exploited in further work in the dynamic domain.

2.2 Literature Survey

Optimization problems similar to sensor coverage have been treated in the location theory literature. The general location paradigm is one in which a number of servers (sensors in our context) serve a spatially distributed set of demands (surface objects here). The underlying structure may be discrete or continuous, possibly with some special network structure. The goal is to locate the servers so as to optimize an objective that in some fashion is spatially dependent. It is easily seen that both the objective and general character of this research domain share significant features with the sensor coverage problem.

Using location research as a starting point, then, we identify various modeling approaches and problems whose foci are germane to those of interest to us. At times, it is appropriate to consider the paradigms so identified more broadly than in just the location literature. An example is the set covering problem, which has received considerable attention elsewhere. Our intent is not to review all possible different models or to describe their analysis, but rather to furnish an overview of how relevant problems have been treated. In particular, we identify how previous work relates to our interests; to a substantial degree this was also done in section 2.1.

In [20] Brandeau and Chiu present a taxonomy to distinguish different location problems based upon their objective function, decision variables, and system parameters. Common objectives that they identify include: minimizing average travel time or distance between demands and servers; minimizing average response time (travel time plus any queue delay); minimizing a cost function of travel or response time; minimizing maximum travel time; or maximizing minimum travel time. They briefly review a representative sample of more than 50 different problem types. Examples of typical location problems are warehouse siting, firebox coverage, competitive facil-

ity location and network design. Brandeau and Chiu identify five problems that are concerned with the notion of coverage: covering, maximal covering, minimum cost partial covering, p-cover, and hierarchical covering problems.

As noted, the objective in the *covering* problem is to locate a minimum number of servers, typically on a network, so that every demand point is within a given distance of the nearest server. In [40] Elzinga and Hearn consider the minimum covering sphere problem, in which they seek the sphere of smallest radius that covers a set of points in Euclidean n-space. Moon and Chaudry [79] examine a class of network location problems with minimum or maximum separation requirements between uncapacitated facilities or between demand points and the facilities, or both. Other applications of the set covering model include the location of emergency facilities [93], [108], [112], assembly line balancing [100], and information retrieval [34]. Although the concept of coverage used in all of these cases is comparable to that for sensors (for which coverage may be *variable*), our goal is not to determine the minimum number of sensors needed, but instead to optimize the use of an existing set of resources.

The *maximal covering location problem* seeks to position a specified number of servers so as to maximize the aggregate demand that falls within a given distance of the nearest server, capturing the dimension of finite resources. Church and Revelle [28] maximize coverage (population covered) within a desired service distance S by locating a fixed number of facilities. They distinguish this objective from those of minimizing the measures of total weighted distance or time for travel to the service facilities (*p-median* problem), and the distance or time that the user most distant from a service facility would have to travel to reach that facility (*p-center* problem). The p-median and p-center problems are similar to the maximal covering location problem in that they, too, specify the number of service facilities (p) that will be opened as problem input. But instead of focussing on the service *provided* (maximizing coverage), they concentrate on the service *not* provided. Respectively, the p- center and median problems are *minimax* and *minisum* formulations. There is a

substantial literature for both of these problems [65], [66], [55], [56], [77], [107].

In the context of facility layout, Francis and White [45] discuss what they call the *partial-covering* problem for discrete plant location, which is essentially the same as a maximal covering location problem: the demand, or service population, is only *partially* covered. They review both cutting plane and heuristic solution approaches. Megiddo et al. [76] consider the maximal covering location problem from the point of view of a company which is interested in establishing new facilities on a network so as to maximize the company's "share of the market." The company gains w_i if customer i switches to one of their new facilities, and the decision to switch is based only on a customer dependent distance r_i. The problem is to locate p new facilities so as to maximize the total gain. Daskin [32] extends the maximal covering location model to account for the chance that when a demand arrives at the system it will not be covered since all facilities capable of covering the demand are engaged serving other demands. The computational performance of commercial MPS software in solving maximal covering location problems is compared to the performance of a special purpose optimal algorithm (DUALOC) [41] in solving MCLPs structured as median location problems in [113]. None of the maximal covering location problem work addresses the issue of service distance being dependent on the particular server, or on the combination of both the server and where it is located. Megiddo [76] does, as noted, consider a *customer* dependent service distance.

When the number of servers is specified and the goal is to locate the servers so that the maximum distance between any two servers is minimized, the problem becomes the *p-cover problem*; an example of which is [39]. The *hierarchical covering problem* assumes that different facilities provide different levels of service, and seeks to maximize the total demand that has access to all levels of service. Moore and Revelle [80] formulate such a problem as an integer program, and give as examples of service hierarchies an educational system with primary and secondary school levels, and health care systems. Daskin and Stern [33] use a hierarchical formulation to

extend a conventional set covering problem for locating emergency medical service vehicles. They account for interdistrict responses by finding the minimum number of vehicles needed to cover all zones while simultaneously maximizing the extent of multiple coverage of zones. The concept of service *hierarchies* could be useful in modeling satellite coverage when one considers sensors of different types or quality, and there is potentially a desire to either seek or require multiple coverage.

Mirchandani and Francis [78] recently edited a comprehensive reference text on discrete location theory. Their treatment of covering problems focusses on those with an underlying network structure, in which both servers and demands are assumed to be located on the nodes or arcs of a network. Service facilities must be established on the network, and all are assumed to be of the same type, with each having sufficient capacity to satisfy the total demand. Three types of costs associated with a given set of facility locations are considered: setup, transportation, and penalty costs, where the latter is a cost which is applied only if a demand is not served by any facility. In addition they look at budget, client, and facility constraints. Client and facility constraints include service distance restrictions. Mirchandani and Francis identify five different classes of objective functions:

1. minimize the maximum transportation cost (the *center* problem);
2. minimize the sum of the transportation costs (the *median* problem);
3. minimize the sum of the transportation costs and the setup costs (the *uncapacitated facility location* problem);
4. minimize the sum of the setup costs and penalty costs (the *covering* problem);
5. minimize the penalty costs (the *coverage* problem).

Many of these problems are solvable in polynomial time when the underlying network has a tree structure. The *general* problems, (p-center, p-median, and set covering) are all $\mathcal{NP}$-complete [49]. This highlights the most critical difference between the

particular variants of location problems that Mirchandani and Francis [78] consider and ours: there is no underlying network or tree structure to the sensor coverage problem. The ideas relating to costs and coverage are much the same, though.

The literature on the set covering problem itself extends well beyond its specific application to location analysis. Two somewhat dated surveys of prior research are Garfinkel and Nemhauser [50] and Christofides and Korman [27]. Theoretical results are presented in Balas [5] and Bellmore and Ratliff [13]. Algorithms have been developed by Lemke, Salkin, and Spielberg [70], Salkin and Koncal [99], Etcheberry [42], Balas and Ho [6], and Beasley [10]. Most of these algorithms employ Lagrangian relaxation with subgradient optimization in some fashion, and are similar to one of the approaches that we take in solving the sensor coverage problem. Recent algorithmic work, however, has focussed on characterizing the facial structure of the set covering polytope [7], [8], [30], [85], [102]. Any facets that can be identified are then used to strengthen the linear programming relaxation. There has been no examination of the maximal covering location problem's polyhedral structure; adding the complicating side constraint significantly alters it from the fundamental set covering problem.

2.3 The Static Coverage Problem

2.3.1 Problem Definition

We consider a system of sensors, referred to also as satellites, that can be oriented in different positions. Note that a single satellite may well have a collection of up to a dozen or more instruments/sensors, each of which has different scan related parameters. The scan parameters of an instrument include characteristics such as the swath dimensions for various mapping modes (local, regional, and global), spatial resolution, pointing capabilities, instantaneous-field-of-view, field-of-view, and data rate. To give an example, consider the Eos synthetic aperture radar (SAR) [83]. Geophysical products generated from the SAR image data include forest biomass and

deforestation extent; soil, vegetation, and snow moisture; sea ice type and motion; and geomorphological properties. Its swath dimension ranges from 30- to 50-km (with 20- to 30-m spatial resolution) in the local high resolution mode to 350- to 500-km (250- to 500-m spatial resolution) in the global mapping mode. Its field-of-view is a $15 - 50°$ look angle from nadir, on both sides of nadir, and its data rate is 15 mbps (average) and 180 mbps (peak).

Despite the potential multiplicity of instruments, when we speak of a "satellite," we are referring to a single discrete sensor. In addition, there is an array of distributed point objects, or demands, to be covered and a nonnegative set of values that reflect their relative importance. At any given point in time a satellite will not likely be visible to or able to establish visibility with all of the objects; however, optimizing the assignment of satellites to feasible positions can maximize the ones covered. We assume that the set of "feasible positions" to which a sensor can be assigned is different for different sensors. Since the footprints of the various sensors may differ even when oriented toward the same location, we consider a version of the maximal covering location problem that distinguishes the coverage for each sensor at its feasible locations.

The geometry of the basic model, then, is that shown in Figure 2-1. "Pointing" involves aiming the sensor at a targeted object or region, with the sensor footprint then taking a particular shape. Different sensors may have different shapes, and for a given sensor its shape may vary depending on the geometry with a particular target. We assume that each sensor will be assigned to some point. The mathematical formulation of our problem models the objective of covering as many of the ground sites or regions as possible with a limited number of sensors.

2.3.2 Mathematical Formulation

Defined discretely, a mathematical formulation of the problem is as follows:

$$\text{Maximize} \sum_{i \in M} c_i y_i \tag{2.1}$$

$$\text{subject to} \quad \sum_{k \in P} \sum_{j \in N^k} a_{ij}^k x_j^k \geq y_i \quad \forall\, i \in M \tag{2.2}$$

$$\sum_{j \in N^k} x_j^k = 1 \quad \forall\, k \in P \tag{2.3}$$

$$y_i \in \{0,1\} \quad \forall\, i \in M \tag{2.4}$$

$$x_j^k \in \{0,1\} \quad \forall\, j \in N^k,\ k \in P \tag{2.5}$$

where M = set of demand sites; $|M| = m$

N = set of server locations; $|N| = n$

N^k = set of server locations for sensor k; $|N^k| = n^k$

P = set of servers; $|P| = p$

c_i = value of demand i

$$a_{ij}^k = \begin{cases} 1 & \text{if server } k \text{ at position } j \text{ can cover demand } i \\ 0 & \text{otherwise} \end{cases}$$

$$x_j^k = \begin{cases} 1 & \text{if server } k \text{ is assigned to position } j \\ 0 & \text{otherwise} \end{cases}$$

$$y_i = \begin{cases} 1 & \text{if demand } i \text{ is covered} \\ 0 & \text{otherwise .} \end{cases}$$

A common variable substitution is to let $\bar{y}_i = 1 - y_i$. Then

$$\bar{y}_i = \begin{cases} 1 & \text{if demand } i \text{ is } \textit{not} \text{ covered} \\ 0 & \text{otherwise .} \end{cases}$$

This will produce the equivalent generalized upper bound (GUB) constrained maximal covering location problem:

$$\text{Minimize} \sum_{i \in M} c_i \bar{y}_i \tag{2.6}$$

$$\begin{aligned}
\text{subject to} \quad \sum_{k \in P} \sum_{j \in N^k} a_{ij}^k x_j^k + \bar{y}_i &\geq 1 \quad \forall\, i \in M && (2.7)\\
\sum_{j \in N^k} x_j^k &= 1 \quad \forall\, k \in P && (2.8)\\
\bar{y}_i &\in \{0,1\} \quad \forall\, i \in M && (2.9)\\
x_j^k &\in \{0,1\} \quad \forall\, j \in N^k,\ k \in P && (2.10)
\end{aligned}$$

The objective now is to minimize the uncovered demand; this is the version we will consider henceforth in the paper. Equations (2.3) and (2.8) are what are referred to as *generalized upper bound constraints*, where the (j,k) combinations for a given constraint produce disjoint subsets of $N \times P$. The classical formulation of a maximal covering location problem does not have GUB constraints.

Note that the use of the a_{ij}^k variables implies a large binary matrix that is potentially quite sparse. Another way to represent the coverage constraints (2.7) is to define:

$$\begin{aligned}
N_i^k &= \{j \mid a_{ij}^k = 1\}\\
&= \{j \mid i \in M_j^k\}\\
M_j^k &= \{i \mid a_{ij}^k = 1\}\\
&= \{i \mid j \in N_i^k\}\,.
\end{aligned}$$

N_i^k is the set of positions from which sensor k is capable of covering object i. Similarly, M_j^k is the set of objects that sensor k is capable of covering when it is pointed at position j. The two can be related to one another by observing that if a given position

is in the set of positions from which a particular sensor can observe a specific object, then that object must, in turn, be in the set of objects observable by that sensor in the given position. Thus $N_i^k = \{j \mid i \in M_j^k\}$. Analogously, if a given object is in the set of objects that are observable by a particular sensor in a specific position, then that position must be in the set of positions from which that sensor can observe the given object. Hence $M_j^k = \{i \mid j \in N_i^k\}$. We will let $m_j^k = |M_j^k|$ and $n_j^k = |N_j^k|$. This leads to the following alternative specification of constraint (2.7):

$$\sum_{k \in P} \sum_{j \in N_i^k} x_j^k + \bar{y}_i \geq 1 \quad \forall\, i \in M\,. \tag{2.11}$$

This notation is consistent with the implementations described in section 2.5.

The typical formulation of a maximal covering location problem does not discriminate between the various servers at different locations; the assignment variables are traditionally defined as:

$$x_j = \begin{cases} 1 & \text{if a server is placed at position } j \\ 0 & \text{otherwise}\,. \end{cases}$$

For the problems of interest here we do not assume that each server can be assigned to the same set of locations or that they are capable of covering the same objects if it is feasible for more than one to be pointed toward the same region. Thus the covering constraints have a different set of columns for each sensor. In addition, there is an individual assignment constraint for each sensor. Note also that the $\bar{y}_i$'s actually need not be binary but could be continuous

$$0 \leq \bar{y}_i \leq 1 \quad \forall\, i \in M$$

since in a minimization problem they will always attain their lower bound, which here is either 0 or 1.

2.3.3 Computational Complexity

The SENSOR COVERAGE problem is $\mathcal{NP}$-complete by transformation from the problem MINIMUM COVER [49, p. 222], which is stated as follows:

INSTANCE: Collection C of subsets of a finite set S, positive integer $K \leq |C|$.

QUESTION: Does C contain a cover for S of size K or less, i.e., a subset $C' \subseteq C$ with $|C| \leq K$ such that every element of S belongs to at least one member of C'?

A recognition version of the SENSOR COVERAGE problem, using previously introduced notation is:

INSTANCE: Set M of objects with weights $c_i > 0$ for all $i \in M$; set P of sensors with $|P| \leq |M|$; set N of locations to which each sensor k can possibly be assigned to the subset N^k; and positive integer $V \leq \sum_{i=1}^{|M|} c_i$.

QUESTION: Does there exist an assignment of the sensors $k \in P$ to locations in the respective sets N^k, subject to constraints (2.7)–(2.10), such that the total weight they cover is greater than or equal to V?

Equivalently, the question asks if there is a solution to the mathematical program (2.6)–(2.10) with objective function value less than or equal to $\sum_{i \in M} c_i - V$.

Theorem 2.1 *The* SENSOR COVERAGE *problem is $\mathcal{NP}$-complete.*

PROOF: Given an instance of MINIMUM COVER, let $M = S$, $|P| = K$, $|N| = |N^k|$ for $k = 1, \ldots, |P|$; $N = C$; $V = |S|$, and $c_i = \frac{|S|}{K}$ for $i = 1, \ldots, |M|$. By construction of this instance of SENSOR COVERAGE, each sensor can be placed at any position ($|N^k = |N|$ for all k)—so assigning $|P|$ sensors is equivalent to selecting $|P|$ positions (or columns in the mathematical programming representation).

C will contain a cover for S of size K or less if and only if there is an assignment of sensors to locations such they contain a subset of M whose total weight is greater than or equal to V. Since $c_i = \frac{|S|}{K}$ for all i and a total weight of greater than or equal to $V = |S|$ is sought, the sensor assignments must produce a full cover of the object array, and SENSOR COVERAGE becomes equivalent to MINIMUM COVER. □

2.4 Solution Techniques

2.4.1 Marginal Return Algorithm

A natural approach to assigning sensors to targets is to successively choose that combination of sensor k and position $j \in N^k$ that provides the greatest marginal decrease in uncovered demand. We will term the matching of a sensor with one of the positions to which it can feasibly be assigned to be a "pairing." Among all feasible pairings, the one that captures the most additional uncovered demand is the assignment made; this is a "greedy" approach. If demand values are all equal, this will correspond to selecting the sensor-location pairings that cover the greatest number of additional demands. The heuristic will be referred to as **MR-1**, and in the following specification we use this additional notation: z is a counter; v_j^k is the value that sensor k covers if it is assigned to position $j \in N^k$; V is the set of all possible v_j^k's; and C is the set of objects already covered by sensors that have been assigned.

MR-1:

1. Initialize. $z \leftarrow 0$; $v_j^k \leftarrow \sum_{i \in M_j^k} c_i \;\; \forall j \in N^k,\, k \in P$; $V \leftarrow \{v_j^k \mid j \in N^k,\, k \in P\}$; $C = \{\phi\}$.

2. $z \leftarrow z + 1$; $v_{j^*}^{k^*} \leftarrow \max_{v_j^k \in V} v_j^k$.

3. Assign server k^* to position j^*. If $z = p$, stop. All servers have been assigned. Otherwise, $C \leftarrow C \cup M_{j^*}^{k^*}$.

4. Update V. $V \leftarrow V \setminus \{v_j^k \mid j \in N^{k^*},\, k = k^*\}$.

5. Update v_j^k. $v_j^k \leftarrow v_j^k - \sum_{i \in \{M_j^k \cap C\}} c_i$. Return to step (2).

In the initialization step we create a "value list" $V = \{v_j^k\}$, where v_j^k is the aggregate value covered by assigning sensor k to position $j \in N^k$. The highest value

in V is selected, and then V is updated appropriately. The update process involves removing from V other feasible pairings that involve the sensor that has just been selected, as well as updating the individual v_j^k's remaining on V to account for the value of objects that were just covered by the most recent assignment. This procedure repeats until all servers have been assigned.

Step (1) requires at most mnp arithmetic operations; steps (2) and (4) each require scanning a list of at most np elements per iteration for a total of $O(np^2)$ steps. Step (5) can require up to m additions in computing $\sum_{i \in \{M_j^k \cap C\}} c_i$, which is the value just covered, and then m subtractions of this quantity from up to np v_j^k's. These calculations occur over all iterations, for a total of $O(mnp)$ operations. There will be p iterations of the loop. Since we assume $n \geq p$, $m \geq p$, and $n \simeq m$, the overall run time is $O(mnp)$.

The runtime analysis can be refined by considering specific data structures to efficiently perform the operations indicated; nonetheless, it still comes out to be $O(mnp)$. The list V created in step (1) can be implemented as a heap, which will require $np \log np$ time. Then in step (2) finding the maximum will be $O(1)$, while "fixing" the heap will take $O(\log np)$. Updating V in step (4) can be done in $O(n \log np)$ time if there is a pointer array into the heap such that each pairing involving the sensor just assigned can be found in $O(1)$ time. M_j^k can be stored as a singly linked list of the objects covered when sensor k is at position j, and, similarly, N_i^k as a singly linked list of the locations from which sensor k can cover object i. Two bit-vectors, M' and P', respectively, can indicate the objects uncovered and the satellites not yet assigned. Then, in general, $v_j^k = \sum_{i \in (M_j^k \cap M')} c_i$. Now, suppose that object i is covered in the most recent step and was not covered earlier. Updating v_j^k for all $j \in N_i^k$ and $k \in P'$ will take $O(\sum_{j,k} |N_i^k|)$ steps, with an extra $\log np$ factor if the v_j^k's are stored as a heap. Ignoring the extra *log* factor, the update time is $O(mnp)$. In addition, there is still the potential for there to be mnp arithmetic operations in the initialization, step (1). And while the sets N_i^k and M_j^k speed up the updating process, their creation

can take $O(mnp)$ time.

A variant of the marginal return algorithm considers the placement of each sensor individually and successively, creating a much smaller value list in the process. A priori, one would expect this version to run more quickly than the former since any list operations would be working with significantly shorter lists, although we will see that the worst case run time is the same. It can be stated as follows:

MR-2:

1. Initialize data structures. $k = 1$.
2. $v_j^k = \sum_{i \in (M_j^k \cap M)} c_i \;\; \forall j \in N^k; \;\; V^k = \{v_j^k \mid j \in N^k\}$.
3. $v_{j^*}^k = \max_{j \in N^k, v_j^k \in V^k} v_j^k$.
4. Assign server k to position j^*. If $k = p$, stop. All servers have been assigned.
5. $P = \{P \setminus k\}; \;\; N^k = \{N^k \setminus j^*\}$ for all $k \in P; \;\; M = M \setminus \{i | i \in N_{j^*}^k\}$.
6. $k = k + 1$. Return to step (2).

In the update step, step (5), we remove the sensor just assigned from the set P, and the position to which it went from the sets N^k, $k \in P$. Those objects just covered are removed from the set M, so that when the v_j^k's for the next sensor to be assigned are computed in step (2), only uncovered objects ($\{M_j^k \cap M\}$) are considered.

As observed in **MR-1**, the creation of data structures in step (1) can take $O(mnp)$ time. Computing the v_j^k's in step (2) can require $O(mnp)$ arithmetic operations. Finding the maximum v_j^k in step (3) can be done in $O(1)$ time if they are stored in a heap, however heap construction requires $m \log m$ time and is done for each of p sensors. Updating the index sets in step (5) require, respectively, $O(1)$, $O(p)$, and $O(m)$ time, and are performed for each of p iterations. Overall run time remains $O(mnp)$.

Either **MR-1** or **MR-2** can be used to provide an upper bound on the amount of demand that can be covered (or conversely, a lower bound on the amount of demand that is *not* covered). This is done by not updating the v_j^k's on the value list V in **MR-1**, which occurs in step (5). Instead, we simply return to step (2) after completing step (4). Similarly, in step (5) of **MR-2**, $M = M \setminus \{i | i \in N_{j\bullet}^k\}$ is not performed. The v_j^k's are computed in step (2) as $v_j^k = \sum_{i \in M_j^k} c_i$. This is a greedy upper bound, and equals $\sum_{k \in P} v^k$ where $v^k = \max_{j \in N^k} v_j^k$; note that the value produced by the modified versions of either **MR-1** or **MR-2** is the same. The quality of the bound provided will be dependent on the data; we conjecture that the bound will be excellent for uniformly distributed demand values. It can be arbitrarily poor. Suppose that all p sensors are capable of covering a single object whose value is 1, while all other objects have no value. Then,

$$\frac{Value(\text{bound})}{Value(\text{optimum})} = p\,.$$

2.4.2 Lagrangian Relaxation

In this heuristic a lower bound for the SENSOR COVERAGE problem is generated via Lagrangian relaxation and then used to produce a feasible solution. Subgradient optimization then provides a sequence of such lower bounds and feasible solutions. The approach we use here is based on Fisher [43] and Beasley [11].

Using t_i as the nonnegative Lagrange multipliers for constraints (2.7), the Lagrangian lower bound program, *L(t)*, is defined as follows:

L(t):

$$\text{Minimize} \quad \sum_{i \in M} (c_i - t_i)\bar{y}_i + \sum_{i \in M} t_i \left(1 - \sum_{k \in P} \sum_{j \in N^k} a_{ij}^k x_j^k\right) \tag{2.12}$$

$$\text{subject to} \quad \sum_{j \in N^k} x_j^k = 1 \quad \forall\, k \in P \tag{2.13}$$

$$x_j^k \in \{0,1\} \ \forall j \in N^k, \ k \in P \tag{2.14}$$

$$\bar{y}_i \in \{0,1\} \ \forall i \in M \tag{2.15}$$

The problem, $L(t)$, defined by (2.12)–(2.15) is a *Lagrangian relaxation* of the SENSOR COVERAGE problem. It is easy to see that for any t, $L(t) \leq v(\mathrm{SC})$, where $v(\mathrm{SC})$ is the optimal value of the sensor coverage problem defined by equations (2.6)–(2.10). The strongest Lagrangian relaxation is given by $t = \bar{t}$ such that

$$L(\bar{t}) = \max_{t \geq 0}\{L(t)\}. \tag{2.16}$$

Problem (2.16) is sometimes called a *Lagrangian dual.*

For a given set of multipliers t, this Lagrangian lower bound problem (2.12)–(2.15) is separable, and it follows that in any optimal solution $(\bar{x}, \bar{y})$:

$$(c_i - t_i) \leq 0 \ \Rightarrow \ \bar{y}_i = 1; \ \text{ otherwise } \bar{y}_i = 0\,.$$

The x_j^k's are determined as follows: for each k, choose j^* so as to maximize $\sum_{i \in M} t_i a_{ij}^k$; set the corresponding $x_{j^*}^k$ to 1, all other x_j^k set equal to 0. The objective function of (2.16) is piecewise linear and concave in t, and hence is not everywhere differentiable. As a result, subgradient optimization is used to maximize the lower bound.

We now state the algorithm.

LLB:

1. Initialize. $z_{max} = 0$; $z_{ub} = \sum_{i \in M} c_i$; $t_i = 0$ and $\bar{c}_i = c_i \ \forall i \in M$.

2. Compute the solution to the Lagrangian relaxation as detailed above with the current multipliers and let the objective function value be z_{lb}, with decision variables x_j^k and $\bar{y}_i$. Update z_{max} with the maximum of (z_{max}, z_{lb}).

3. Produce a feasible solution to the SENSOR COVERAGE problem as follows:

(a) for all i: if $x_j^k = 0 \;\; \forall\, k \in P,\; j \in N_i^k$ then object i is uncovered $\Rightarrow \; \bar{y}_i = 1$; otherwise $\bar{y}_i = 0$.

(b) Compute $\sum_{i \in M} c_i \bar{y}_i$. Update z_{ub} with the minimum of $(z_{ub}\,,\, \sum_{i \in M} c_i \bar{y}_i)$ for the feasible solution just generated.

4. Stop if $z_{max} = z_{ub}$; this is the optimal solution.

5. Calculate the subgradients for the lower bound solution:

$$G_i \;=\; 1 - \bar{y}_i - \sum_{k \in p} \sum_{j \in N^k} a_{ij}^k x_j^k \quad \forall\, i \in M\,.$$

6. Stop if $\sum_{i \in M} G_i^2 = 0$ since a stepsize cannot be computed in step (7).

7. Determine a stepsize T using:

$$T \;=\; f\,(1.05 z_{ub} - z_{lb}) \,/\, \left(\sum_{i \in M} G_i^2 \right) .$$

Initially f=2, and if the lower bound has not increased for the last 30 iterations then f is halved. (This is a typical approach for subgradient optimization; the computation of f is based on [11] and [43].)

8. Stop if $f \leq \epsilon$ (e.g., $\epsilon = .005$).

9. Update the multipliers with $t_i = t_i + T G_i \;\; \forall\, i \in M$, subject to $t_i \geq 0$. Return to step 1 and iterate.

It can be shown [60], [101] that this approach converges if $\sum_{k=1}^{\infty} T^k = \infty$ and $\lim_{k \to \infty} T^k = 0$, where T^k is the step length in iteration k. These conditions are satisfied if one starts with f=2 and periodically reduces f as shown in the algorithm.

The gap between the upper and lower bounds reflects the quality of the best feasible solution produced.

2.4.3 Decomposition

The block-angular structure of this problem, in which a number of separate satellite assignment constraints (2.8) are linked together by a common covering constraint (2.7), makes the LP relaxation of this problem a candidate for solution by Dantzig-Wolfe decomposition. This is an iterative approach wherein a number of different subproblems, each of which is a linear program, are solved. The objective functions of these subproblems for one iteration are based on the dual price information of the linear programs in the preceding iteration. The costs serve to "coordinate" the subproblems so that eventually the original problem is solved. Our description of Dantzig-Wolfe decomposition follows that of Luenberger [75, pp. 68–75].

The kth subproblem has as its constraint set

$$\sum_{j \in N^k} x_j^k = 1 \tag{2.17}$$

$$x_j^k \geq 0 \quad \forall\, j \in N^k \tag{2.18}$$

for $k = 1, \ldots, p$. The subproblem polyhedrons are all bounded, and thus their feasible regions, S^k, consist of points that are expressible as convex combinations of their respective extreme points. If the extreme points of S^k are $\cup_{j \in N^k} \{\tau_j^{k1}, \tau_j^{k2}, \ldots, \tau_j^{kG^k}\}$, where G^k is the number of extreme points for S^k, then any point x_j^k, $j = 1, \ldots, N^k$, in S^k can be represented by

$$x_j^k = \sum_{l=1}^{G^k} \alpha_j^{kl} \tau_j^{kl}$$

where

$$\sum_{l=1}^{G^k} \alpha_j^{kl} = 1$$

$$\alpha_j^{kl} \geq 0 \quad l = 1, \ldots, G^k,\ j \in N^k.$$

The α_j^{kl}'s are the weights for the extreme points.

The LP relaxation of the original integer program can be transformed to an equivalent master problem for which the objective is to optimize the weights for each S^k, as well as to minimize uncovered demand ($\sum_{i \in M} c_i \bar{y}_i$). The original SENSOR COVERAGE problem becomes

$$\text{Minimize} \quad \sum_{i \in M} c_i \bar{y}_i \tag{2.19}$$

subject to

$$\sum_{k \in P} \sum_{j \in N^k} \sum_{l=1}^{G^k} a_{ij}^k \alpha_j^{kl} \tau_j^{kl} + \bar{y}_i \geq 1 \quad \forall i \in M \tag{2.20}$$

$$\sum_{l=1}^{G^k} \alpha_j^{kl} = 1 \quad \forall k \in P,\ j \in N^k \tag{2.21}$$

$$\bar{y}_i \leq 1 \quad \forall i \in M \tag{2.22}$$

$$\bar{y}_i \geq 0 \quad \forall i \in M \tag{2.23}$$

$$\alpha_j^{kl} \geq 0 \quad l = 1, \ldots, G^k,\ k \in P,\ j \in N^k. \tag{2.24}$$

The master problem has as decision variables α_j^{kl} for $l = 1, \ldots, G^k$, $k \in P$, $j \in N^k$ and $\bar{y}_i$ for all $i \in M$.

We will suppose that at some stage of the simplex method for the master problem we know the basis and the corresponding simplex multipliers (λ_i for the ith linking constraint and π_k for the constraint summing the weights for the extreme points of the kth subproblem. An algorithm for the solution of the master problem is the following:

DW:

1. Calculate the current basic solution for the master problem, and compute the simplex multipliers.

2. For all $k \in P$ find the optimal solution x^{k*} of the kth subproblem using constraints (2.17) and (2.18) and with the following as the objective function:

$$r_k^* = \text{Minimize} \left(- \sum_{i \in M} \lambda_i \sum_{j \in N^k} a_{ij}^k x_j^k - \pi_k \right) . \tag{2.25}$$

3. If $r_k^* \geq 0$ for all k, stop. The current solution is optimal.

4. Determine which extreme point (column) is to enter the basis by choosing the minimum r_k^*.

5. Update the basis of the master problem using the revised simplex method and return to step (1).

The subproblem in step (2) can be solved for *binary* x_j^k quite easily: select the column $j^* \in N^k$ that maximizes $\sum_{i \in M} \lambda_i a_{ij}^k$ and then set $x_{j^*}^k = 1$, $x_j^k = 0$ for $j \neq j^*$. Note that this is a greedy algorithm: the sensor location that provides the greatest weighted coverage of demands, where the weights are given by the simplex multipliers, is selected.

We have not constructed subproblems for the $\bar{y}_i$ decision variables. The subproblem that would be solved in the Dantzig-Wolfe algorithm would be:

$$\text{Minimize} \sum_{i \in M} (c_i - \lambda_i) \bar{y}_i \tag{2.26}$$

$$\text{subject to } \bar{y}_i \leq 1 \ \forall i \in M \tag{2.27}$$

$$\bar{y}_i \geq 0 \ \forall i \in M . \tag{2.28}$$

This can be solved by inspection: $\bar{y}_i = 1$ if $c_i - \lambda_i \leq 0$ and $\bar{y}_i = 0$ if $c_i - \lambda_i > 0$. The extreme points generated for $\bar{y}_i$ are either 0 or 1, which are also the values that it will

assume in the master problem as we have stated it. As noted earlier, $\bar{y}_i$ will assume its minimum, which is either 0 or 1.

2.5 Computational Performance

2.5.1 Introduction

In this section we discuss a probabilistic model for satellite location and coverage. With this model, the marginal return and Lagrangian relaxation (LR) heuristics have been tested using representative problems formed with randomly generated demand data. The general parameters that distinguish a representative satellite-based sensor coverage problem include the altitude and characteristics of the platform's orbit and the individual instrument's specific technical features. Examples from the Eos program will help to clarify what we mean.

The altitude of a low-earth orbit satellite will range from a couple of hundred kilometers up to 1500 or so. For a satellite orbiting at 400 km the horizon is approximately 2300 km distant (a farther observation distance is possible if one considers events occurring above the surface of the earth), which is easily within a nominal sensing range of 3000 km for most types of surveillance and environmental instruments. A 2300 km feasible coverage circle, if centrally placed, would effectively cover the continental US. For a 15° sensor field-of-view (FOV) the actual imaging region is approximately 300 km wide when 2300 km distant and 50 km wide at 400 km. These figures highlight the different types of field-of-view that are relevant in coverage planning. The instrument instantaneous-field-of-view (IFOV) is the size of the area that is characterized in a single pixel, or element, of a measurement which may be composed of many pixels. If many pixels are measured, it can be done either simultaneously by some type of area array detector, or sequentially, using perhaps a "push-broom" type detector to sweep over a region. Many pixels need *not* be measured, however, as will be shown in the following Eos examples. The general instrument field-of-view refers

to the total area that is characterized in a single measurement, which typically consists of many smaller pixels. The total region within which measurements can feasibly be made is determined by either pointing or geometry (line-of-sight) restrictions that limit instrument coverage. Sensor pointing can be accomplished by physically moving a scanning mirror, employing electronic beam steering, or other means. These terms are often blurred: field-of-view is sometimes used to refer to the angles within which the pointing capabilities of an instrument permit it to make measurements, and IFOV is sometimes cited as the the angle that specifies the region captured within a single measurement.

The numbers just given are nominal and will vary depending on orbit altitude and sensor parameters; a couple of specific examples taken from the suite of instruments proposed for Eos should make the ideas more concrete. Consider first the Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER) [83]. ASTER's science objectives include surface and cloud imaging with high spatial resolution and with multispectral channels from visible to thermal infrared. At the Eos 705 km orbital altitude, ASTER will have pointing capabilities such that any point on the globe will be accessible at least once every 16 days. The instrument's swath is 60 km at nadir, but is pointable cross-track ± 106 km. It's spatial resolution, which is determined by the IFOV, varies by wavelength: 15 meters for visible and near-infrared (VNIR); 30 meters for short-wave infrared (SWIR); and 90 meters for thermal infrared. All pointing is near nadir except for VNIR forward, which is 29.7° forward of nadir. Fields of view, including SWIR and TIR, are as follows:

FOV: TIR = SWIR = 4.9° (nadir) × IFOV
VNIR = 6.09° (nadir) × IFOV, 5.19° (forward) ×IFOV

IFOV: SWIR = 43μrad (nadir)
TIR = 128 μrad (nadir)
VNIR = 21μrad (nadir), 18.1μrad (forward).

The field-of-view determines the 60 km measurement swath.

The Eos Synthetic Aperture Radar (SAR), mentioned in section 2.3.1, uses electronic beam steering in the cross-track direction to acquire images at selectable incidence angles of 15° to 50° from both sides of nadir—this is termed its field-of-view in [83]. Its IFOV varies from 30-500 km, with variable spatial resolution, depending on the mapping mode (which can be local, regional, or global). The SAR is an active instrument (it emits electromagnetic radiation and then measures the response, rather than passively observing), and its instantaneous measurements characterize the entire swath. Their resolution is determined by the frequency used in each of the mapping modes (L, C, and X bands).

Fully at the opposite end of the resolution spectrum from the high spatial capabilities of ASTER is the Multifrequency Imaging Microwave Radiometer (MIMR). MIMR will be used for retrieval of numerous atmospheric and oceanic parameters, including precipitation, soil moisture, global ice and snow cover, sea surface temperature and wind speed, atmospheric cloud water, and water vapor. It is designed to have a cross-track swath of 1400 km at an incidence angle of 50°, with spatial resolution ranging from 4.86 km at 90 GHz to 60.3 km at 6.8 GHz. This will provide a minimum of three day global coverage of the earth; at high latitudes there will be daily coverage. MIMR's IFOV is $\pm 80°$ cross-track.

As can be seen from the examples just given, the potential area within which an instrument can be pointed and images taken is quite large. The number of feasible sensor locations is in principle infinite, and it is still typically quite large even if we discretize the space so that the corresponding footprints constitute a partition. This has implications for the algorithms that are employed to determine assignments. Thus in our model we make assumptions regarding the set of positions to which each sensor can potentially be assigned.

One way to limit the size of N^k is to let N^k be restricted to be a subset of M, the target locations. Although this restriction can be sub-optimal, it is regarded as a reasonable one. Instances in which it results in poor assignments will be either

infrequent and pathological. An example would be three objects that form the vertices of a properly sized equilateral triangle. They could all be viewed in a single image if the sensor were pointed at the center of the triangle, while only two could be viewed at a time if the sensor were directed at any of the vertices of the triangle.

For a plane of 5 sensors that can cover a consolidated region containing 200 demands, however, there are potentially 1000 possible sensor-location combinations—fewer need be considered if some demands are not visible to a subset of the satellites. Moreover, there are realistic satellite scenarios that envision a constellation of 50–60 satellites, although they would not all be in a given region. For example, Motorola's proposed Iridium communications satellite array would ultimately consist of 66 satellites; prior to complete deployment there will be significant tasking problems with the number of satellites in the dozens. When fully operational, the system would provide wireless telephone connections over virtually every spot on the earth's surface and be capable of handling 200 simultaneous calls. With fewer satellites available, far less area can be covered and there will be a need to solve dynamic coverage problems so as to insure that ground "gateways" have a satellite link (otherwise communications is lost). Conversely, gateways cannot be established for a given array of satellites unless optimization of the coverage problem establishes that they can be serviced adequately. A recent policy decision by NASA regarding Eos [2] will result in more, smaller (and less expensive) satellites, again producing greater management complexity. Part of the original motivation to have large platforms on which a dozen or more instruments could be placed was that it reduced the scheduling complexity in trying to achieve desired degrees of simultaneity and similarity between measurements taken by complementary instruments.

The objects, or demand points, of which sensor coverage is sought, were produced by generating up to 4500 randomly placed points on the surface of the earth. Different sets of such demand points were used for different problem instances. To generate a random vector uniformly distributed on the surface of a sphere of radius

r, we simulate a random vector uniformly distributed in the 3-dimensional hypercube $\{-1 \leq x_i \leq 1\}_{i=1}^{n}$ and then accept or reject the sample (X_1, X_2, X_3), depending on whether the point is inside or outside the sphere [97, p. 51]. The algorithm is as follows:

1. Generate U_1, U_2, U_3 from a uniform distribution over the interval (0,1).
2. $X_1 \leftarrow 1 - 2U_1$; $x_2 \leftarrow 1 - 2U_2$; $X_3 \leftarrow 1 - 2U_3$; $Y^2 \leftarrow \sum_{i=1}^{n} X_i^2$.
3. If $Y^2 < 1$, accept $Z = (Z_1, Z_2, Z_3)$, where $Z_i = r \times (X_i/Y_i)$, $i = 1, 2, 3$ is the desired vector.
4. Return to step (1).

Observe that to generate a random vector uniformly distributed inside a sphere, we need only modify step (3) to accept the point under consideration if $Y^2 \leq 1$.

For a total of 64 satellites with an instrument field-of-view of 15° and at an orbital altitude of 1000 km, we can generally cover at most 20% of the earth in our examples. If demand placement is uniformly distributed, it constitutes a worst case test of the ability of our algorithms' coverage performance; for certain scenarios, such a distribution is also realistic. A strategic surveillance system will potentially want to scan the entire surface of the earth. A typical surveillance strategy might be to cover the entire globe at all times with broadband sensors, and then "sample" those areas that are of particular interest with high resolution sensors. Note that the nominal scan parameters we have assumed, e.g., a 15° field-of-view, are more on the order of moderate, rather than high, resolution. For applications other than surveillance, however, demand will have different distributions. HIRIS is used primarily to scan the continental land masses, coastal zones, and inland bodies of water; it is not used to make any measurements over the open ocean. Thus "demand" for an imaging spectrometer is clustered over less than 25% of the earth's surface. Given Eos's orbital inclination, the North-South America corridor is an ideal opportunity for

imaging—in a single pass targets on both continents can be imaged. Coverage for the communications satellites envisioned for Iridium would be even more focussed: earth gateways, which would require continuous antenna coverage from the satellites, would be located primarily in the vicinity of high population density areas in North America, Japan, and Western Europe.

The 15° field-of-view for the area coverage of a single measurement was selected as a compromise value for computational purposes. In practice, the size of the region covered in an image will be the product of both design decisions made before a sensor is ever fielded, and operational decisions. Design specifications will include the size and type of detectors, as well as determining the scanning capabilities. In addition, the imaging characteristics will be regulated by operational decisions regarding how much of the scanning capabilities to use, and image sequencing strategies that produce continuous imaging—essentially turning on the sensor and letting it run. Such approaches will be examined in more detail in Chapter 3.

2.5.2 Computational Tests

Computational tests of the algorithms implemented have been performed on both VAXstation 3100s and DECstation 5000s. Both the marginal return and LR heuristics have been implemented in the programming language C, using linked lists to store the sparse coverage matrix—a data structure that requires much less space than a matrix representation. Linked lists call for $\sum_{i \in M, k \in P} n_i^k + \sum_{j \in M, k \in P} m_j^k$ space, while the matrix takes $m \times \sum_{k \in P} n_k$. Since $n_i^k \leq n^k$ and $m_j^k \leq m$, it follows that the linked lists make for much more efficient use of memory. The a_{ij}^k's (the elements of the coverage matrix A) are very useful conceptually, but should not be stored explicitly. Problem dimensionality is less of a concern in the performance of the greedy heuristic than in the Lagrangian relaxation, since the greedy heuristic eliminates all of the assignment variables associated with a particular server at each iteration. In the Lagrangian relaxation, however, each Lagrangian lower bound problem and corresponding feasible

solution entails dealing with all feasible assignment variables, and thus each iteration is comparable to solving the entire greedy problem.

In generating satellite positions, we assume circular orbits and symmetric constellations. The true orbit of a satellite around the earth is an ellipse, but since we are considering low-earth orbits the orbital periods are short and there is only very small orbital eccentricity. This makes circular orbits a reasonable approximation. Two terms from astrodynamics are helpful in understanding symmetric constellations: mean anomaly and ascending node crossing. Mean anomaly is the angle at which a satellite is located within its orbital plane. The angle must be related to something, and here mean anomaly is referenced to the ascending node crossing. The ascending node crossing is the point (measured in longitude) at which a satellite's orbital plane crosses the equatorial plane as the satellite ascends into the northern hemisphere. Symmetric satellite constellations mean that satellites in a single orbital plane are distributed evenly in mean anomaly, and orbital planes are distributed evenly in longitude of ascending node. The assumption of symmetric constellations has previously been used in [9] for the continuous global and continuous regional earth coverage problem, and in [35] to obtain intermittent ground coverage of the earth. Note that these two assumptions are used to generate data for the static coverage problems, but the algorithms themselves do not exploit any symmetry, nor is it obvious how to do so.

Tests were performed with 8 planes of 8 satellites each. With 4500 ground sites, typically less than 1% of the demand was invisible to all satellites, while, as observed earlier, generally at most 20% could be covered with a single set of assignments. With fewer than 4500 demands (e.g., 500 or 1500) usually all were capable of being covered. The computational results are shown in Table 2.1, where we use previously introduced abbreviations for the algorithms (**LLB, MR-1, MR-2**). The entries for upper and lower bounds are expressed as the fraction of total value that is *uncovered*, and represent averages taken over 50 problem instances. For **MR-1** and **MR-2**, they

have been converted from the corresponding values for the *lower* and *upper* bounds on covered value that the marginal return algorithms actually compute (remember that in **LLB** our objective is to *minimize* the *uncovered* value); this is done to provide ease of comparison in contrasting **LLB** performance with that of the marginal return algorithms **MR-1** and **MR-2**. For instance, the figure of 0.8026 for **MR-1** and 4500 demand points is a lower bound on the fraction of demand value that will be left uncovered; as an output of the algorithm, however, it was an *upper* bound on the covered value of 0.1974 ($= 1 - 0.1762$). The gap between the upper and lower bounds for both **LLB** and the **MR** algorithms is expressed an a percentage of the uncovered value lower bound, as was computed directly for the Lagrangian. Alternatively, the gaps could have been computed as a percentage of the *covered* value lower bound; if so calculated, the gaps will be larger. Using the same example as before for **MR-1** and 4500 demand points, we have the gap in Table 2.1 shown as a percentage of uncovered value

$$\frac{0.8143 - 0.8026}{0.8026} = 1.46\%$$

or, alternatively, as a percentage of covered value

$$\frac{(1 - 0.8026) - (1 - .8143)}{(1 - .8143)} = 6.30\% .$$

Heuristic	# Demands	# Iterations	Lower Bound	Upper Bound	Gap in %
LLB	500	500.04	0.6332	0.6644	4.93
	1500	492.62	0.7445	0.7561	1.55
	4500	492.86	0.8124	0.8238	1.40
MR-1	500	—	0.5846	0.6388	9.27
	1500	—	0.7227	0.7482	3.53
	4500	—	0.8026	0.8143	1.46
MR-2	500	—	0.5846	0.6492	11.05
	1500	—	0.7227	0.7529	4.18
	4500	—	0.8026	0.8170	1.79

Table 2.1: Summary of computational experience for coverage algorithms; bounds and gaps are for *uncovered* value.

In an absolute sense, the entries in Table 2.1 for the upper and lower bounds don't convey much useful information, since they are the *averages* from 50 problem instances. It is a *relative* figure of merit, which, if averaged, is most useful in comparing different heuristics—it conveys expected performance. The solution gap, computed as a percentage of the lower bound, is such a relative measure. But here, however, we are randomly generating problems from a specified distribution (uniform), and it makes sense to compute the *expected value* of upper and lower bounds on the optimal solution for this problem *class*. If viewed in this light, the figures in Table 2.1 provide insight into how the heuristics perform.

As can be seen, the average gap between the Lagrangian relaxation upper and lower bounds for uncovered value was found to be less than 2% for the problems with 1500 and 4500 demands. For the larger problem instances, the value that was covered represented approximately 18% of the total coverable value. In comparison, the gap between the upper and lower bounds from marginal return assignments for problems with 1500 and 4500 demands ranged from 1.5% to 4.2%. In all cases, however, the value covered by assignments from the marginal return algorithm fell *within* the gap from the Lagrangian relaxation. In particular, this means that the greedy algorithms provided better feasible solutions. We conclude that the marginal return algorithm provides very good assignments, but not very good bounds. The marginal return algorithm ran significantly faster than the Lagrangian for large problems, typically taking about 30 seconds per instance compared to almost 240, although they were comparable when tested against smaller sized instances. Considerable code optimization could be done for both implementations, so this is more an indication of relative performance rather than a finding.

In Figures 2-2 and 2-3 the gaps between bounds are displayed in dot charts, grouped in the first case by the algorithm used and in the second by the size of the problem instance. Both serve to reinforce the observation just made, which is that smaller gaps are provided by the Lagrangian; it should be remembered, though,

that both **MR-1** and **MR-2** provide better feasible solutions than did the **LLB**—this can be seen in boxplots of the lower bounds. Figure 2-3 shows that as demand gets increasingly dense, there is increasingly less difference between "good" and "bad" assignments. In effect, no matter where you point the sensor, you capture comparable amounts of value. Thus the gap between upper and lower bounds shrinks since different assignments become roughly equivalent.

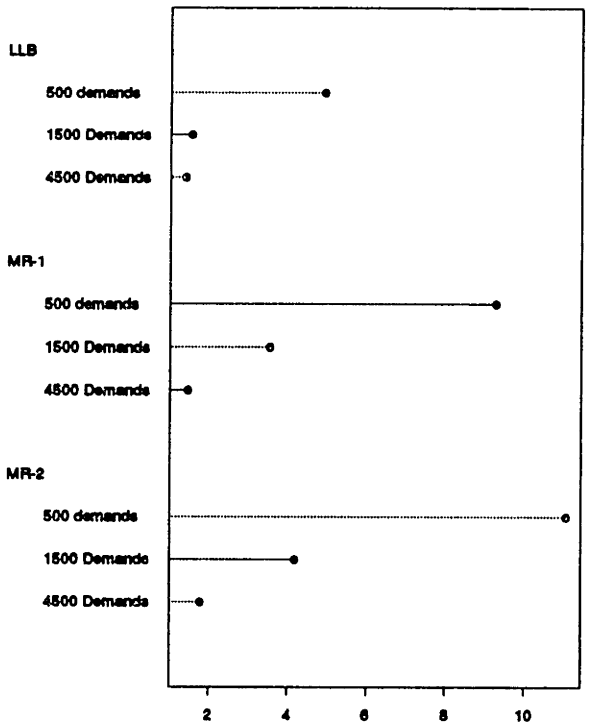

Figure 2-2: Solution gaps in %, grouped by heuristic.

In Figures 2-4, 2-5, and 2-6, boxplots[1] show the spread of lower bounds (feasible solutions) for *covered* demand in 50 problem instances. This is the data as computed by **MR-1** and **MR-2**, as opposed to **LLB**, which computes a lower bound on *uncovered* demand. The outcomes plotted for **LLB-1** reflect the complements of upper bounds on *uncovered* demand. For all levels of demand, both **MR-1** and **MR-2**

[1]The box in a boxplot contains the middle half of the data. The whiskers extending from the box reach to the most extreme non-outlier; points lying beyond 1.5 times the inter-quartile range are plotted individually.

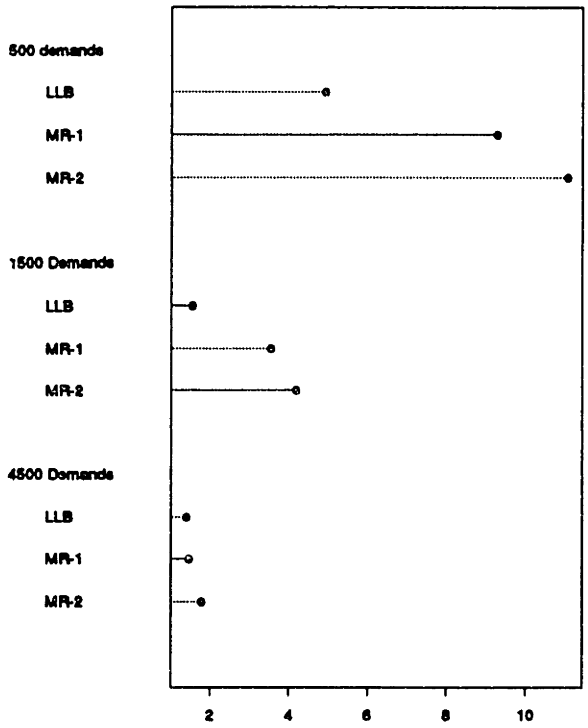

Figure 2-3: Solution gaps in %, grouped by number of demands.

provide better feasible solutions (e.g., higher expected covered value from these algorithms). Moreover, both of the **MR** heuristics provide *consistently* good solutions, as evidenced by the tighter distributions about the mean; this is particularly true for the problem instances with 1500 and 4500 demands.

2.5.3 Summary

Both marginal return and Lagrangian relaxation algorithms have been specified, implemented, and computationally tested. They have been demonstrated to be capable of providing good bounding information, as well as quite satisfactory feasible solutions. In addition, they can perform very quickly. A question that arises is whether or not the problems against which these heuristics have been tested, although it has been argued that they present a worst case scenario of demand distribution, are inherently easy. In particular, might it be that any algorithm, albeit perhaps simple-minded

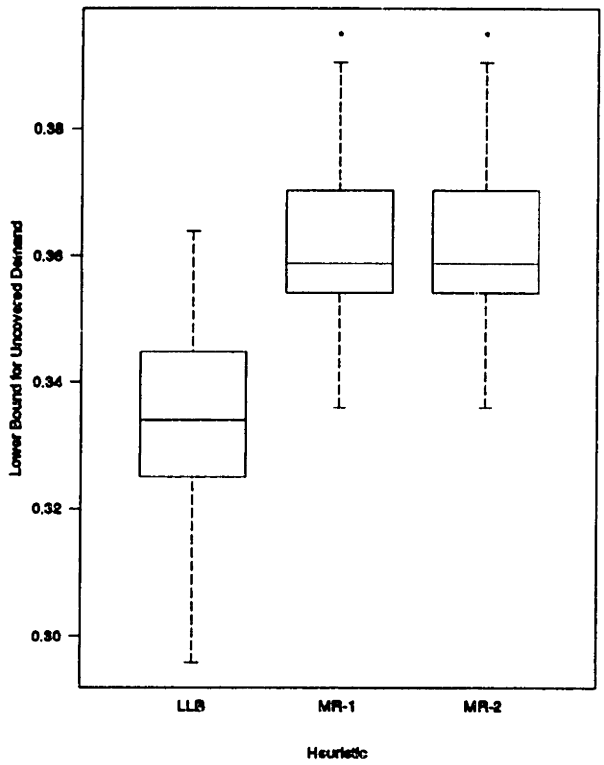

Figure 2-4: Boxplot of 50 lower bounds for *covered* demand (500 demands).

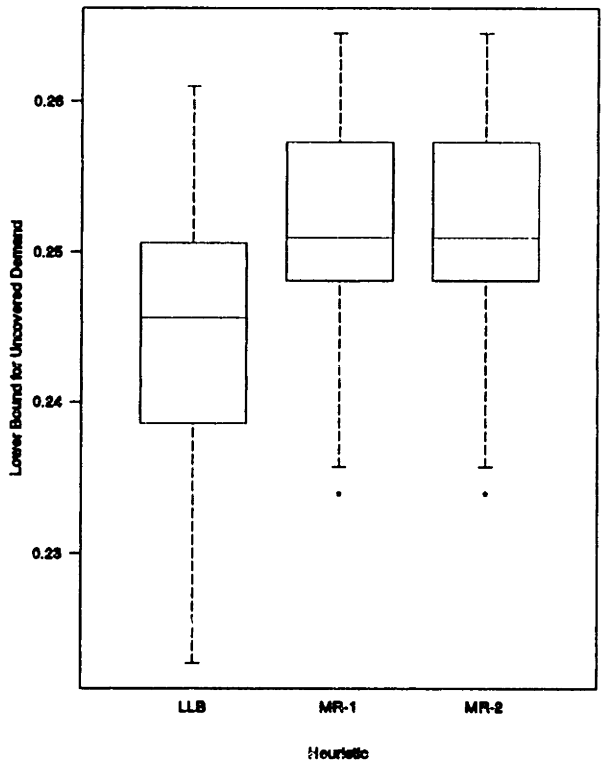

Figure 2-5: Boxplot of 50 lower bounds for *covered* demand (1500 demands).

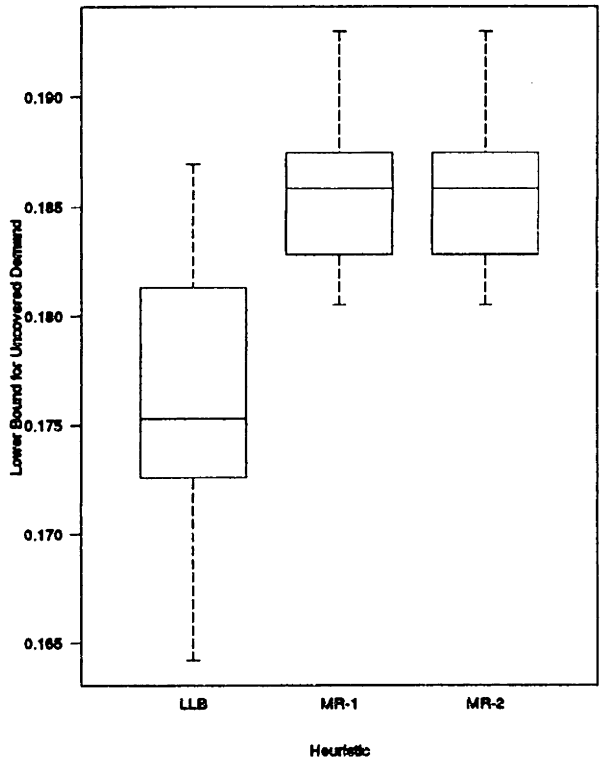

Figure 2-6: Boxplot of 50 lower bounds for *covered* demand (4500 demands).

(and straightforward), would perform just as well?

With this objective in mind, three "dumb" algorithms were specified and tested against some of the same problem instances used in the computational results displayed in Table 2.1. The three algorithms, each increasingly less "dumb," are

1. Point each sensor at nadir. Objects will be observed only if they fall within a path running along the satellite ground-track whose width is the sensor field-of-view.

2. Point the sensor at the first object that is found to be within the sensor's pointing range. The value covered will be that within the field-of-view of the sensor when it is targeted at the object so identified.

3. Point the sensor at the highest value object within the sensor's feasible region; if it is covered, then target the next highest value object. If that is covered

also, then target as in algorithm 2. The value covered will be that within the field-of-view of the sensor when it is pointed at the appropriate object.

The fraction of demand value covered increased with each heuristic, averaging 0.0253, 0.0644, and 0.0731, respectively. They were run against problem instances with 4500 demands, for which experience indicated they should perform the best (Figure 2-3). Even the third one, however, captured only about a third of the value that **MR-2** did. For a greedy algorithm to be effective, it must search a reasonably sized neighborhood—the ones searched here are too small.

2.6 The Dynamic Coverage Problem

Our interest now is in scheduling the satellites over a small time horizon. As noted in the introduction, maximizing the coverage of ground sites is no longer the sole objective. There are new concerns that must be considered, either as additional objectives or as constraints on the assignment process. We will discuss each additional dimension in turn.

2.6.1 Erratic Scanning

The time it takes to form an image on a receiver is on the order of tenths of a second or less, while it takes seconds to reorient the sensor itself—an order of magnitude difference. By reorientation we refer to the actual repositioning of the sensor (this may entail physically moving a scanning mirror) that is necessary to cover a different surface region which is considerably distant from the location last covered. Thus the desire to sequence demands so as to achieve a smooth scan profile is a reasonably important one; if the objects targeted by a particular sensor are well-ordered, it should be possible to provide more coverage within a given time frame.

There are several ways of addressing this issue in the context of a finite horizon problem where each sensor will perform several scans while moving in space. One

approach is to define pairwise costs between potential sensor locations that reflect the desirability of scheduling sites as successive assignments for a particular sensor: locations that are close incur a smaller cost than those that are far apart. The cost structure could be either linear or concave, the latter reflecting a diminishing incremental cost for increasing distance. A realization might simply be the Euclidean or great circle distance between locations, appropriately scaled; it could also be the *time* it takes to reposition the sensor when the indicated locations succeed one another. Such a sequencing cost, d_{j_1,j_2}, can be included in the objective function:

$$\text{Minimize} \sum_{k \in P} \sum_{j_1, j_2 \in N^k} \sum_{l \in L} d_{j_1,j_2} x^k_{j_1,j_2}$$

where $x^k_{j_1,j_2}$ is 1 if sensor k is assigned to j_2 immediately after being assigned to j_1, and 0 otherwise. The cost is incurred only when the appropriate assignment variable is nonzero.

2.6.2 Time Coverage Constraints

Time constraints must be included to insure that any sensor-position pairings selected are geometrically feasible at the time they are to be executed. Thus sensor k can be assigned to position $j \in N^k$ only during some specified time interval: (b^k_j, f^k_j). For an assignment to be feasible, using the variable $x^k_{j_1,j_2}$ introduced above, it is then required that:

$$x^k_{j_1,j_2} = 1 \;\Rightarrow\; b^k_{j_1} \le t^k_{j_1} \le f^k_{j_1} \text{ and } b^k_{j_2} \le t^k_{j_2} \le f^k_{j_2}$$

where $t^k_j \in [0, T]$ is the time that sensor k is assigned to position j and T is the time horizon. Note that when a time variable is employed, we also need a constraint of the form

$$t^k_{j_2} \ge x^k_{j_1} t^k_{j_1} + x^k_{j_1} t_{j_1,j_2}$$

where t_{j_1,j_2} is the "transit" or repointing time between j_1 and j_2. If the pairwise costs d_{j_1,j_2} between successive locations take the value of the transit time, however, then this constraint is not needed.

Alternatively, the coverage constraints can be changed to reflect that the footprint of a sensor will vary with time, even when assigned to the same location. Then a_{ij}^k becomes $a_{ij}^k(t)$, which equals 1 if sensor k can cover position j from location i at time t and 0 otherwise. With time as a continuous variable, this will no longer be a discrete model. Time can be discretized, however, and then the coverage coefficients would take the form a_{ij}^{kt}, where a discretization interval is defined and the coefficients are specified for each increment. Even for a modest time horizon this would result in a huge state space, so some sort of a state space relaxation solution approach would likely be needed.

A heuristic approach to addressing the erratic scan issue in a dynamic formulation essentially decomposes the assignment process. It entails first determining the places that each satellite will scan, and then ordering the assignments so as to create a smooth sequence. The ordering could be done greedily, successively choosing the closest location among the sites regardless of their value. Following this with an interchange heuristic would reduce the impact of the final several assignments, which could be very unfavorable. Unless all of the assignments are feasible throughout the time horizon, it becomes necessary to insure that the ordering produced does not violate the time windows during which the different sites selected for a particular sensor are available for its placement. An alternative to the greedy ordering with feasibility checks is to solve the sequencing of the selected sites for a particular sensor as a shortest path problem with time windows. This heuristic approach has the advantage of decomposing a large, complex problem into smaller components that are probably more tractable.

2.6.3 Time-varying Demand Value

2.6.3.1 Recency Effect

In the static problem it is assumed that there is a single value that represents the relative importance or desirability of viewing each of the various objects that are within the range of the sensor array. From a practical perspective, it may be difficult to assign such values to a large number of demands. One alternative is to create a relatively small number of equivalence classes: groupings into which the various objects are placed such that all of those that are in a particular class are of roughly comparable importance. Presumably it would be much easier to make such judgments.

When considering a finite horizon dynamic problem, the recency with which a particular object has been visited will affect how desirable it is to revisit it. If this were not the case, we would tend to observe the same set of objects repeatedly until it became infeasible to do so because of geometry. There are a couple of ways to address this concern; one is to allow a demand's value to vary with time. There are several different functional forms that could be assumed to reflect the time-varying character of object values. For instance, there could be zero value for observing an object i again if the time since the last visit is less than some threshold level, η_i, and then once that level is reached the value climbs in either a linear or a non-linear and concave fashion. Alternatively, it could be assumed that it is always desirable to observe the demands, and that when viewed their value merely resets to a lower level from which it then grows. Both can be represented by:

$$c_i(t) = \alpha_i + \beta_i t_i$$

for linear growth and by:

$$c_i(t) = \alpha_i - \beta_i e^{-\gamma_i t_i}$$

for concave growth. In both cases t_i is measured from the time that object i was

last observed; α_i and $\alpha_i - \beta_i$ are the intercepts, respectively. The linear case grows without bound, while the nonlinear case has a limit of α_i. Either scenario is arguably plausible: in the former case a revisit will ultimately be forced, while in the latter it is implied that there is a limit to the value of that particular site.

Another means of addressing the frequency of observations/ground contacts is to establish minimum "revisit times"—the interval of time during which no satellite observes a particular ground site. An upper bound on the maximum revisit time t_i for an object i would be specified based on mission objectives:

$$t_{j_2}^{k_2} - t_{j_1}^{k_1} \leq t_i$$

where $t_{j_2}^{k_2}$ is the minimum time greater than $t_{j_1}^{k_1}$ such that $i \in M_{j_1}^{k_1}, M_{j_2}^{k_2}$ for some j_1, j_2 and k_1, k_2. The effects of such a requirement would drive the assignment process in a dynamic model. There would be two levels of observation priority, revisit time and intrinsic value.

2.6.3.2 State Changes

Another dimension to the time-varying character of object values in addition to that attributable to the recency with which a site was viewed, is the potential for "state changes." This could correspond, for instance, to an event such as a missile being launched or a volcano erupting at a particular location. The implication is that it becomes highly desirable to scan that region again immediately. If indeed a launch has occurred, then sensor assignments will subsequently be made with the objective of tracking (with a potential need for multiple sensors per object) rather than surveillance. If a relook reflects that a launch has not occurred, however, then the value of that particular demand will reset to the level that would be appropriate for having just been viewed.

There are potentially several ways to model such object value state changes. At one level, a sensitivity analysis of an algorithm's performance will reflect how great a

change is necessary in an object's value to produce different assignments or sequencings. Alternatively, if the solution to the dynamic model entails forming a rolling schedule then only the first period's decisions will be implemented. One period later the multiperiod problem is updated as better information on demand value becomes available, and the procedure iterates. Such a rolling horizon approach inherently addresses the problem of changing demand values. For the purposes of simulations one could consider demand value to be Markovian where there is a nonzero probability that the existing value jumps to one of a finite number of other states at specified times.

2.6.4 Problem Approaches

Several simplifying assumptions can be made in considering solution approaches to the dynamic problem. A significant one is that if the time frame over which sensor assignments are to be computed is sufficiently short, such as 30–60 seconds, it can be assumed that the satellite does not move to an extent that would significantly affect the sequencing of selections due to geometric restrictions on the time of the assignments. This does not eliminate such constraints, but rather relegates them to the domain of preprocessing. For each assignment update computation, the input data will reflect the changes in sensor coverage capabilities that have occurred since the last assignment. This implies that over up to a one minute period, the areas that cannot be continuously covered are small relative to the area that is covered. Thus if the geometric domain under consideration for sensor assignments is restricted to that for which there is continuous coverage, there is little loss.

In addition, focussing on assignments for a single satellite is a means to decompose the dynamic problem. This is not done without loss of generality, but the potential reduction in problem complexity and increase in computation speed are reasons for doing so.

We have suggested two general approaches to the dynamic problem. The first

would exploit work done on the static problem, by essentially embedding it in a rolling horizon model for sensor assignments. Rolling planning horizons come from production planning [23], [69], where intermediate-range forecasts are often done on a "rolling" basis. At the start of each period, a production plan is set for that period and for the next several periods on the basis of the current inventory and the forecasted demands for those periods. In each period, the current period's plan becomes firm and the rest remain tentative. Various heuristics are used to set a plan for the current period in the hope of avoiding poor inventory positions later on. It is a *rolling* procedure because it is redone each period; the *time horizon* is the number of periods into the future for which demands are forecast and tentative production plans are set. Examples of models for which rolling horizon procedures are applied are the facilities in series inventory model [23] and the dynamic lot size model [69].

In such a framework for dynamically managing sensor assignments, the static coverage model could be used to provide current assignments based on the the best known values for various surface objects to be scanned. Future assignments could be computed based on current estimates of values for objects not being scanned. Then in the next period, assignments would be updated for those periods for which coverage had tentatively been computed, with the assignments for the current period being made firm. The updating would incorporate any changes that had occurred in estimated object values based upon observed events or recent coverage. A plan would also be drafted for tasking sensors in the newest period added to the time horizon. An adaptation to the static model would be necessary to insure that smooth scan profiles are created. One way of doing this would be to modify object values to reflect their distance from the nearest location that was scanned in the most recent period. This could be done prior to updating and then making firm the current period's assignments.

The other general approach would take advantage of the vehicle routing problem (VRP) flavor that arises in the dynamic problem if we incorporate pairwise costs

between surface locations. In a problem with P sensors for which L assignments are to be determined, we essentially seek to construct P minimum length paths each consisting of L arcs that maximize covered value. The use of flow variables ($x^k_{j_1,j_2}$) and time-coverage restrictions produce a problem similar to a VRP with time windows. There are a couple of critical differences: a site need not be visited for its "demand" to be satisfied, and not all demands need be met. Nonetheless, this approach allows drawing upon the rich literature for vehicle routing problems; a detailed survey compiled by Bodin et al. in 1983 [19] contains about 700 references, documenting optimization as well as approximation methods proposed to solve the vehicle routing problem or its many variants.

Chapter 3

Scan Sequencing Models

A fundamental management issue in the use of space-based sensors is the sequencing of their scans, which are the individual images that they create so as to characterize an area or target of interest. An image can be thought of as a photograph, although that is a specific realization of an imaging product using visible wavelengths. Beginning with Landsat in 1972, and continuing through numerous other sensors in the visible and near-visible wavelengths, instruments have been used to image the surface of the earth. Since specific regions will be targetable only during relatively short time windows, it is essential that the selection and ordering of scans be done prudently. We will examine strategies for use by such instruments in either covering a region or acquiring a sequence of samples. To lend concreteness to the analysis, a description of a representative sensor will be provided. The one we discuss is a high resolution imaging spectrometer (HIRIS) intended for use in NASA's Earth Observing System. It should be noted that sensors will in many cases be mounted on board aircraft, and the scan sequencing strategies that we discuss here are equally applicable to them.

The problem of interest is as follows: given a target region, such as in Figure 3-1, what selection of scans is best and in what sequence should they be made? The area that is covered by an individual scan will have some characteristic pattern or shape that is dictated primarily by the technical parameters of the sensor, and to

a much smaller degree by the geometry between the sensor and the target. As an example, the basic shape of an instantaneous HIRIS image is a rectangle that is 30 meters deep in the direction of the satellite's ground track (this is called the down- or along-track direction), and 24 kilometers wide in the direction orthogonal to the ground-track (the cross-track direction). These dimensions are the exact pattern for when the sensor is pointed directly at the ground beneath the satellite, which is called the satellite's nadir; when the sensor is pointed $\pm 45°$ cross-track, the width of the rectangle is 60 kilometers wide. Figures 3-2 and 3-3 show possible alternatives for the scans that are selected to cover the area in Figure 3-1; we assume that the scan shapes are the same in both figures. The scans are graphically represented in these figures using a rectangle, which is intended to convey the general sense of scan planning for HIRIS, but that is obviously not drawn to scale. The example in Figure 3-2 has fewer scans than the example in Figure 3-3, while the latter example requires less lateral "maneuvering" of the sensor. This identifies one trade-off, i.e., between the number of scans taken and the repositioning of the sensor between scans. The relative importance of these two criteria depend on the specific sensor. We will describe the technical characteristics of this trade-off for the HIRIS sensor in section 3.1.

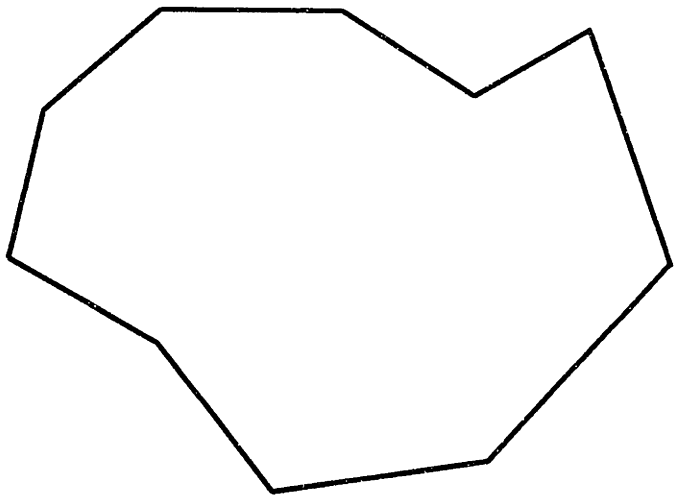

Figure 3-1: Representative region to be imaged.

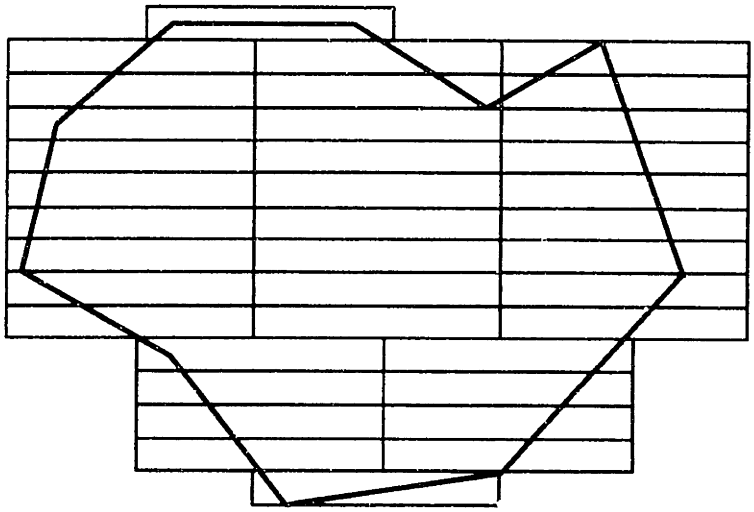

Figure 3-2: Scanning strategy for region in Figure 3-1.

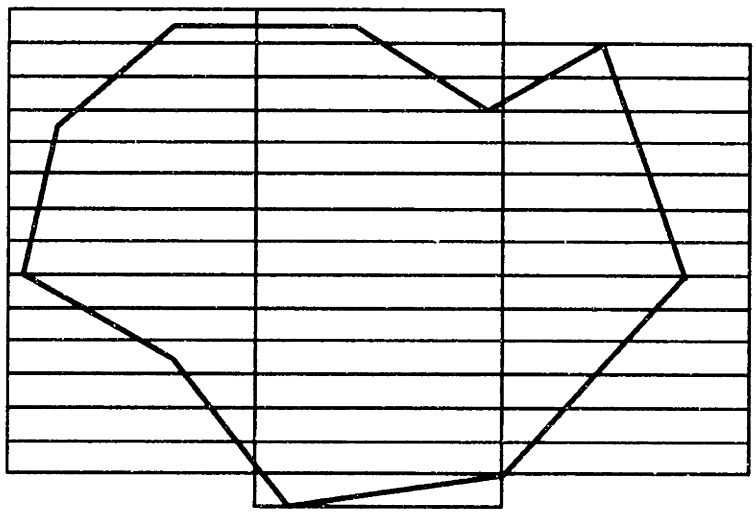

Figure 3-3: Alternate scanning strategy for region in Figure 3-1.

3.1 HIRIS

For a specific instrument to motivate our analysis, we consider the high resolution imaging spectrometer (HIRIS) [36], [53], [83]. It is an Eos facility instrument that is planned for the Eos-A2 and -A3 platforms, which are scheduled for launch near the turn of the century. If included, they will provide the capability to acquire high spatial and spectral resolution observations over at least a decade. Such measurements are not currently possible with existing ground-based or spaceborne instruments. A detailed description of the motivation, science objectives, and technical design of the HIRIS sensor is provided in Appendix A. For completeness, however, we describe here the salient technical characteristics that drive our modeling of image sequencing for space (and airborne) based sensors.

3.1.1 Instrument Design

HIRIS is designed to acquire simultaneous images in 192 spectral bands in the dominant wavelengths of the solar spectrum, 0.4 to 2.5 microns, at an approximate sampling interval of 10 nanometers. The ground instantaneous field-of-view will be 30 meters over a 24 kilometer wide swath. What this means is that an individual picture element, or pixel, in an image is 30 meters square. The 24 kilometer wide swath consists of 800 such pixels side by side. Figure 3-4 depicts an instantaneous HIRIS scan.

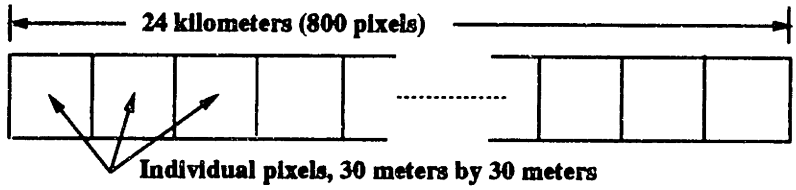

Figure 3-4: An instantaneous image from the High Resolution Imaging Spectrometer.

To make the ensuing discussion of HIRIS more understandable, we provide a small

glossary of some of the terms commonly used in describing the technical characteristics of satellite or airborne based surface imagers.

Term	Definition
along-track	The direction in which the satellite moves
cross-track	The direction that is orthogonal to along-track
integration time	The time over which the detectors measure solar radiation within a pixel
instantaneous-field-of-view (IFOV)	The angle that determines the smallest distinguishable area that can be delineated
ground-IFOV	The surface distance that the IFOV produces
field-of-view (FOV)	The angle that determines the size of the region that is characterized in a single image
resolution	The precision with which an item or activity is measured or performed
settling time	The time it takes the sensor to settle down after having been repointed
slew-rate	The speed with which the sensor is repointed
stability	The accuracy with which the sensor can be held still
swath	The width of an image (determined by the FOV)

Table 3.1: Glossary of terms used to describe sensor capabilities.

The "ground-instantaneous-field-of-view" of 30 meters is the ground distance over which a measurement is made of reflected solar radiation. The sensor may be used to scan a swath at either the spacecraft ground-track speed of 6.83 kilometers/second or at reduced effective ground speeds of $\frac{1}{2}$, $\frac{1}{4}$, or $\frac{1}{8}$ of the satellite ground-track speed, which is produced by a compensating scan of the along-track mirror. This increases the effective pixel integration time, and thus improves the signal-to-noise ratio when imaging dark targets; this will be especially useful for water studies. Figure 3-5 [89, Figures 2d and 33] shows the general geometry and concept of imaging spectrometry using area array detectors. Physically, the way that the spectrometer works is that incident radiation passing through a slit at the front of the fore-optics is collimated, separated into two bands, and then dispersed and reimaged onto separate detector arrays for visible and near-infrared wavelengths, and for shortwave infrared wavelengths.

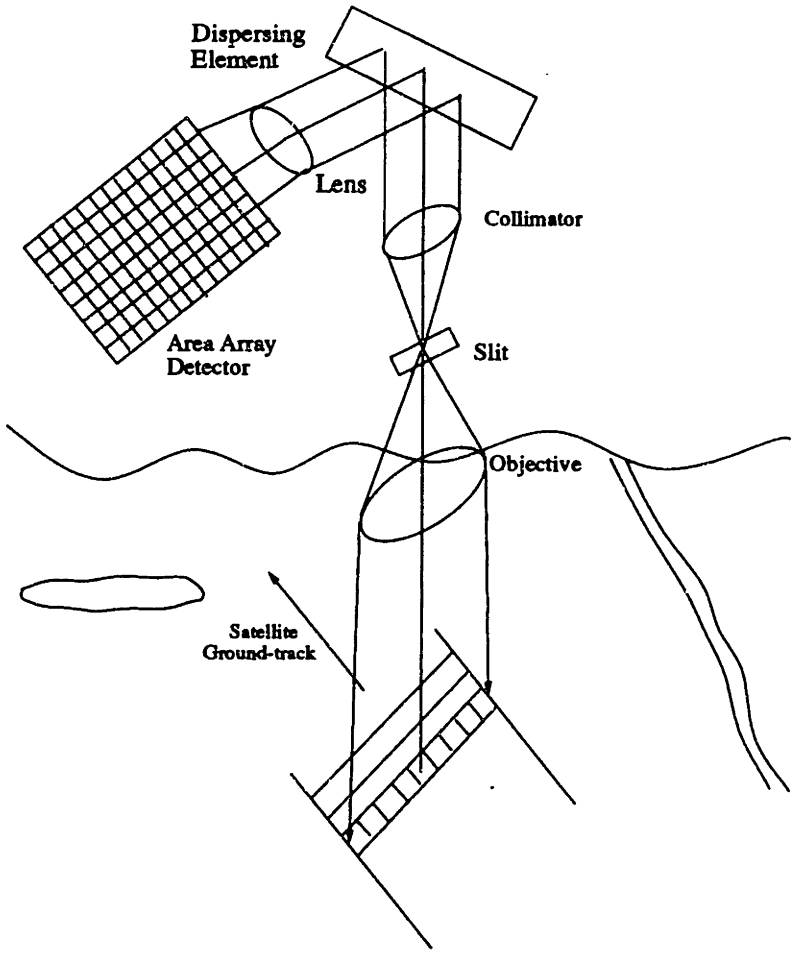

Figure 3-5: Imaging spectrometry concept.

The center of the spatial swath line may be placed anywhere within the instrument pointing capabilities that are given in Table 3.2 [53, p. 141]. Figure 3-6 shows the dimensions of the overall area within which HIRIS can acquire data based upon these pointing parameters. To give a sense of proportion, a 96 kilometer by 96 kilometer scan box is shown roughly to scale; this is about the largest size of area that can be imaged in a single pass of the satellite.

Parameter	Along-track	Cross-track
Total Range	$+60°/-30°$	$\pm 45°$
Pointing Resolution	.070°	.070°
Stability (jitter)	0.74 arc-sec (0.1 pixel)	0.74 arc-sec (0.1 pixel)
Slew Rate (max)	5°/sec	2°/sec
Settling Time (after slew)	0.5 sec	0.5 sec

Table 3.2: HIRIS pointing parameters.

In operation, the image from a line on the ground is spectrally distributed over a detector array. At the end of one line integration period the photon-generated charge on each line is shifted into a storage area, and then shifted into registers that are shifted out and read through amplifiers. The integration time corresponding to the 6.83 km/s satellite ground-track speed and 30 meter pixel size is 4.39 milliseconds. For an 800 pixel swath width and 192 spectral bands digitized to 12 bits, the corresponding data rate is about 410 Mbps. The amount of time that it takes to shift out and then read the charge that has been generated takes several orders of magnitude less time than a single integration period. Thus the instrument can essentially be turned on and allowed to run continuously, taking and recording a large number of contiguous along-track images, so long as it is not repointed between images. This does not take into account, however, limitations on the allowable data rate.

Performing a scan consists of pointing the sensor at a spot on the ground and then "turning it on." After 4.39 milliseconds (or the appropriate multiple of 4.39; longer scan times provide a better signal-to-noise ratio) the photon charge that has been generated while the sensor moves along its ground-track is processed as described

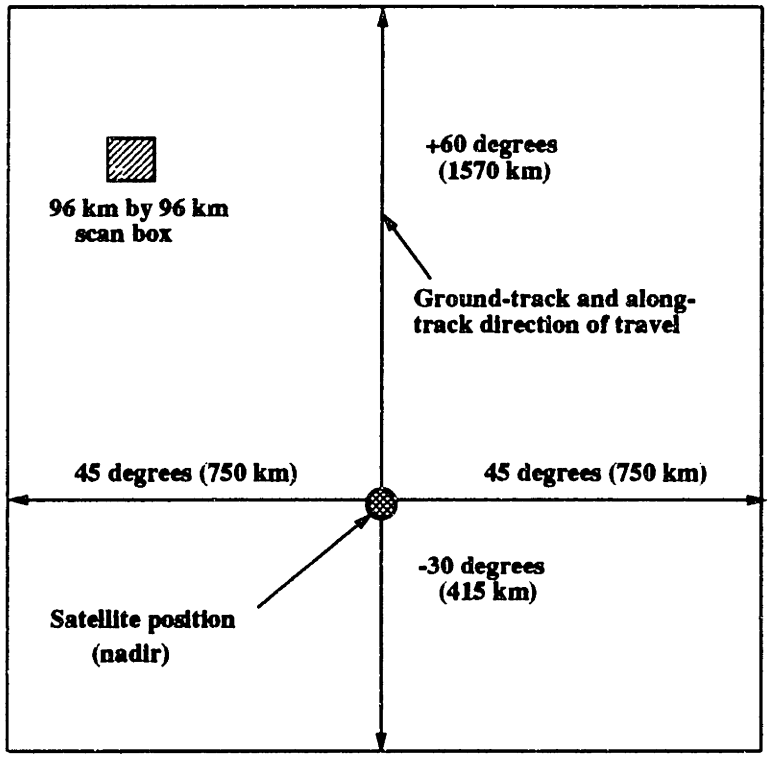

Figure 3-6: Region within which HIRIS can acquire targets, based on its pointing capabilities.

above. To model scannning, we represent an individual scan with a point located in the middle of the rectangular sensor ground pattern. The point is symmetric with respect to the distance traveled during a pixel integration period, as well as with respect to the scan swath.

3.1.2 Current Planning

Many science users are interested in imaging regions that are very small, and for which the best sequence of scans is obvious. One major exception to this is the cloud researchers, whose goal is to scan areas that are quite wide in the across-track direction. *Because of the amount of time that it takes to reposition the sensor*, planners think in terms of placing "boxes", which consist of scanning continuously in an along-track direction. This is often done for 800 along-track scans, creating a 24 km by 24 km square—800 × 800 pixels. Such a scan sequence takes approximately 3.5 seconds. A 24 km wide by 96 km long scan region would take roughly 14 seconds, and a 96 km by 96 km region would take about 56 seconds. The 56 seconds for the 96 km × 96 km region does *not* include the time needed to reposition the sensor between along-track scans. *Planning figures for sensor maneuvering, as shown in Table 3.2, are 2°/sec across-track and 5°/sec along-track; after moving the sensor there will be about a half-second settling time.* The strategy employed to image a 96 km × 96 km region is to point the sensor well ahead of nadir to begin the first 96 km long scan line, and then to move it over and back for each scan line as needed, until the desired number of scan lines have been imaged. Repointing and sensor settling between each scan line will take approximately 3.5 seconds, producing a total scan time for a 96 km box of approximately 66 seconds. As has been noted, with the detector readout time being considerably less than the pixel integration time, along-track scanning for many instantaneous images is essentially continuous.

3.1.3 Management Issues

As currently envisioned, the scheduling of HIRIS for imaging a particular region is fairly restricted. The amount of time needed to repoint the sensor from one location to another, plus the settling time afterward, dictates that as little of these movements be incurred as is possible. Even at the expense of covering areas that are of little or no interest, it is clearly advantageous to identify a single (or several) long scan line(s) and then continuously collect data. Spatial editing can be used to filter out information that is not needed; along with spectral editing and pixel averaging this will be used to reduce the data rate to a manageable level. Nonetheless, instrument repointing will be necessary both to move from one region to another for different observation applications, and to implement sampling strategies within a specific area that is of interest.

Potential scan sequencing strategies for airborne imaging spectrometers (and other sensors) are much broader, though. An aircraft is certainly not as severely restricted in its movements as is a satellite. Extensive pointing isn't needed since multiple views of a location can be obtained by another overflight, although some off-nadir viewing capability is necessary to compensate for atmospheric effects and to measure different reflectance characteristics of specific phenomena. Future generations of space-based sensors are likely to be more responsive, and hence the strategies that we consider capture a wider range of sequencing options than will strictly be permitted for HIRIS.

3.2 Problem Formulation

Many projects are likely to require far more than a single 24 km by 30 m scan, thus the following problem is posed:

> Given a region that is to be imaged, what selection of scans and in what sequence will minimize the amount of time necessary to cover the desired area?

Note that in its most general form, the region to be imaged for a particular project need be neither contiguous nor convex. This problem really has two coupled parts:

1. Determining what scans produce the optimal partition or cover of the targeted area.

2. Sequencing the scans.

The two parts are coupled in that the scan selection that ultimately minimizes the processing time might not be a minimum cardinality partition or cover. In a partition the scans would be non-overlapping yet still contain the desired area (they would be both a packing and a covering). A certain amount of "dispersion" of the scan points would result, even if the region partitioned were convex. In addition, though, areas that are of little or no interest might be imaged. A cover of the region of interest, however, would tend to minimize the dispersion of the scan points. Overlapping scans would result, and duplicate measurements could be either discarded or used for comparison purposes. Moreover, the cover that produces the scan sequence requiring the minimum amount of time to complete might intentionally incur greater overlap than is necessary—because it requires less repointing of the sensor. This is an important issue, as was noted in section 3.1.2, when we described current planning for HIRIS. A single instantaneous image requires only 4.39 ms, whereas repointing is at best as fast as 5°/sec, with roughly .5 seconds required for settling after the sensor is moved. At the present platform altitude of 705 km, it will take approximately 975 milliseconds to move the surface point at which the sensor is targeted 24 km cross-track ($\frac{2\times\arctan(\frac{12\text{ km}}{705\text{ km}})}{2°/\text{sec}} \approx 975$ ms). Thus any reduction in the repointing that occurs as a consequence of an increased number of scans is likely to result in a favorable trade-off in terms of the overall time that is needed to complete the desired mapping. So although the objective of minimizing the time it takes to image an area cannot truly be optimized if the two constituent subproblems of selection and sequencing are separated and addressed individually, the preponderant impact of

sequencing in comparison to selection supports an approach in which they they are considered separately for modeling tractability.

We have assumed that scans constituting either a cover or a partition of a region are required; this need not be the case. If the region is very large yet measurements characterizing it must be made within a single time window, then some sort of sampling strategy appropriate for the particular scientific use will have to be employed. While such sampling strategies are of interest in their own right, they are beyond the scope of this thesis. Scan sequencing, though, would still remain an issue.

Even if the number of scans to be made were known, the set of locations at which they could be placed is of infinite cardinality. From a practical perspective there are several reasonable approaches that can be used to select both the number and location of HIRIS scans, given the sensor's 24 km wide by 30 meter deep scan swath. A straightforward heuristic can produce a near-optimal cover of the target area by simply overlaying the contiguous regions with 30 meter deep (in the along-track direction) non-overlapping strips. Within each strip (in the cross-track direction), the requisite number of 24 km wide scans can be placed; these can either overlap or be non-overlapping. If the final scan were to cover area that was not of interest, then those pixels could simply be discarded as noted earlier. This is explained in greater detail when we discuss lower bounds for scan sequencing in section 3.6; Figure 3-11 displays the heuristic in a diagram.

If we assume that we already know the optimal scan cover, the remaining problem of sequencing the scans has the flavor of a traveling salesman problem. The traveling salesman problem, or TSP, is a classic combinatorial optimization problem that is easy to state. If a salesman starts from his home and is to visit each city on a given list exactly once and then return home, he will want to order the cities such that the distance that he has to travel is minimized. It is assumed that for each pair i, j of cities, the salesman knows the distance c_{ij} between the cities. There is a large and rich literature dealing with a wealth of applications and analysis approaches that

are related to this problem; a book edited by Lawler, Lenstra, Rinnooy Kan, and Shmoys [68] cites over 400 such references.

Scan sequencing is most similar to a combination of special cases of the TSP. A standard special case is the symmetric TSP, which is restricted to only those instances where $c_{ij} = c_{ji}$ for all cities i and j—this is typically what people mean when they refer to the 'TSP.' Another way to restrict the general TSP is to require that the distances between cities obey the *triangle inequality*: $c_{ij} + c_{jk} \geq c_{ik}$ for all i, j, k. An important special case of the symmetric TSP with triangle inequality is the Euclidean TSP. The cities are given as points with coordinates in the two-dimensional plane, and their distances are computed according to the Euclidean metric. A final special case of the TSP that is relevant here is the 'Wandering Salesman' problem. It is the same as the TSP, except that the salesman can start his tour in any city, and does not have to return to the first city at the end. The ordering of scans is much the same in that there is no particular spot at which the imaging must begin or end.

In the case of scan sequencing, "distances" are symmetric if each movement of the sensor between scans entails repointing. When scanning is done continuously along-track, the sensor does not have to be repointed between instantaneous images, and no sensor maneuvering or settling time is incurred. We assume pairwise sensor movement times to be symmetric by considering strategies in which we constrain ourselves to placing scan rectangles consisting of some predetermined number of consecutive along-track scans. The true calculation of repointing times, even with this assumption, is an involved computation that must account for the orbital dynamics of the satellite—which will cause *forward* movements of the sensor scanning mirror to be faster than *backward* movements, since the satellite is itself moving.

Sensor movement times are computed by a metric that that can be specified as a function of the L_2 distance between scan points in Euclidean three-space. The time it takes the sensor to travel between two scan locations is determined by the angle between them as perceived by the sensor at its orbital position (the sensor slew rate

was given in degrees per second in Table 3.2). If the locations of all three points are known, then the angle can be computed using the law of cosines:

$$c^2 = a^2 + b^2 - 2ab\cos\theta\,, \tag{3.1}$$

where θ is the angle opposite the side of length c. While the triangle itself is a two-dimensional body, the lengths a, b, and c are computed from the three dimensional locations of each vertex, with the two scan points in question being located on the surface of the earth.

Instead of calculating the precise angle subtended by two scan points, we instead use the Euclidean distance, scaled appropriately. This follows from an approximation: we assume that the Euclidean distance between two scan points is comparable to the *great circle* distance between them on the surface of a sphere whose radius is the satellite's orbital altitude. The great circle distance between two points on a sphere is the geodesic distance that comes from traversing the surface between them, and equals $r \times \theta$, where r is the radius of the sphere and θ is the angle subtended by the two points in question. This results in the following approximation for the angle between scans:

$$r\theta \approx d(s_1, s_2) \Longrightarrow \theta \approx \frac{d(s_1, s_2)}{r}$$

where s_1 and s_2 are two scan points, $d(s_1, s_2)$ is the L_2 distance between them, and r and θ are defined as before. Since computing the sensor movement *time* entails only including another scale factor (the sensor slew rate), the Euclidean metric provides an adequate proxy for the transit time.

In reality, the scan points are on the surface of what is essentially a planar region—the surface of a *large* sphere (the earth). Since the scan points for a regional imaging project are relatively close to one another, the error introduced by this approximation is minor. The error due to further assuming that the points are on the surface of a sphere centered at the satellite will be greater, but still relatively small, again because

we are assuming that the scan points are clustered together in an area, rather than distributed over the surface of the sphere. Given these considerations, the use of the scaled L_2 distance is a reasonable approximation. Bounds on the size of the difference that these approximations introduce as a function of the radius of the sphere at the center of which we assume the satellite is located are developed in section 3.4: *Computational Complexity*.

For more general imaging projects in which the sensor may move up to its design limits (see Table 3.2), this approximation cannot be used. It is well known that there exists no general isometric (distance-preserving) transformation from a sphere to the plane, although there are isometric transformations from the sphere to the plane from a single point. This eliminates the possibility of a transformation so that Euclidean geometry can be used to measure all spherical distances. Hence using the scaled Euclidean metric instead of an exact computation of the subtended angle is indeed an approximation, however, over the relatively small regions that are being considered the potential error is small. In particular, we will make use of this approximation in conducting computational experiments in section 3.8: *Computational Testing*.

The performance of the algorithms that we present is independent of the metric, so long as the one used is both symmetric and obeys the triangle inequality. The physical mechanics of moving the sensor may actually result in either an L_1 or L_∞ metric equivalent, rather than L_2. And in reality, the calculations aren't quite this simple since, as has been noted, the satellite continues to move while the sensor is being repointed and the computations must address orbital dynamics.

3.3 Literature Survey

3.3.1 Scan Selection

The problem of scan selection is one of a larger class of problems that deals with partitioning or covering a geometric figure with a minimum number of more fundamental

geometric figures. Applications arise in VLSI artwork data, image processing, and architectural databases. Triangles, convex polygons, and rectangles have been chosen to be the fundamental figures. In most cases these problems have been shown to be $\mathcal{NP}$-complete, although there are some polynomial time algorithms for specific cases. One such problem is that of finding a minimum size collection of rectangles, which are allowed to overlap, whose union is a polygon. An application of this problem is to efficiently create masks for photolithography, using a pattern generator which is constrained to print rectangles [22]. An algorithm for constructing a minimum rectangle cover when the region in question is vertically convex is presented in [46]. It is quadratic in the number of vertices of the polygon.

Ohtsuki [86] gives an $O(n^{5/2})$ algorithm for partitioning a rectilinear polygonal region (which may contain holes) into a minimum number of rectangles, where n is the number of vertices of the polygon. Asano and Asano [1] consider the problem of partitioning a polygonal region into a minimum number of trapezoids with two horizontal sides, and include triangles as degenerate trapezoids. In both cases the motivation is again VLSI photolithography. In such systems the circuit layout is stored as a set of polygonal regions per layer. These regions are bounded by straight lines that may have any slope. Processing time is proportional to the number of regions in the cover/partition, and thus a minimum number of figures is sought.

In comparison to the work just described, the problem of scan selection that we consider is highly constrained: the cover/partition of the polygonal region being imaged is to be constructed of uniformly shaped rectangles, not ones of variable size. The heuristic already described provides a minimum cardinality cover for rectangles that are the same shape as the focal plane image of the HIRIS sensor. An optimal cover of the region to be scanned is less obvious when we consider the linkage between selection and sequencing, but, as has been noted, scan selection is relatively unimportant in the overall context of the sensor scheduling process. For our purposes, we can effectively use even simpler algorithms to determine a cover or partition of a target

region than are provided by the polynomial approaches described in the work above.

3.3.2 The Traveling Salesman Problem

Scan sequencing has the character of a traveling salesman problem, for which there is a vast literature. At the present time TSP's with several hundreds of nodes can be solved routinely to optimality using methods that exploit the facial structure of the feasible polytope and branch and bound techniques. Currently the largest problem solved to optimality has 2392 cities and was done using a branch and cut algorithm [88]. Because the TSP is $\mathcal{NP}$-complete, considerable effort has been devoted to approximation algorithms that produce high quality, if not optimal, solutions. There is a wide spectrum of such approaches that include insertion heuristics [95], local search exchanges [72], simulated annealing [21], [54], threshold accepting [37], tabu search [52], neural networks [47], simulated tunneling [98], and genetic methods [81]. A book edited by Lawler, Lenstra, Rinooy Kan, and Shmoys [68] is devoted to the TSP, and it provides many references. Its first chapter gives the history of the problem and shows its role in several disciplines. A more recent survey of TSP algorithms that use local optimization is provided by Johnson [64].

Instead of attempting to review the entire body of literature for the TSP, we selectively identify references that deal with specific algorithmic approaches that we either use or adapt for our purposes in analyzing scan sequencing. Since our final goal in scan sequencing is a shortest Hamiltonian path, any TSP heuristic that we use must be adapted appropriately. Our work is similar to the intersection of a number of special cases of the TSP, the most salient of which is the geometric, or Euclidean, TSP. Heuristics for geometric traveling salesman problems are discussed in Bentley [16] and Reinelt [92]. Geometric versions of the TSP are still $\mathcal{NP}$-hard [48].

Two of the heuristics that we use are tour building heuristics. The nearest neighbor TSP heuristic starts with a partial tour consisting of a single, arbitrarily chosen city. It then successively visits the closest unvisited point, and it ultimately adds

an arc from the last city to the first city, completing the tour. This is an approach that has long been employed in the analysis of the TSP and that is not attributed to a specific researcher; Flood [44] refers to this approach as the "next closest city method." Analysis of this heuristic [95] has shown that the length of a geometric nearest neighbor tour can never be more than $(\lceil \log n \rceil + 1)/2$ times the length of the shortest tour. Another approach that grows a tour is the multiple fragment heuristic, in which edges of the underlying graph are successively selected so long as they do not create cycles or nodes of degree three. Steiglitz and Weiner [106] consider an approach similar to this, while Bentley and Saxe [15] analyze its performance. Its worst case length for n points in the unit square is $O(\sqrt{n})$.

The Minimum Spanning Tree (MST) heuristic first finds the minimum spanning tree of the cities, and then traverses the nodes of the tree in a depth-first, inorder traversal to produce a tour. This tour can be no more than twice the length of the optimal tour, since the MST itself is a lower bound and the MST tour is at most two times the length of the MST. This heuristic is well known; an elaborate version was published in [95]. Christofides' heuristic [25] improves upon the minimum spanning tree by computing a matching of all vertices that have odd degree, and then finding an Eulerian tour through the graph that combines the matched edges with those from the MST.

The first local optimization heuristic that we employ is the straightforward 2-Opt heuristic, which reverses a subsequence of the tour. A 2-Opt swap is referred to as an "inversion" in [31]; in [44] it is described as removing intersections. A 2-Opt algorithm can be implemented in many ways; we follow the approach outlined in [16]. More powerful than 2-Opt is what is referred to as Two-and-a-Half-Opt swap, or 2H-Opt. Bellmore and Nemhauser [12] describe this heuristic by viewing a tour as a sequence of points, and 2H-opt moves a single point from one place in the sequence to another. These heuristics are described in greater detail in section 3.7.4: *Local Optimization*, and diagrams depicting how they work are shown in Figures 3-15 and 3-16.

One of the best lower bounds for the traveling salesman problem is provided by what is called the 1-tree problem with Lagrangian objective function. This relaxation has been used successfully by a number of researchers [59], [24], [58], [105], [111]. For the complete undirected graph $G = (N, A)$, the relaxation consists of finding an n arc connected spanning subgraph of G that minimizes the total distance traveled. Such a subgraph consists of a spanning tree plus one more edge. This can be further restricted by specifying a node, say 1, that is to be of degree 2 and that is to be in the unique cycle that the n arc subgraph contains. It is easily seen that this is a relaxation of the original TSP. Mathematically, the constraints defining a 1-tree can be stated as [68, p. 371]:

$$\sum_{i\in S}\sum_{j\in N'-S} x_{ij} + \sum_{i\in N'-S}\sum_{j\in S,\, j>i} x_{ij} \geq 1, \ \forall S \subset N' = N - \{1\},\ S \neq \phi \quad (3.2)$$

$$\sum_{i\in N}\sum_{j>i} x_{ij} = n \quad (3.3)$$

$$\sum_{j\in N} x_{1j} = 2 \quad (3.4)$$

$$x_{ij} \in \{0,1\}, \quad (3.5)$$

where x_{ij} is 1 if an arc from node i to node j is included, and 0 otherwise. Equation (3.3) is a relaxed subtour breaking constraint that insures that the tree is connected; (3.4) requires n arcs to be chosen; and (3.5) makes node 1 the arbitrarily selected vertex.

Finding the minimum length 1-tree defined by (3.3)–(3.5) decomposes into first finding the MST defined over the cities$-\{1\}$, and then finding the two smallest cost arcs among those incident to node 1. There are many 1-trees contained in the complete undirected graph induced by the cities of a TSP, and computational experience with randomly generated symmetric problems found the value of an optimal 1-tree to be only about two thirds that of the optimal solution. The relaxation is much stronger,

though, if the constraint

$$\sum_{j<i} x_{ji} + \sum_{j>i} x_{ij} = 2, \quad \forall\, i \in N \tag{3.6}$$

from the integer programming formulation of the TSP is taken into the objective function in a Lagrangian dual; equation (3.4) in the 1-tree is the sum of all equations (3.6) divided by 2. The Lagrangian is then maximized as a function of the multipliers. Computational experience with this approach has produced lower bounds that are greater than 99% of the optimal solution [26], [111]. The Lagrangian dual problem is typically solved by subgradient optimization, a general description of which is in [60] or [101].

3.4 Computational Complexity

Several results on the complexity of problems related to the ones considered here have been noted. In the area of scan selection, the general problem of partitioning a geometric figure into a minimum number of more fundamental ones has been shown to be $\mathcal{NP}$-complete in many cases [63], [73]. On the other hand, there are polynomial-time algorithms for partitioning specific figures (e.g., polygonal regions) into other forms [22], [46], even if the figures contain holes [1], [86]. The complexity of the specific covering/partitioning done here, however, isn't really an issue since relatively simple (and fast) heuristics are capable of providing very good covers in the context of scan selection *and* sequencing. As a result, we focus on the complexity of scan sequencing.

If we assume that we already *know* the optimal scan cover, or at least have identified one that is acceptable, then the remaining problem of sequencing the scans is similar to a geometric traveling salesman problem. The geometric traveling salesman problem [48] is derived from the original traveling salesman problem, both of which we will state [49, p. 211-212]:

TRAVELING SALESMAN PROBLEM (TSP):

Instance: Set C of m cities, distance $d(c_i, c_j) \in Z^+$ for each pair of cities $c_i, c_j \in C$, positive integer B.

Question: Is there a tour of C having length B or less, such that

$$\left(\sum_{i=1}^{m-1} d(c_{\pi(i)}, c_{\pi(i+1)})\right) + d(c_{\pi(m)}, c_{\pi(1)}) \leq B \ ?$$

GEOMETRIC TRAVELING SALESMAN PROBLEM (GTSP):

Instance: Set $P \subseteq Z \times Z$ of points in the plane, positive integer B.

Question: Is there a tour of length B or less for the TRAVELING SALESMAN instance with $C = P$ and $d((x_1, y_1), (x_2, y_2))$ equal to the discretized Euclidean distance $\lceil ((x_1 - x_2)^2 + (y_1 - y_2)^2)^{1/2} \rceil$?

A tour is a permutation $< c_{\pi(1)}, c_{\pi(2)}, \ldots, c_{\pi(m)} >$ of the cities in the set C. The transformation in the $\mathcal{NP}$-completeness proof of the GTSP is from EXACT COVER BY 3-SETS (X3C) [48]. Note that this problem is strongly $\mathcal{NP}$-complete, and remains strongly $\mathcal{NP}$-complete if the distance measure is replaced by the L_1 "Manhattan" metric or the L_∞ metric. If the non-discretized Euclidean metric is used, the problem remains strongly $\mathcal{NP}$-hard, but is not known to be in $\mathcal{NP}$ [49, p. 212].

It can be shown that the optimal scan sequencing problem, which we refer to as OSSP, is $\mathcal{NP}$-complete by transformation from the geometric traveling salesman problem. A "scan" is a point on the surface of the earth, which for all practical purposes is a planar region. In the $\mathcal{NP}$-completeness proof we show how points in an instance of the GTSP (which is specified for the planar lattice) transform to scan points on the surface of a sphere whose radius is the orbital altitude of the satellite. Assuming that the scan points are on such a sphere allows us to employ a scaled discretized Euclidean distance for the scan time metric:

$$\alpha\theta \approx \alpha \frac{\lceil d(s_1, s_2) \rceil}{r} \ , \tag{3.7}$$

where α is the sensor slew rate that converts the angle θ subtended by the scan points to an actual *time*, and r is the satellite's orbital altitude. In the proof, the conditions such that the Euclidean metric in the GTSP can be polynomially transformed to this approximation are established. The cities in the GTSP also transform to scan points on the surface of the earth, providing the link between points on the surface of the satellite centered sphere and points actually on the surface of the earth. As before, a sequence of scans is a path in the induced complete graph.

We state the recognition version of OSSP as follows:

OPTIMAL SCAN SEQUENCING PROBLEM (OSSP):
Instance: Set Q of n scans on the surface of the earth, sensor slew rate α, and positive integer T.
Question: Is there a sequence of the scans in Q that can be performed in time less than or equal to T, such that

$$\sum_{i=1}^{n-1} t(s_{\pi(i)}, s_{\pi(i+1)}) \leq T \ ,$$

where $t(s_i, s_j) = \alpha\theta$ is the pairwise transit time between scan points s_i and s_j?

As above, α is the sensor slew rate, θ is the angle subtended by s_i and s_j, and $< \pi(1), \ldots, \pi(n) >$ is a permutation of the scan points.

Theorem 3.1 *The Optimal Scan Sequencing Problem is $\mathcal{NP}$-complete.*

PROOF: Given an instance of OSSP, it can easily be determined if a specified sequence can be completed in time less than or equal to T. Problem input includes a set Q of n scans, which are specified by giving three-dimensional coordinates on the surface of the earth for each point, as well as the sensor slew rate α. The angle θ subtended by each pair can be calculated by using the law of cosines (equation (3.1)), and then the transit time between them computed by multiplying θ by α. Determining the total

scan sequencing time entails adding the transit times between successive locations in the sequence, which is then compared to the given integer T. Thus OSSP has the succinct certificate property and is in $\mathcal{NP}$.

The transformation to OSSP is from the GTSP, using the discretized Euclidean metric. First, using an instance of the the GTSP, we show how to construct an instance of scan sequencing in which the scan points are on the surface of a sphere whose radius is the orbital altitude of a satellite. Conditions are established for the construction so that even though the points are on the surface of a sphere, the distances between them will *discretize* to be the same as those on the planar lattice. The observing satellite is taken to be at the center of the sphere, and conditions on the construction so that the approximation of equation (3.7) can be transformed to are found.

We start with an instance of the GTSP, which consists of a set $P \subseteq Z \times Z$ of points in the plane, and a positive integer B. Above the plane we construct a sphere of radius r that is tangent to the plane at the origin (0,0). The scan points for which we consider a sequencing will be the vertical projection of the points in P onto the sphere. By "vertical projection" we mean the point at which a line orthogonal to the plane and passing through a point $p \in P$ intersects the sphere. Figure 3-7 displays an example and several other elements of the construction.

The length of the pairwise distances between points on the sphere will be *greater than or equal* to the corresponding distance between the original points in the GTSP. The original distances in the plane are the projection of the Euclidean distances between points on the sphere onto the plane: $d_{sphere} \times \cos\gamma = d_{plane}$. In Figure 3-7 d^{max}_{sphere}, d^{max}_{plane}, and γ are examples of d_{sphere}, d_{plane}, and γ. If the radius r of the sphere at which the satellite is centered is sufficiently large, then the length d_{sphere} will *discretize*, or round up, to $\lceil d_{plane} \rceil$. For this to hold, it is required that:

$$2 \times \lceil d^{max}_{plane} \rceil \geq 2 \times \left(\frac{d^{max}_{plane}}{\cos\gamma} \right) , \tag{3.8}$$

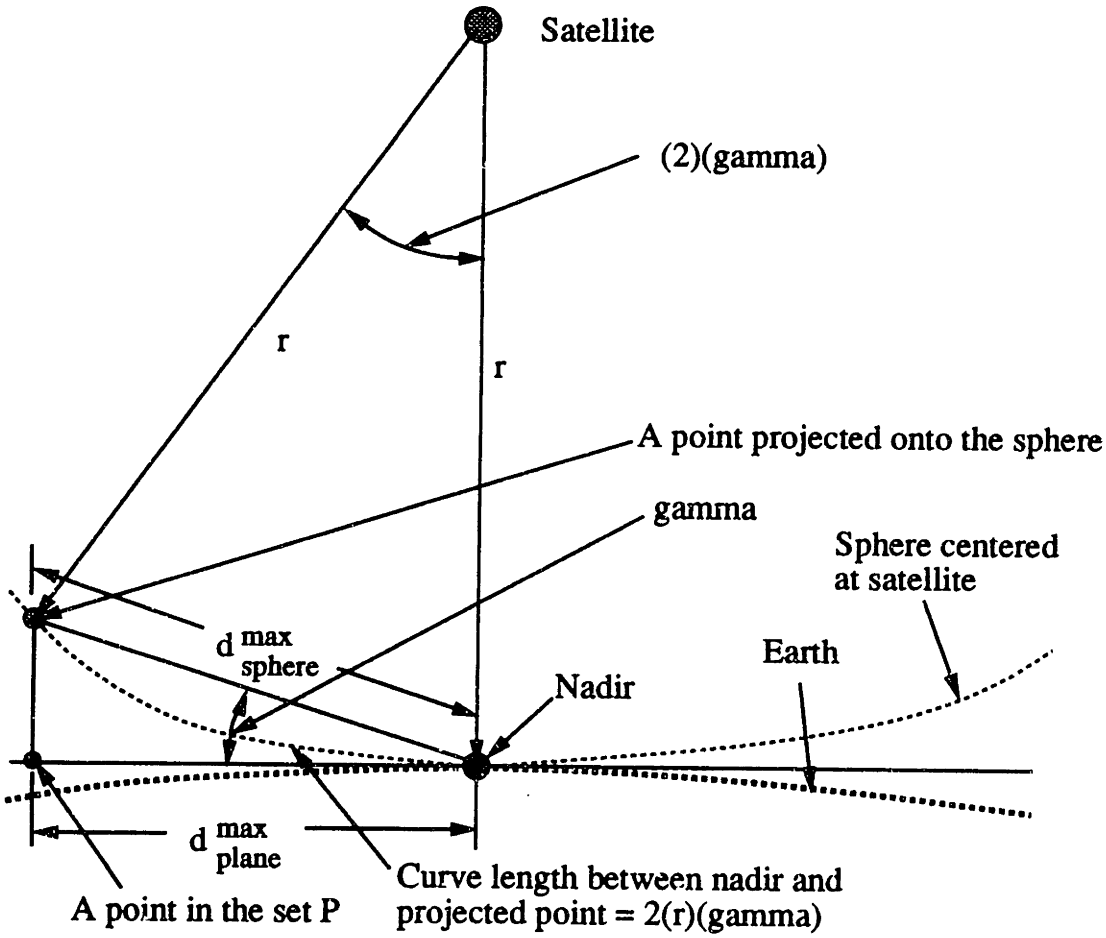

Figure 3-7: Construction for complexity proof.

where $\lceil d_{plane}^{max} \rceil$ is the maximum distance from any point in the GTSP to the origin (and thus twice that is an upper bound on the greatest pairwise distance), and γ is the largest possible "inflating" angle. γ is shown in Figure 3-7; the angle can be no greater than that subtended by arcs from the origin to the points producing the distances d_{plane}^{max} and d_{sphere}^{max}. Equation 3.8 produces the following condition for γ:

$$\gamma \leq \arccos\left(\frac{d_{plane}^{max}}{\lceil d_{plane}^{max} \rceil}\right) . \tag{3.9}$$

From planar geometry it follows that

$$r = \frac{\lceil d_{plane}^{max} \rceil}{2 \times \sin\gamma} \tag{3.10}$$

will be a radius sufficiently large such that the two L_2 distances discretize to the same value.

To approximate the discretized Euclidean distance between scan points with the great circle distance, a similar such bound and ensuing condition on γ must be established: the great circle distance can be no greater than the the discretized L_2 distance on the sphere.

$$2 \times \lceil d_{sphere}^{max} \rceil \geq r \times (4\gamma) \Longrightarrow \gamma \leq \left(\frac{2 \times \lceil d_{sphere}^{max} \rceil}{4r}\right) . \tag{3.11}$$

Note that Figure 3-7 shows the angle 2γ and L_2 chord d_{sphere}^{max}; for the bound we must use 4γ as the greatest possible angle, and $2 \times \lceil d_{sphere}^{max} \rceil$ as its subtending chord. Using equation (3.10) to set $r = \frac{\lceil d_{plane}^{max} \rceil}{2 \times \sin\gamma}$, the second condition that γ must satisfy is

$$\frac{\sin\gamma}{\gamma} \geq \frac{d_{sphere}^{max}}{\lceil d_{sphere}^{max} \rceil} . \tag{3.12}$$

Thus if we select a γ satisfying equations (3.9) and (3.12), then r as computed in (3.10) will be sufficiently large so that the great circle distance between scan points

discretizes to the planar L_2 distance. The transit *time* between scan points becomes

$$\alpha\theta = \frac{\alpha\lceil d_{plane}\rceil}{r}. \tag{3.13}$$

We now make a simple modification to the instance of optimal scan sequencing that has been constructed from the instance of the GTSP. It insures that if an optimal path producing algorithm is applied to the modified construction, it provides a path whose *length* is equal to the optimal tour length in the original instance of GTSP. The change consists of arbitrarily selecting a node, and "splitting" it into two nodes. The distances from these two points to all other scan points remain as they were for the original point; the distance between them is taken to be greater than $\max_{i,j} c_{ij}$, say Ψ. An optimal path will *not* contain the arc between the split nodes, since its length Ψ is greater than all others.

Now let $T = \frac{\alpha B}{r}$. It follows that there is an optimal tour among the points in the set P of length less than or equal to B if and only if there is a scan sequence for the set Q that can be done in time less than or equal to T. □

In practice, particularly when the exact metric for transit time is being employed, the discrete Euclidean distance will not be used. It is used here for the $\mathcal{NP}$-completeness proof, for otherwise interpoint distances could be irrational numbers and the distance matrix might require infinite precision. In addition, there is the problem of evaluating path lengths that are the sums of square roots. There is no known method to predict the number of significant digits needed in order to compare a sum of square roots to an integer. Thus it is not known if the version of OSSP with a non-discretized Euclidean metric is in $\mathcal{NP}$.

3.5 A Special Case of Scan Sequencing

As has already been noted, the time-savings that can be realized from intelligently selecting and sequencing scans has the potential to be quite large. Planning figures for HIRIS sensor slew rates (max) are 5°/sec along-track and 2°/sec cross-track, with a .5 sec after slew settling time. Since the sensor can essentially image continuously for scans that are contiguous and aligned in the along-track direction (*along-track aligned* scans lie on a line parallel to the satellite ground-track), a selection of scans that maximizes along-track alignment is desirable in comparison to a selection that incurs a greater amount of between scan repointing, especially in the cross-track direction. This can remain true even if along-track alignment requires more scans than does an alternative selection with more repointing. It follows that a sequence that generally minimizes "back-and-forth" sensor movement as opposed to "up-and-down" movement is likely to be considerably better. This observation leads to a special case of scan sequencing, for which the optimal sequence can be determined polynomially. Given the previous result on the complexity of OSSP, this can only be true if there are restrictions on the selection and placement of scans. Specifically, they must be non-overlapping, aligned in the along-track direction, and constitute a polygon-convex[1] region that is bounded. It is assumed that the sensor is capable of scanning along-track in either direction.

Theorem 3.2 *The optimal scan sequencing problem for a bounded polygon-convex region in which scans are constrained to be along-track aligned and non-overlapping can be solved as a weighted bipartite matching problem.*

PROOF: The bipartite matching that we solve is on the graph which is defined as follows. The nodes correspond to the top and bottommost scan points within a column of vertically aligned scans. Each node is adjacent (connected by an arc) to the

[1] A region is *vertically (horizontally) polygon-convex* if every column (row) of scan pixels in the polygon is connected.

analogously defined node in the left and right contiguous scan columns (see Figure 3-8). The left and right most (in a cross-track sense) scan columns are represented by a *single* node which is adjacent t_ both nodes in the adjoining column. The weights attached to each arc are the time that it would take to transit between the two end nodes. It is easily seen that this graph is two-colorable.

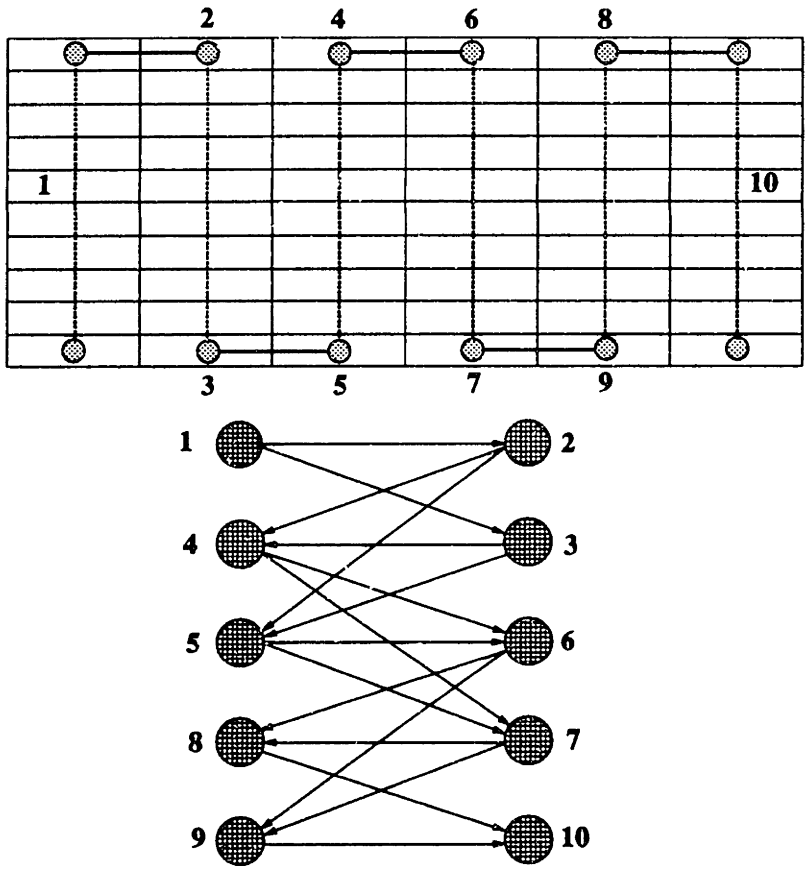

Figure 3-8: Special case of scan sequencing that reduces to bipartite matching.

If the along-track distance between paired nodes is appropriately bounded, then it will always be the case that for vertically aligned, non-overlapping scans it is optimal to transit the entire column without leaving it, i.e., departing only from the top or bottommost scan points and going to an analogous scan point in one of the two adjacent columns. Suppose, to the contrary, that it were optimal to leave a vertical scan column prior to one of the endpoints. At a minimum, it would be necessary to

return to the scan column departed, and so doing would incur the additional distance of the cross-track spacing between (non-overlapping) vertical scan lines.

Figure 3-9 shows the construction for the bound. The *time* needed to transit either a or b must be less than that needed to transit c, for otherwise, a strategy of leaving the middle of the long scan line to perform the scan on the left would be quicker than the strategy produced by the matching described above. The difference

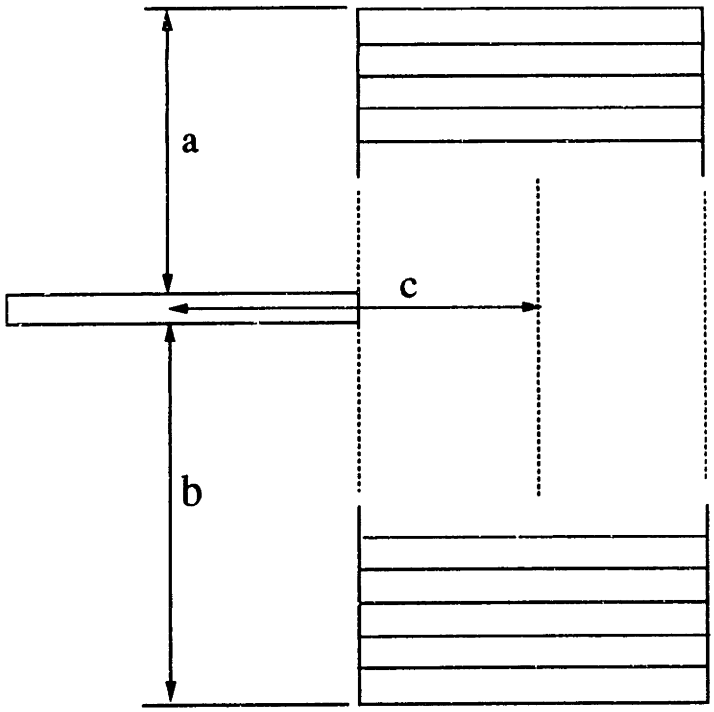

Figure 3-9: Relevant dimensions used in computing bounds on region size for the optimal sequencing of non-overlapping, along-track aligned, polygon-convex scans.

in along-track versus cross-track distances for a single instantaneous scan (a factor of 800 for HIRIS) allows for a large bound on the along-track distance between paired nodes. The bound serves to insure that the cross-track distance (c in Figure 3-9) that would be covered in moving *back* to a long scan column after having performed scans in an adjacent track is a greater distance than is any along-track sensor movement (a or b in Figure 3-9) over unscanned area that is incurred after departing a vertical scan line endpoint. If the cross- and along-track scan dimensions are α and β, and

the cross- and along-track sensor slew rates are γ and δ, then the bound is

$$\left(\frac{\alpha}{\beta}\right) \times \left(\frac{\delta}{\gamma}\right) \times \alpha \, .$$

Moreover, a shorter path cannot result from jumping to a non-adjacent column in a polygon-convex region, because it would then be necessary to backtrack so as to scan the jumped column. □

The optimal scan sequencing strategy for a polygon-convex coverage region that results from solving the bipartite matching is essentially to successively move up-over-and-down. Note that this is fundamentally the same strategy that NASA has employed in the simulations that they have done to date for HIRIS, although HIRIS cannot scan along-track in both directions. While long, continuous scan lines may not be ideal for many applications, the time saved by avoiding repositioning the sensor leads to consideration of scan strategies in which some minimum, predetermined number of instantaneous images must be made before contemplating any repointing that entails lateral movement to another location.

The bound on the distances a and b in Figure 3-9 such that the strategy of Theorem 3.2 will be optimal for HIRIS is 60 km:

$$\frac{24 \text{ km}}{30 \text{ m}} \times \frac{5^\circ/\text{sec}}{2\frac{1}{2}^\circ/\text{sec}} \times 30 \text{ m} = 60 \text{ km} \, ,$$

which represents 2000 along-track scans. Note that this bound means that *both* a and b cannot be greater than 60 km, and thus the length of the entire scan column cannot exceed 120 km. The conditions of polygon-convexity and vertical alignment are necessary, as can be demonstrated by simple counterexamples for which the strategy of Theorem 3.2 is not optimal. The necessity of solving a bipartite matching rather than simply comparing the lengths of the two alternating sequences in which one successively goes up-over-down can be seen in the example of Figure 3-10.

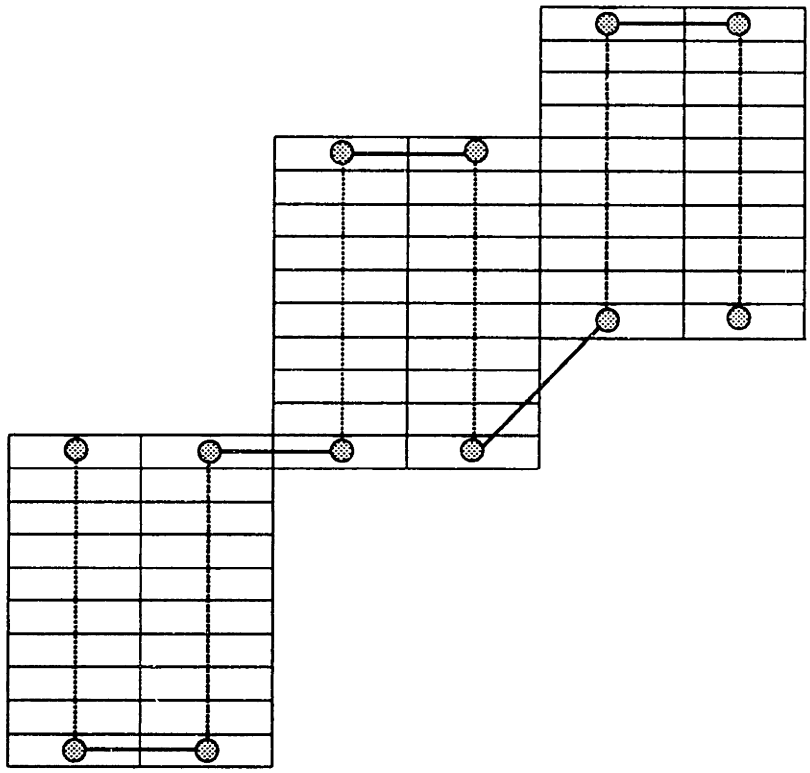

Figure 3-10: Example that shows need for bipartite matching.

As an alternative to bipartite matching, the scan sequence can be computed by dynamic programming. A backward recursion for computing the optimal sequence is:

$$v_i(s_i) = \min_{s_{i-1}} \{d(s_i, s_{i-1}) + v_{i-1}(s_{i-1})\}. \tag{3.14}$$

where

i = number of stages remaining, $i = 1, \ldots, n$

s_i = state of the system at stage n, $i = 1, \ldots, n$

$d(\cdot, \cdot)$ = L_1, L_2, or L_∞ distance metric

$v_i(s_i)$ = optimal value of all subsequent decisions, given that we are in state s_i with i stages to go, $i = 1, \ldots, n$.

Stage n represents the number of vertical columns remaining to be scanned; in each stage there are two states s_i, corresponding to the top or bottommost scans in that column. Whether we move from left-to-right or right-to-left does not matter. The overall computation will run in $O(n)$ time, since there are a constant number of computations and states for each stage. This is faster than solving the bipartite matching, for which an $O(n^{\frac{5}{2}})$ algorithm in [61] is the best known time bound. In both cases n is the number of nodes.

3.6 Lower Bounds for Scan Sequencing

Our objective here is as follows: *Given a region for which we do not know the best selection of scans, bound the time it will take to image the region.* As has been discussed previously, this can be separated into consideration first of bounds for the number of scan points, and then of the time needed to move the sensor between them. Although our analysis and algorithms are stated using the specific technical parameters of HIRIS, they are easily generalized to the characteristics of other sensors.

3.6.1 Bounds for the Number of Scan Points

A bound on the minimum number of instantaneous scans necessary to cover a region can be provided by the following construction.

Point Bound 1 (PB-1):

1. Let the y-axis coincide with the along-track direction of the sensor. Beginning with the scan point having the minimum y-coordinate, create 30 meter deep horizontal strips until the point with the maximum y-coordinate is contained within such a strip. The minimum and maximum y-coordinates are taken over all scan points. The *strip width* of the nth strip will be taken to be the length of the line forming its upper boundary.

2. Within each strip, place $\left\lceil \frac{\text{strip width}}{24\,\text{km}} \right\rceil$ scan points.

Figure 3-11 is an example.

The sum of the points needed for each strip will be a lower bound on the number of scans needed to cover the entire region. This is because the number of points that we compute for each strip need only be capable of covering the upper boundary. If we redefine the *strip width* we can get an upper bound on the number of points needed. Instead of using the length of the *upper boundary*, we compute the strip width as the distance between the two points within a given strip that have the minimum and maximum x-coordinates. Note that these two points need not have the same y-coordinate, as is the case for the previous definition of strip width. This definition of strip width insures that we cover any surface features that extend beyond the intersection points of successive boundary lines with the polygon exterior. Figure 3-11 shows the distinction between the two definitions of strip width.

While the difference between strip width definitions may not seem important when we are layering the scan region with 30 meter strips, it becomes significant if we consider a strategy in which we use layers that are longer in the along-track direction.

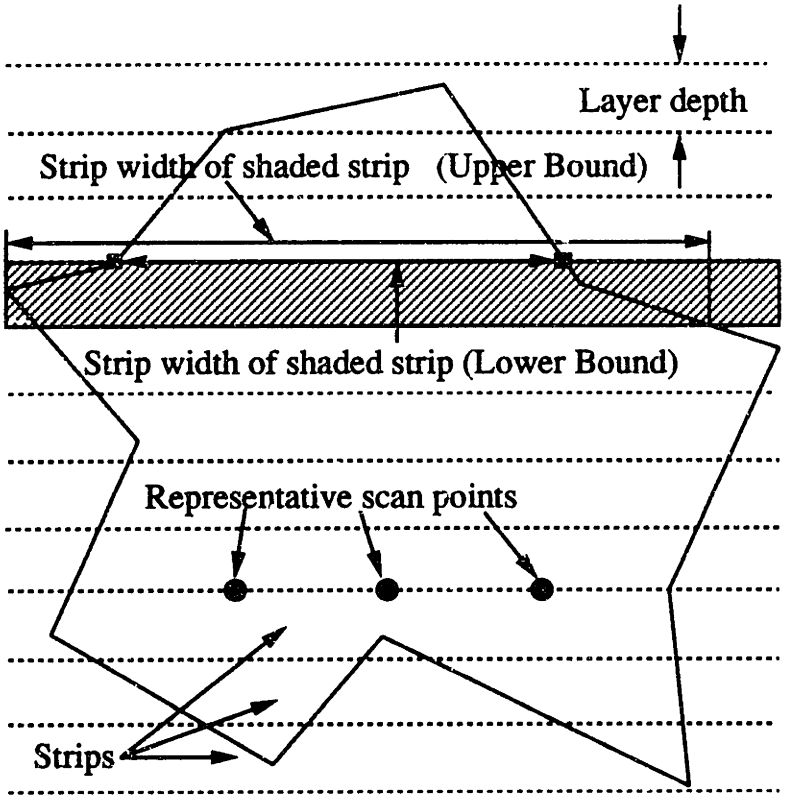

Figure 3-11: Polygon with horizontal layers.

Scan rectangles consisting of from 200 to 3200 consecutive instantaneous images will range in along-track length from 6 km to 96 km. Over these distances, it becomes much more reasonable to expect that terrain features within a given strip may extend beyond the arbitrarily defined upper boundary.

The positions of the scan points within a strip can be determined by a simple placement algorithm, such as the following.

Point Placement 1 (PP-1):

1. If the strip is less than 24 km, place the single point in the middle of the strip.
2. For strips wider than 24 km, place two outer points 12 km inside each edge of the polygon, referenced from the intersection of the upper boundary line for the strip with the exterior boundary of the polygon.
3. If more than two scan points are needed, the first two are positioned as in step (2) and additional points are equally spaced between them.

3.6.2 Bounds for the Scan Sequencing Time

With a point placement heuristic, we can now bound the time necessary to scan the region:

Scan Sequencing Bound 1 (SSB-1): Compute a minimum spanning tree for the points computed by **PB-1** and placed by **PP-1**.

The weight of a minimum spanning tree (MST) for the complete graph induced by the scan points provides a lower bound on the time necessary to complete OSSP, with weights defined to be transit times. This is because an MST, which is an acyclic, minimum weight collection of $n-1$ arcs that connect all n nodes, need not visit the scan points sequentially as does an OSSP path. Thus the shortest OSSP path is at

least as long as a minimum spanning tree. This bound is valid independent of whether the region is polygon-convex, connected, or the scans are vertically aligned. It can be strengthened considerably by placing the MST in a Lagrangian objective function and using subgradient optimization to solve the dual.

Tree Relaxation with Lagrangian Objective Function. This algorithm uses a Lagrangian dual relaxation. It takes as input the induced graph $G = (N, A)$ where node 1 is an additional, fictitious node added to the original set of scan points and the arc costs c_{ij} are defined to be transit times. The distance from node 1 to all others is zero: $c_{1j} \equiv 0$. $\lambda = (\lambda_1 = 0, \lambda_2, \ldots, \lambda_{|N|}) \in \Re^{|N|}$ is the set of Lagrange multipliers; λ_1 is set to zero. The Lagrangian objective function is

$$z(\lambda) = \min_{x \in B^{|A|}} \left\{ \sum_{(i,j) \in A} (c_{ij} + \lambda_i + \lambda_j) x_{ij} \right\} - 2 \sum_{i \in N} \lambda_i, \tag{3.15}$$

where x is a 1-tree as defined in equations (3.3)–(3.5) and the fictitious node is the one constrained to be of degree 2. The constraint that is relaxed and placed in the Lagrangian objective function is equation (3.6); it requires that all nodes be of degree 2 in a tour. We observe that a tour through the set of scan points *with a fictitious node added* is the same as a shortest path. From equation (3.15) and the fact that the set of 1-trees contains all tours, it is easy to see that for any λ, $z(\lambda) \leq v(\text{OSSP})$.

The Lagrangian dual problem is

$$L(\bar{\lambda}) = \min_{\lambda \in \Re^{|N|}, \lambda_1 = 0} z(\lambda).$$

For given λ, we find a minimum weight spanning tree with respect to the weights $c_{ij} + \lambda_i + \lambda_j$, and then add to the weight of the MST the two smallest Lagrange multipliers (adding the two smallest λ_i's corresponds to including the two shortest arcs from node 1 to the MST in a 1-tree, since $c_{1j} \equiv 0$ for all j and $\lambda_1 = 0$). The lower bound is provided by then subtracting $(2 \sum_{i \in N} \lambda_i)$ from the tree weight. If

the tree computed is a path (all nodes are of degree 2, including two connected to the fictitious node), then the solution to the Lagrangian is feasible for OSSP. Since $z(\lambda) \leq v(\text{OSSP})$, such a solution is the optimal solution to OSSP. If the tree is not a path, however, subgradient optimization can be used to iterate on the λ's. Intuitively, we want to decrease λ_i when the degree of node i in the tree is equal to 1 and to increase λ_i when the degree of node i is greater than 2. For a given λ, the vector $\pi(\lambda)$ = (degree of node i in a minimum weight tree-2) is a subgradient to the objective function of the Lagrangian dual [84], and we only need to determine a step size to solve the dual by subgradient optimization. If the iterations are ended prior to producing an optimal solution, the algorithm could be continued using branch and bound to ultimately reach optimality.

A summary of the tree relaxation with Lagrangian objective function for a set of scan points N is as follows, where the subscript LB stands for "Lower Bound":

Tree Relaxation (TR):

1. Initialize. $k \leftarrow 0$; $\alpha \leftarrow 2.0$; $\lambda_i \leftarrow 0$ for all $i \in N$; $z_{LB}^k = 0$. The dummy node will be taken to be node 1, thus $c_{1i} = 0$ for all i and $\lambda_1^k = 0$ for all k.

2. Find a minimum spanning tree using $c_{ij} + \lambda_i + \lambda_j$ as the arc costs.

3. Compute the Lagrangian lower bound, $z(\lambda_k)$:

$$z(\lambda^k) \leftarrow \min_{x \in B^{|A|}} \left\{ \sum_{(i,j) \in A} (c_{ij} + \lambda_i^k + \lambda_j^k) x_{ij} \right\} - 2 \sum_{i \in N} \lambda_i \qquad (3.16)$$

   where x is also a 1-tree. If $z(\lambda^k) > z_{LB}$, then $z_{LB} \leftarrow z(\lambda^k)$. z_{LB} is the *greatest* Lagrangian lower bound from any iteration.

4. If z_{LB} has not improved for 20 iterations, then $\alpha \leftarrow \frac{\alpha}{2}$. If $\alpha \leq .005$, stop.

5. Compute the subgradients, an n-vector with components $(d_i^k - 2)$ where i is

an element of the node set N and d_i^k is the degree of node i in the minimum spanning 1-tree at iteration k. If $\sum_{i \in N}(d_i^k - 2)^2 = 0$ then stop, because the minimum spanning 1-tree just constructed is the optimal solution.

6. Set

$$\lambda_i^{k+1} \leftarrow \lambda_i^k + t^k(d_i^k - 2),\ i \in N \tag{3.17}$$

where

$$t^k \leftarrow \frac{\alpha(z_{UB} - z_{LB})}{\sum_{i \in N}(d_i^k - 2)^2}. \tag{3.18}$$

z_{UB} is an upper bound from a feasible solution to OSSP, which can be provided by any of a variety of heuristics described in section 3.7.

7. $k \leftarrow k + 1$ and return to step 2.

Conceptually we deal with *1-trees* in this algorithm; in our actual *implementation*, however, we deal with *trees*. Bookkeeping for the Lagrange multipliers requires computational details that are most easily explained using a 1-tree that includes a fictitious node.

The tree relaxation provides us with a second scan sequencing bound:

Scan Sequencing Bound 2 (SSB-2): Use **TR** for the scan points computed by **PB-1** and placed by **PP-1**.

3.6.3 Time Bounds for Continuous Scanning

Sensor transit times could reasonably follow either an L_1, L_2, or L_∞ metric[2]. If we assume a 2°/sec cross-track slew rate for moving the sensor, it will take approximately 975 ms to move 24 km laterally ($\frac{2 \times \arctan(\frac{12\text{ km}}{705\text{ km}})}{2°/\text{sec}} \approx 975$ ms). The along-track slew-rate

[2]The L_1 and L_∞ metrics in Euclidean 2-space are equivalent under a 45° rotation for the purposes of an optimization problem; the difference between them is a constant that can be neglected. Their iso-norms are identical except for the 45° rotation noted.

is assumed to be 5°/sec; another .5 seconds are needed for settling after movement in either direction. This leads us to consider bounds for classes of problems in which we employ the strategy referred to earlier: constrain ourselves to placing scan rectangles consisting of some predetermined number of consecutive along-track scans. Consecutive along-track scans avoid the time needed to move the sensor to a new location, and, in particular, avoid the settling time after such a movement. By considering bounds for problems in which we first constrain ourselves to perform some minimum number of along-track scans, we can parametrically evaluate the advantage of such restrictions by comparing the total scan time required for different such strategies. Otherwise, the only way to guarantee a lower bound on the amount of time needed to scan the entire target area is to use a scan point placement algorithm that *optimizes* the degree of along-track scan point alignment. *Maximizing* the amount of along-track alignment is tantamount to *minimizing* the number of scan tracks needed, where a *scan track* is a line on which scan points lie that is parallel to the satellite's ground track. Note that there is an obvious trade-off: although along-track scan alignment may require more scans overall to cover the target area, the time savings due to avoiding sensor movement may dwarf the time cost due to the additional scans. Any data collected from areas that are not of interest can simply be discarded—spatial data-editing. An example of a point placement algorithm that forces maximal scan alignment is as follows, where LB is the lower bound on the number of scan points computed in **PB-1**.

Point Placement 2 (PP-2):

1. Divide the region to be scanned into 24 km wide columns, starting from the point having the most negative x-coordinate. Let C be the number of columns produced.

2. Within each column i, compute the y-range, l_i, by differencing the y-coordinates of the along-track extreme points. Then $\left\lceil \frac{l_i}{30\,\mathrm{m}} \right\rceil$ will equal the number of along-

track scans in that particular column.

3. Continue to fill the columns in this fashion until LB is reached.

Constructed in this fashion, the scans are vertically aligned. The scan time lower bound for this sequence can be determined directly:

$$\text{Scan Time Bound} = \left(4.39 \text{ ms} \times \sum_{i=1}^{C} l_i\right) + (C-1) \times (975 \text{ ms} + .5 \text{ sec}).$$

4.39 ms is the amount of time it takes to perform a single instantaneous HIRIS scan, and 975 ms is the time necessary to move the sensor laterally 24 km.

3.6.4 Bounds for Regions with Holes

PB-1 is appropriate only for an area that is contiguous and contains no windows, or "polygon holes." More general regions, which we assume are bounded by straight lines that can have any slope, may contain such holes. Figure 3-12 is an example of the polygon in Figure 3-11 with a polygon hole added.

First we will consider a simple extension of **SSB-1** for a contiguous region with holes. The primary difference is that we compute a new point bound. In a region with holes, step 1 of **PB-1** will create strip boundary lines that cross the boundaries of interior holes, as well as the exterior boundaries of the polygon. As a result, we must modify step 2 in which we determine the number of scan points.

Point Bound 2 (PB-2):

1. Let the y-axis coincide with the along-track direction of the sensor or satellite. Beginning with the scan point having the minimum y-coordinate, create 30 m deep horizontal strips until the point with the maximum y-coordinate is contained within such a strip. The minimum and maximum y-coordinates are taken over all scan points. This is the same as step 1 of **PB-1**.

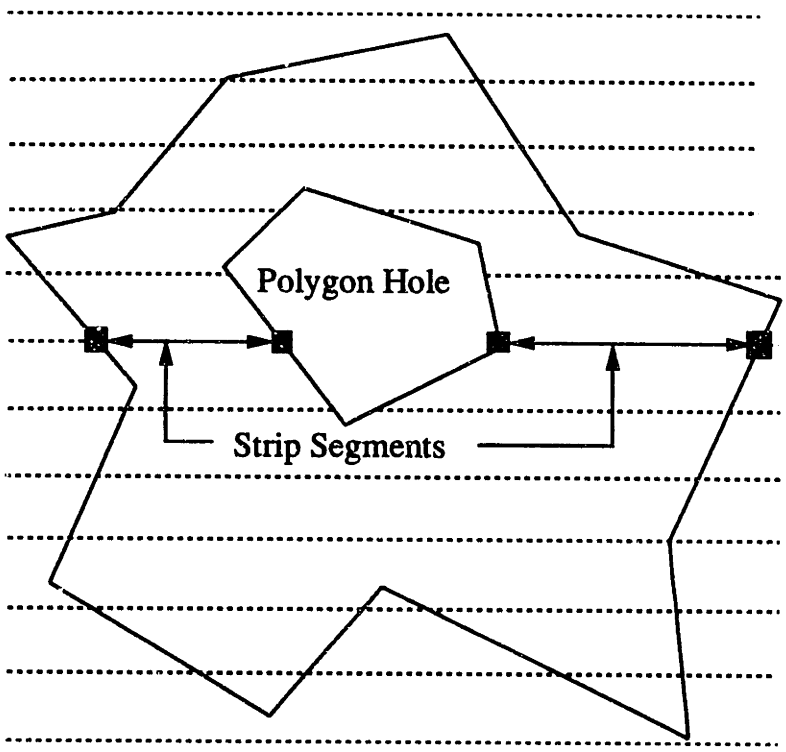

Figure 3-12: Polygon with hole.

2. Place $\left\lceil \frac{\text{segment width}}{24\text{ km}} \right\rceil$ scan points in each segment within a given strip.

A *segment* is a chord of the upper strip boundary line that runs between any two polygon boundaries (exterior or holes); its *width* is the length of that chord. Figure 3-12 gives an example of strip segments. The point placement algorithm must also be modified.

Point Placement 2 (PP-2): Same as **PP-1**, except we substitute "segment" for "strip."

Scan Sequencing Bounds 3 & 4: *Same as* **SSB-1** *and* **SSB-2**, *but use* **PB-2** *and* **PP-2** *as inputs.*

An alternative approach to establishing a lower bound on the time necessary to scan a region containing holes is to first partition it into more fundamental figures. Such partitioning problems have been considered in the context of VLSI lithography for which the simpler figures are trapezoids [1], rectangles [22], [86], and squares. Typically, however, the figures used to cover the target area are allowed to be of varying sizes, such as different shaped trapezoids. Our interest is in covering a region with uniformly sized rectangles (e.g., 24 km $\times$30 m).

A straightforward partition method is as follows:

> *For each vertex of a polygonal region, draw as many vertical chords as possible within the region to the first edge encountered.*

Figure 3-13 shows such a partition of a polygon with a hole.

This is not only very simple, but can be shown ([1], for a horizontal partition method) to be an optimal trapezoidal partition for a polygonal region P where no two edges are collinear and no two vertices have the same x-coordinate. The number of trapezoids is given by :

$$n(P) + w(P) - v(P) - 1$$

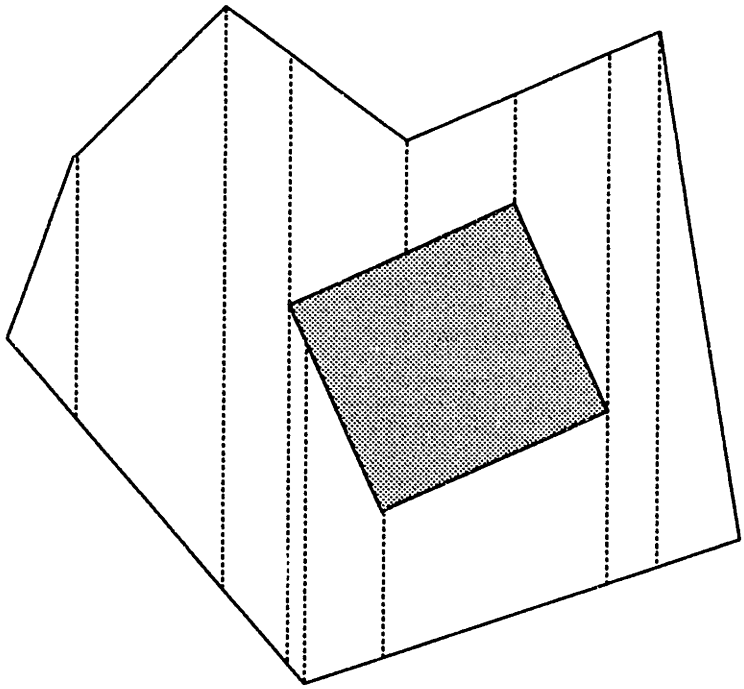

Figure 3-13: Partitioned polygon.

where $n(P)$, $w(P)$, and $v(P)$, are the number of vertices, windows, and vertical edges, respectively, of polygon P.

A lower bound on the amount of time necessary to scan each of the regions produced by this partition could then be computed by **SSB-1**. Another modification to **PB-1** is needed, though: we replace $\left\lceil \frac{\text{strip width}}{24\text{ km}} \right\rceil$ by $\left\lfloor \frac{\text{strip width}}{24\text{ km}} \right\rfloor$. The points selected will cover a strip only if its width is an integral multiple of 24 km; if a strip is less than 24 km wide, then $\left\lfloor \frac{\text{strip width}}{24\text{ km}} \right\rfloor = 0$ and the strip will be unscanned. This is done because the positions of the vertices of a particular polygon, both for its exterior boundary and its holes, could drive the vertical partition of the region to produce many sub-regions that are of very small width. If each was scanned, the sum of those scan times could exceed the optimal value even when ignoring the need to patch them together. An alternative to "under-covering" strips would be to use some sort of a merging process to join together adjacent strips that are very slender.

Since the spanning trees (which need not be paths) for each region are not patched together, the sum of their lengths represents a lower bound on the time needed to scan the region. We have constructed a spanning forest for the *union* of the scan points, rather than a spanning tree. Scan time bounds provided by **SSB-2** generally dominate those produced by the partitioning process, since the under-estimation of scan points needed will be less in the former case than in the latter.

3.7 Scan Sequencing Algorithms

In this section we consider algorithms for several different feasible solution producing heuristics, as well for local optimization heuristics. The best worst-case bound for most of them is $O(n^2)$. All of the ones that we describe have been implemented in C, and their performance is described in section 3.8. Despite their lack of good analytical time bounds, the empirical evaluation of such algorithms demonstrates their utility as tools for use in analyzing practical, real-world problems such as scan sequencing.

Since geometric versions of the TSP remain $\mathcal{NP}$-hard, heuristics are where much of the experimental work for them has been directed. The algorithms we describe are not limited in their use to geometric problems, but rather to problems for which the triangle inequality holds. The focus of our work is on the lengths of the paths computed, rather than the computational time it takes to produce them. Nonetheless, we have pursued only those approaches that have a history of efficient implementation and performance.

3.7.1 Nearest Neighbor

The nearest neighbor algorithm is a heuristic that grows a fragment of a tour or path. The following is a straightforward implementation:

NN:

1. Start with an arbitrarily chosen scan point, ρ_1.
2. If the current fragment is $\rho_1, \rho_2, \ldots, \rho_k, k < n$, let ρ_{k+1} be the scan point that is not currently on the path and which is closest to ρ_k, and add it to the path.
3. When all the scan points have been visited, add an arc from the last scan point back to the first. This produces a tour.
4. Delete the single longest link in the tour to produce a feasible path.

Note that the final arc that is added from the last scan point visited back to the starting point need not be the longest link in the tour. That is why it is included, and in the final step we delete the arc that is the longest.

The greatest shortcoming of this approach is that although the earlier arcs that are selected may be relatively short, the last several arcs may be quite long. Rosenkrantz, Stearns & Lewis [95] have shown that the length of a geometric nearest neighbor tour is at most $(\lceil \log n \rceil + 1)/2$ times the length of the optimal tour. If the triangle inequality does not hold, the ratio of *value*(**NN**) to the optimal solution can be arbitrarily large, independent of n.

An obvious implementation of this algorithm accesses the scan points as an array. The search for the next point to be added to the path would then take $O(n)$ time, and must be done n times, for an $O(n^2)$ time bound. We implement **NN** using a 2-dimensional binary search tree (K-d tree, in general, for K dimensions). The construction of a K-d tree takes $O(n \log n)$ time, and deletions can be done in $O(n)$ time. Bentley [16] hypothesizes a constant expected time for nearest neighbor searching,

yielding an overall $O(n \log n)$ time bound for computing a **NN** path. A common strategy is to build the data structure, and then construct m different **NN** tours using m different starting points (m is arbitrarily chosen, and presumably much less than n; computational experience can indicate what a good value of m is relative to n). Such an approach would result in $O(n \log n + mn)$, assuming that Bentley's constant-expected-time hypothesis is correct.

3.7.2 Multiple Fragment

The nearest neighbor algorithm is a straightforward greedy algorithm. The multiple fragment (**MF**) algorithm is a less obvious greedy approach that can be viewed as an extension of Kruskal's algorithm for computing a minimum spanning tree. Kruskal's MST algorithm considers the arcs of the underlying network in increasing order of length, including in the tree any arc that does not create a cycle. **MF** does the same except that an arc is excluded if it creates a cycle or a node of degree three. Thus it is called "multiple fragment" [16] because the tour is grown from a multiplicity of shorter paths that ultimately will merge as arcs are selected that connect the fragments. Bentley and Saxe [15] analyze a similar heuristic, and find its worst case performance to be $O(\sqrt{n})$ times the optimal distance. The approach just outlined takes $O(n^2 \log n)$ time. If a K-d tree is used for nearest neighbor searching and a priority queue for nearest neighbor links, it is argued that **MF** runs in $O(n \log n)$ expected time overall [16]. The following is a statement of the multiple fragment algorithm:

MF:

1. Begin with each scan point an individual fragment. Construct a K-d tree for nearest neighbor searching, and use a heap to support a priority queue that contains the nearest neighbor of each scan point that is eligible to be linked to another arc.

2. Select the shortest arc from the priority queue (the arc length is its priority).

3. Update the data structures, and return to step 2. Stop when n -- 1 arcs have been chosen.

The key to this algorithm running efficiently is the data structures. In addition to the K-d tree and priority queue, there is an array that has the number of path edges adjacent to each scan point; an array that contains, for each scan point, the index of the nearest scan point; and an array that identifies the end points of the fragments. Note that as opposed to the NN algorithm, we stop when we have $n - 1$ arcs, since the next arc that would be selected to complete a tour must be longer than any that has been selected so far.

3.7.3 Spanning Tree Based Heuristics

The spanning tree based heuristic that we implement is a Christofides-type algorithm, however, before describing it we first detail the original minimum spanning tree heuristic.

3.7.3.1 The Minimum Spanning Tree Heuristic

This heuristic has long been used in the analysis of the TSP. A spanning tree for a graph is a collection of $n - 1$ arcs that produces a connected subgraph. There are a number of fast algorithms for computing a minimum weight spanning tree, the most well known of which are due to Prim [91] and Kruskal [67]. Prim's algorithm is a straightforward greedy approach in which the node closest to any of those already in the tree is added; it is essentially a nearest neighbor routine. Kruskal's algorithm successively considers arcs in order of increasing length, adding them to the tree so long as they do not create a cycle.

If the input for the computation of an MST is in the form of a distance matrix, it can be found in $O(n^2)$ time using either Prim's or Kruskal's algorithms. As observed

earlier, the minimum spanning tree provides a lower bound on the length of the optimal path. If the arcs of the MST are then toured in a depth-first traversal, this will produce a path that traverses no arc of the MST more than twice. Such a path provides a route that visits all the nodes and is of length at most twice that of the minimum spanning tree.

Depth first traversal is also known as *depth-first search plus backtracking*. The arcs of the graph searched are stored in a *last in, first out* (LIFO) fashion when they are visited. If any arc from the current node has not been traversed, that arc is followed to a new node. If all arcs from the present node have been crossed, then the search goes back along the arc by which the present node was first reached to return to the node from which it was visited (thus the arcs are stored in a LIFO data structure). This continues until the starting node is revisited and it has no untraversed edges.

Depth first traversal can be used to find an *Eulerian walk*, a closed walk in which each node appears at least once and each arc appears exactly once. The following theorem is is due to Euler:

Theorem 3.3 *A graph G=(N,A) contains an Eulerian walk if and only if*

1. *G is connected*

2. *All nodes in N are of even degree.*

If the arcs of a spanning tree are doubled, the resulting graph is Eulerian. A depth first traversal of the minimum spanning tree is equivalent to finding an Eulerian walk in the *doubled* minimum spanning tree.

An Eulerian walk may visit a node more than once, and it is here that the triangle inequality can be exploited in the optimal scan sequencing problem. If the Eulerian walk causes the path to go back to an already visited node, we can skip ahead to the next unvisited node in the sequence. A direct link between two scan points can be no longer than previous path that contained intervening points. This construction provides a path, and since its length can be at most twice that of the original tree,

which was a lower bound, this path is within a factor of two of the optimal solution.

A summary of the minimum spanning tree algorithm:

MST:

1. Find a minimum spanning tree for the set of scan points.
2. Compute an Eulerian walk of the MST.
3. Take shortcuts as needed to produce the final path.

The first step will be $O(n^2)$, and the second and third are each $O(n)$.

The following recursive procedure will find an Eulerian walk in a graph, and is based on [90, p. 413]:

Euler:

1. Accept as input a node, n_1, of the graph $G = (N, A)$. If n_1 has no adjacent edges, then return as output n_1; this is an empty walk. Otherwise, go to step 2.
2. Starting at n_1, construct a walk in G that never traverses the same edge twice, and that ultimately returns to n_1. Let $< n_1, n_2, \ldots, n_k >$ be the walk created. Delete from G all the edges in the walk.
3. Return <**Euler**(n_1),**Euler**(n_2),..., **Euler**(n_k)>.

This procedure runs in $O(|A|)$ time. As nodes are visited during a walk, they can be pushed onto a stack, and then popped off for the recursive call. An adjacency list representation of the MST efficiently supports constructing walks and deleting arcs as they are traversed. In our implementation of **Euler** as part of the algorithm described in section 3.7.3.2, step (2) is completed by performing a depth first search that stops when the input node is revisited.

3.7.3.2 Christofides-Type Algorithm

Christofides' [25] heuristic makes very clever use of the minimum spanning tree. It involves the concept of a minimum weight matching. A *matching* is a collection of arcs in a graph such that each node is an endpoint of at most one arc. If each node is the endpoint of *exactly* one arc, then the matching is *complete*. A *minimum weight* matching is one for which the total length of the arcs selected is minimized. Christofides algorithm for finding a TSP tour is summarized as follows:

Christofides:

1. Find the minimum spanning tree for the scan points.
2. Compute the minimum weight complete matching for all the nodes in the minimum spanning tree that are of *odd* degree.
3. Add the edges identified in the matching to the MST; this produces an Eulerian graph (a graph that contains an Eulerian walk).
4. Find an Eulerian walk of the resulting graph, introducing shortcuts to determine the final path.

An optimal minimum weight matching can be found in $O(n^3)$ time, and this step dominates the algorithm. Note that there is always an even number of odd degree nodes in the MST. Christofides' heuristic provides what is currently the best worst case bound for a TSP that obeys the triangle inequality.

Theorem 3.4 *For any instance of the triangle inequality TSP, Christofides' heuristic provides a feasible solution whose length is at most* $\frac{3}{2}$ *times that of the optimal solution.*

A slightly different matching problem is solved to provide a $\frac{1}{2}$-approximate algorithm for the shortest Hamiltonian path problem rather than the TSP. It entails adding two fictitious nodes to the collection of odd-degree nodes in the MST. The distance between these two nodes is infinite, and the distance d between each of them

and all other nodes is equal and greater than $\max_{i,j} c_{i,j}$. The minimum weight matching will then identify the two nodes that are to remain *unmatched* in the shortest path as the mates of the two fictitious nodes. Figure 3-14 shows the construction. The arcs connecting the two nodes on the path to the two fictitious nodes will contribute a constant to the weight of *any* matching computed for the set of odd degree nodes in the MST: $2 \times d$. Thus if there exists a matching of shorter length that leaves two nodes "unmatched" (i.e., mated to the fictitious nodes), then the optimal matching algorithm would have found it.

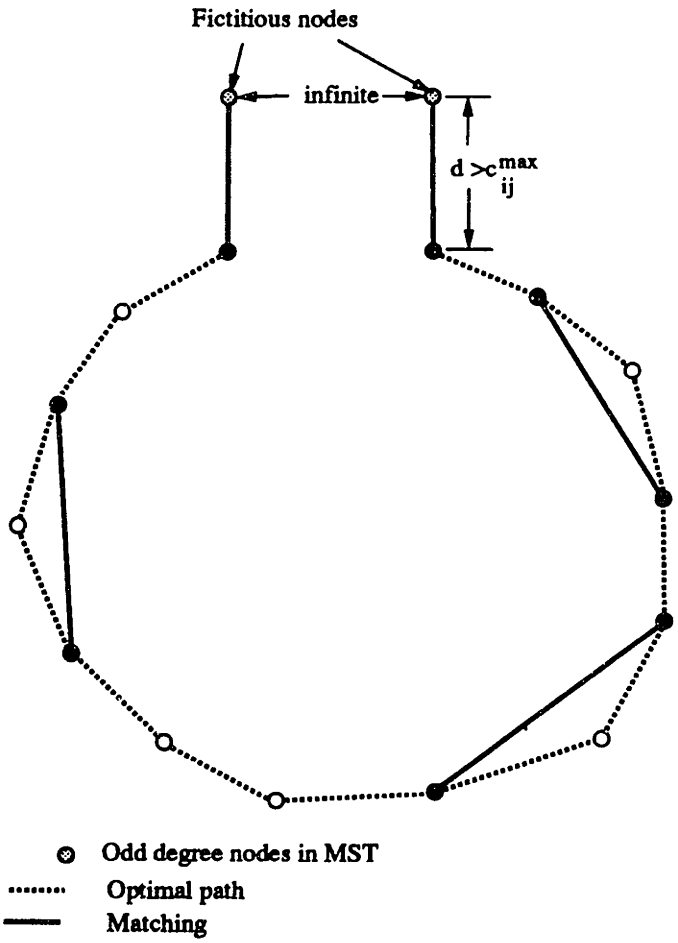

Figure 3-14: The modified matching solved when Christofides' algorithm is applied to a path rather than a tour.

In our implementation we do not actually find the optimal minimum weight matching, but rather find a greedy matching: successively select the shortest arc among those feasible so long as it is not incident on a node that has already been matched. This produces a much faster algorithm, since finding the minimum weight matching was the bottleneck step in the runtime for Christofides' heuristic. Avis, Davis, and Steele [3] show that for every dimension $k \geq 2$, the expected length of a greedy matching grows as $\alpha n^{1-\frac{1}{k}} + o((n^{1-\frac{1}{k}})$, which is within a constant factor of the optimal solution. Matchings produced by the greedy algorithm can usually be improved by using two-opt [16].

In addition, we do not use the minimum spanning tree as input for the matching of odd-degree nodes. Instead, the final spanning tree that is produced by the tree relaxation (**TR**) is used. Solution of the Lagrangian dual identifies a tree with a small number of nodes that are not of degree 2, since the subgradient optimization process serves to so drive the MST computations. A description of the tree relaxation with Lagrangian objective function is contained in section 3.6.

Note that adding the *degree-k* side condition to the minimum spanning tree problem can make it hard:

DEGREE-k MINIMUM SPANNING TREE:

Instance: Integer $n \geq 3$, $n \times n$ distance matrix C.

Question: What is the shortest spanning tree for C in which no vertex has degree exceeding k?

DEGREE-2 MINIMUM SPANNING TREE is the WANDERING SALESMAN version of the TSP, and is thus $\mathcal{NP}$-hard. In fact, the DEGREE-k MINIMUM SPANNING TREE problem is $\mathcal{NP}$-hard for any fixed $k \geq 2$ and general graphs. If the graphs are restricted to being Euclidean ($\Re^2$), the problem is $\mathcal{NP}$-hard for $k = 2$ and 3. Complexity for $k = 4$ is unknown. For $k \geq 5$ the EUCLIDEAN DEGREE-k MINIMUM SPANNING TREE problem is equivalent to the unconstrained problem since it can be shown that the MST of a graph whose nodes are on the planar lattice can have no

vertex of degree greater than 5.

3.7.4 Local Optimization

In this section we describe two heuristics that are used to improve a previously constructed feasible path. The best known tour improvement approaches, of which these are representative, are edge exchange procedures [31], [71], [72]. In general, an r-opt procedure entails exchanging r arcs in a feasible path for r arcs not in the current solution such that a path is maintained and its length is less than it was before.

3.7.4.1 Two-Opt

Figure 3-15 demonstrates a two-opt swap in a path. Two arcs that cross one another, 1–4 and 2–3, are exchanged for two arcs that do not, 1–2 and 3–4. This is equivalent to reversing a subsequence of the cities. If a path cannot be improved by performing any further two-opt exchanges, then it is two-optimal.

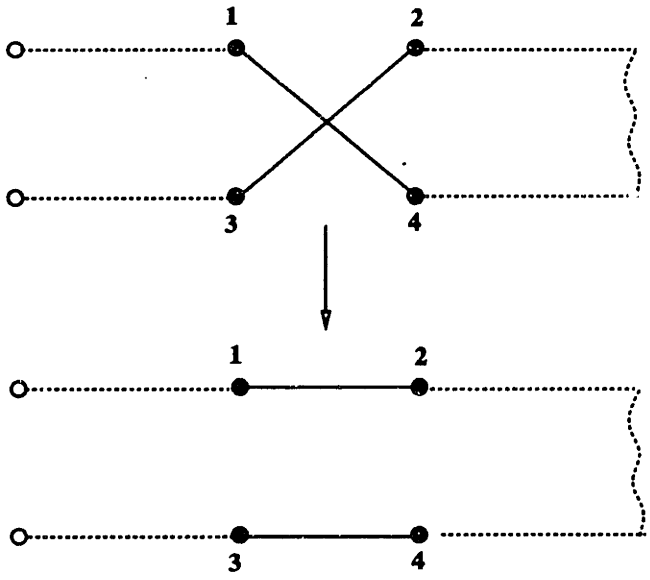

Figure 3-15: Two-opt example.

An obvious but inefficient way to perform two-opt exchanges would involve exam-

ining all possible pairs of arcs in the path, of which there are $(n-1)(n-2)/2$, and executing an exchange if a suitable pair is found. This would take $O(n^2)$ time, and achieving a two-optimal path potentially much longer. In addition, there are many choices for implementing two-opt. They include specifying the order of checking arcs for possible swaps, the selection of the arcs to be exchanged, which two-opt to perform if there is more than one possible, and so on.

Our approach follows that of [16]. We walk along the path, visiting each scan point in order, and inspect both neighbors of the current scan point for candidate arcs to use in a swap. A fixed-radius nearest neighbor search centered at the scan point under consideration is used to identify any other scan points in the vicinity. For each such scan point so found, the neighbor that could be used in a swap is checked to see if an exchange would reduce the path length. Note that there is only one candidate neighbor for the points identified in the search, since using the other neighbor in a swap would serve to break the path into a fragment or a cycle, as can be seen in Figure 3-15. If a walk of the path can be completed without any two-opt exchanges being found, then it is considered two-optimal.

3.7.4.2 Two-H-Opt

A two-H-opt, or two-and-a-half-opt exchange is a limited version of a 3-opt swap. A 3-opt exchange involves 6 points, while two-H-opt only deals with 5. Whereas two-opt reverses a subsequence of the path, two-H-opt takes a single scan point and moves it to another spot in the path. Figure 3-16 shows an example. If no more two-H-opt exchanges improve a path, then it is considered two-H-optimal, analogous to a two-optimal path.

Two-H-opt is implemented as an extension of two-opt. As before, a fixed-radius nearest neighbor search is made around each scan point in the walk to identify other points whose adjacent arcs are candidates for a two-opt swap. Each point found in the search is also considered for a two-H-opt exchange in which it is the point moved.

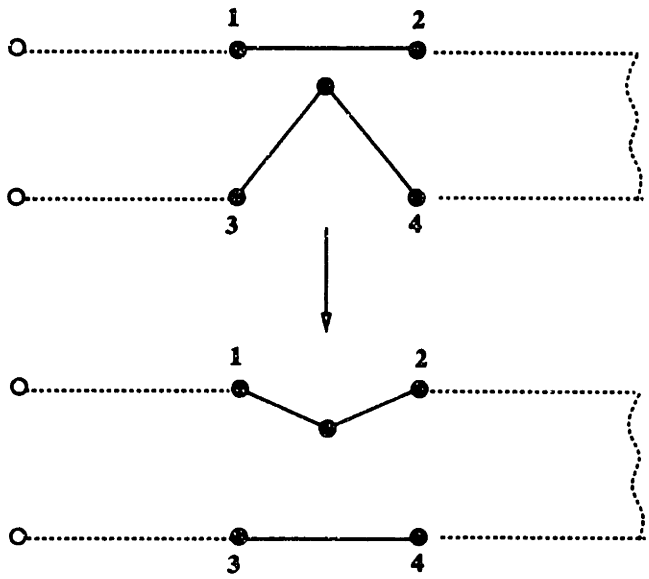

Figure 3-16: Two-H-opt example.

And then the original point about which the search was conducted is considered for a move in a two-H-opt swap.

3.8 Computational Testing

The algorithms described in section 3.7 were implemented in *C* and then computationally tested on either a DECstation 5000 or a VAXstation 3100. Testing involved first generating problems believed to be representative of the uses for which such space-based sensors will be employed. HIRIS is again taken as an example, and regions that are judged to be reasonable representations of the objectives of science users were constructed. This is similar to the approach that has been taken by researchers at the Jet Propulsion Laboratory (JPL), in that the generation of demand for driving their simulations of HIRIS operations was done with little input from the science team—they produced the imaging projects on their own [94].

3.8.1 Problem Generator

The regions for imaging were taken to be polygons, with randomly generated features. This was done by beginning with a circle of radius r, centered at the origin. At equally distributed angles around the circle, two values, γ_x and γ_y, are drawn randomly from $U(-\alpha/2, +\alpha/2)$, a distribution that is uniform over the symmetric interval $[-\alpha/2, +\alpha/2]$. They are then added to r and projected onto the x and y axes, producing the x coordinate $((r+\gamma_x)\sin\theta)$ and the y coordinate $(r+\gamma_y)\sin\theta)$ of the feature that is at the specified angle θ. Inputs for generating the polygon include the radius r of the underlying circle; the number of features η, which determines the angles at which they are computed since the features are evenly distributed around the circle; the range α; and a seed for a random number generator that is used to provide the draws from $[-\alpha/2, +\alpha/2]$. A summary of how the polygons are produced, where (x_i, y_i) are the coordinates of the ith feature of the polygon:

1. Input r, η, , α, and a random number seed.

2. For $i = 1$ to η, do:

   (a) Draw γ_x and γ_y from $U(-\alpha/2, +\alpha/2)$; $\theta = i \times \frac{360}{\eta}$.

   (b) $x_i = (r+\gamma_x)\sin\theta$; $y_i = (r+\gamma_y)\sin\theta$.

   (c) $i = i + 1$.

3. Return (x_i, y_i) for all i.

An example of a polygon generated by this algorithm is shown in Figure 3-17. Its parameters are $r = 40$, $\eta = 36$, $\alpha = 30$. Units for the parameters are unnecessary, since they can be arbitrarily scaled. If we think of r and α as being in kilometers, however, then the region is roughly of a size that could be imaged by HIRIS as part of an imaging project. As can be seen, shapes generated in this manner need not be polygon-convex, although they must be connected.

After generating the region, a slightly modified version of the heuristic **PB-1** (discussed in section 3.6.1) is used to determine the number of scan points that are needed to cover the region. The modification is that the number of continuous along-track scans that are performed before considering moving the sensor is an input, β; in **PB-1** β was set to 1, which for HIRIS produces a strip depth of 30 meters. If β were 167, then the strip depth would be 5 kilometers. This allows us to examine the different classes of strategies referred to earlier. In addition, the *upper bound* strip width (see Figure 3-11) is used in **PB-1** so that we produce a cover of the region.

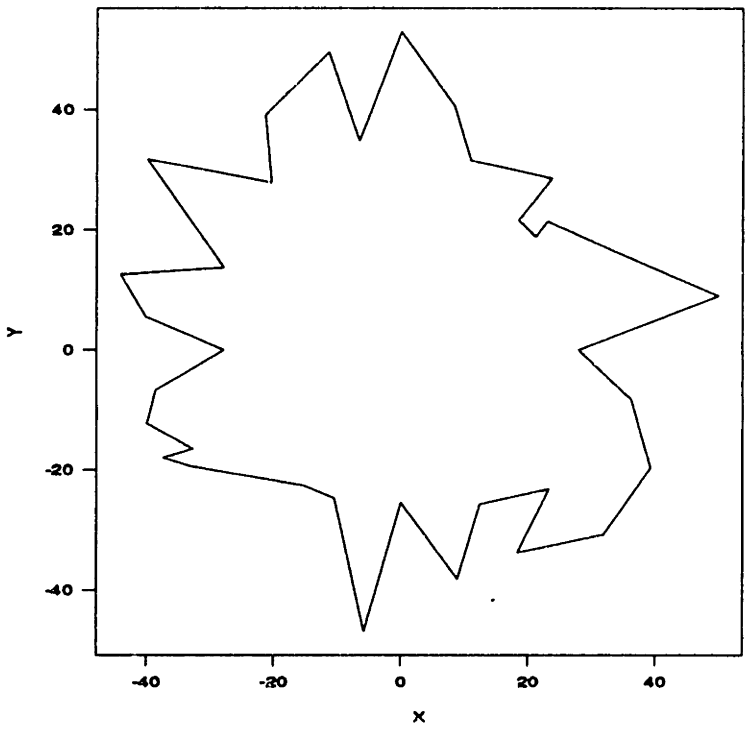

Figure 3-17: Sample polygon produced by the problem generator.

PP-1 is then used to place the scan points in strips whose depth is β times the spatial resolution for the sensor (this depth is the actual input to the heuristic). Figure 3-18 shows the scan points constructed in this fashion for the polygon of Figure 3-17; 56 scans of depth 5.0 are needed to cover the region. If β is reduced much below 5.0, it is generally not possible to distinguish between different scan points on the graphical plot.

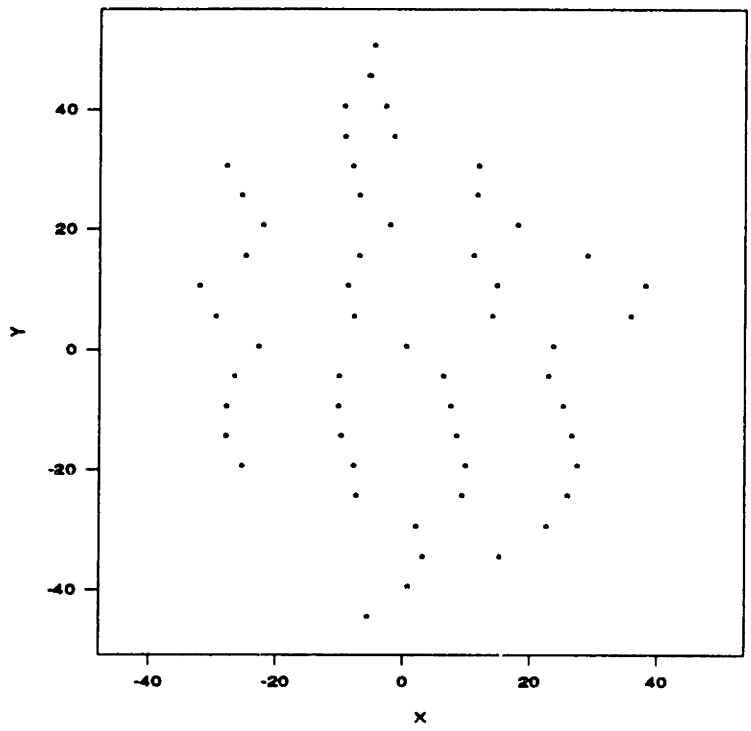

Figure 3-18: Scan points for the sample polygon produced by the problem generator.

3.8.2 Data Structures

A significant shortcoming of many heuristics for the TSP and shortest Hamiltonian path problems is that when they are implemented in the obvious way, they spend a large amount of time considering arcs between nodes that are not likely to be contained in any sensible path. Two different approaches have been described recently in the literature [16], [62], [92] that provide the means to speed up spatial searches that are necessary in geometric based problems. We discuss these approaches because the use of such data structures is vital in efficiently implementing the path heuristics.

Reinelt [92] exploits the information contained in the Voronoi diagram or its dual, the Delaunay triangulation. A Voronoi diagram is a partition of the plane into N polygonal regions, each of which is associated with a given point. The region associated with a given point is the locus of points closer to that point than to any other given point. Points x and y are connected in the Delaunay triangulation if their

Voronoi polygons have an edge in common. There are several efficient algorithms for computing the Voronoi diagram (and hence the Delaunay triangulation). Shamos and Hoey [104] present a divide-and-conquer algorithm that computes the diagram in $O(n \log n)$ time. Ohya, Iri and Murota [87] give an algorithm with a worst case time bound of $O(n^2)$ that is said run in linear time in practice. Nonetheless, writing a program to compute the Voronoi dual is challenging [103].

The Delaunay graph is used in [92] to restrict the extent of spatial searches by providing, for each point, a set of arcs that are candidates to be in a tour. Two such sets are constructed, the first of which is the arcs to a point's k nearest neighbors, where k was taken to be between 5 and 20. The k nearest neighbors are found by examining nodes that are at most k arcs away in the Delaunay graph. This can be done in linear time for fixed k by doing a breadth-first-search. The second set of candidate arcs begins with arcs in the Delaunay graph and adds to them transitive arcs of order 2 (arcs between nodes that are two arcs apart in the Delaunay graph).

The other efficient way for capturing the spatial character of geometric problems and finding nearest neighbors is to use a K-d tree [14], [16], [62], [103], which is a K-dimensional binary search tree. That is the approach we implement; it assists in two types of searches in particular, nearest neighbor and fixed-radius. The version used here is a semidynamic tree that, once constructed, can support deletions and undeletions but not the insertion of new points. Restricting K-d trees to semidynamic point sets allows a pointer array that is indexed by the point number to tell which terminal node of the tree contains a given point. Many functions can then be implemented using "bottom-up" algorithms for which the expected run time is reduced from $O(n \log n)$ to $O(1)$, or constant time. These operations include deletion, undeletion, nearest neighbor searching, and fixed-radius near neighbor searching. The time to build the tree itself is $O(Kn + n \log n)$. Bentley [14] describes a wide spectrum of K-d trees.

A K-d tree supports fast implementation of the various path heuristics and local

optimization routines presented in section 3.7. The nearest neighbor heuristic **NN** can be done in a loop that performs a nearest neighbor search and then deletes the point identified from the K-d tree. The multiple fragment heuristic **MF** uses a heap representation of a priority queue in addition to the K-d tree for a fast implementation. Greedy matching, minimum spanning tree, two-opt and two-H-opt are other proximity problems that can be solved using K-d trees.

3.8.3 Simulations

3.8.3.1 Data

The algorithms presented in section 3.7 have been implemented in C and computationally tested on either a DECstation 5000 or a VAXstation 3100. The Euclidean distance between points was used as an approximation for the actual metric of calculating the perceived angle between points. Since the scale factors needed for this approximation remain the same and would appear in every distance, they are a constant that can be neglected.

One of two heuristics, nearest neighbor (**NN**) or multiple fragment (**MF**), was used to generate an initial feasible solution. Each solution was then potentially followed by one of two local optimization heuristics, two-opt (2-Opt) or two-H-opt (2H-Opt). This produced six different possible upper bounds on the optimal solution, each of which was then used in equation (3.18) of **TR** step (6), the subgradient optimization step of the tree relaxation with Lagrangian objective function. Table 3.3 spells out the different combinations. In Appendix B we provide figures showing the starting tour (from which we obtain an initial path) for each of these six combinations using the scan points of Figure 3-18 as input.

The tree relaxation provides a lower bound for the optimal solution of OSSP, in the process of which it identifies spanning trees for the set of scan points that are successively "better" in that they have increasingly fewer nodes that are not of degree 2. The final tree computed by the Lagrangian is then input to a Christofides-type

Initial Feasible Solutions for OSSP
Nearest Neighbor, followed by 2-Opt
Nearest Neighbor, followed by 2H-Opt
Nearest Neighbor, with no Local Optimization
Multiple Fragment, followed by 2-Opt
Multiple Fragment, followed by 2H-Opt
Multiple Fragment, with no Local Optimization

Table 3.3: Combinations of path-forming and local optimization heuristics that provide initial feasible solutions whose length can be used in the subgradient optimization step of the tree relaxation with Lagrangian objective function.

heuristic, producing another feasible solution to OSSP. This path can then potentially be improved by use of the same two local optimization procedures, two-opt and 2-H-opt. Thus in addition to the feasible solutions listed in Table 3.3, we also produce those shown in Table 3.4.

Final Feasible Solutions for OSSP from Tree Relaxation
Christofides heuristic, with no Local Optimization
Christofides heuristic, followed by 2-Opt
Christofides heuristic, followed by 2H-Opt

Table 3.4: Final feasible solutions for OSSP produced from the tree relaxation.

The cross-product of the initial upper bounds and those derived from the tree relaxation, *plus the initial feasible solutions themselves*, yields 24 different combinations of heuristics that produce feasible solutions. Each of these is compared to the lower bound from the Lagrangian dual problem (LLB), and the gaps between them are computed in either distance units (nominally kilometers) or as a percentage of the lower bound:

$$\text{Gap in percent} = \frac{\text{Upper Bound} - \text{Lower Bound}}{\text{Lower Bound}}.$$

Before displaying the data, we clarify how the various heuristics interact by providing figures that show a sequence of them in use. In each figure, the borders of

the region and symbols for scan points have been removed to improve clearness. The initial heuristic used is multiple fragment, and it takes as input the scan points shown in Figure 3-18, which were computed for the region shown in Figure 3-17. The nearest neighbor starting heuristic is followed by *no* local optimization, and the tour it produces is shown in Figure 3-19 (in Appendix B we provide figures showing *all* of the tours produced by the six different starting heuristics for the scan points of Figure 3-18). Note that we compute and display a *tour*, rather than a path. Although the *last* arc included in the tour formed by the heuristic is often the longest, it need not be. Thus by finding a tour we can then delete the longest arc amongst all the arcs in the tour—a less restrictive approach than specifying the starting and stopping points. When appropriate, this is done in other heuristics as well. The *length* of the path specified by the nearest neighbor heuristic is then used in the subgradient optimization of **TR**. *This is the sole role that the heuristic for an initial feasible solution plays in subsequent computations: it provides an upper bound that is used in calculating the step size for updating the Lagrange multipliers (λ's) in* **TR**. The *first* minimum spanning tree computed by **TR** is shown in Figure 3-20; this is simply an MST for the scan points of Figure 3-18. The *final* minimum spanning tree found by the tree relaxation with Lagrangian objective function is shown in Figure 3-21. The feasible solution computed using the minimum spanning tree of Figure 3-21 as input to Christofides' heuristic is shown in Figure 3-22; no local optimization has been done.

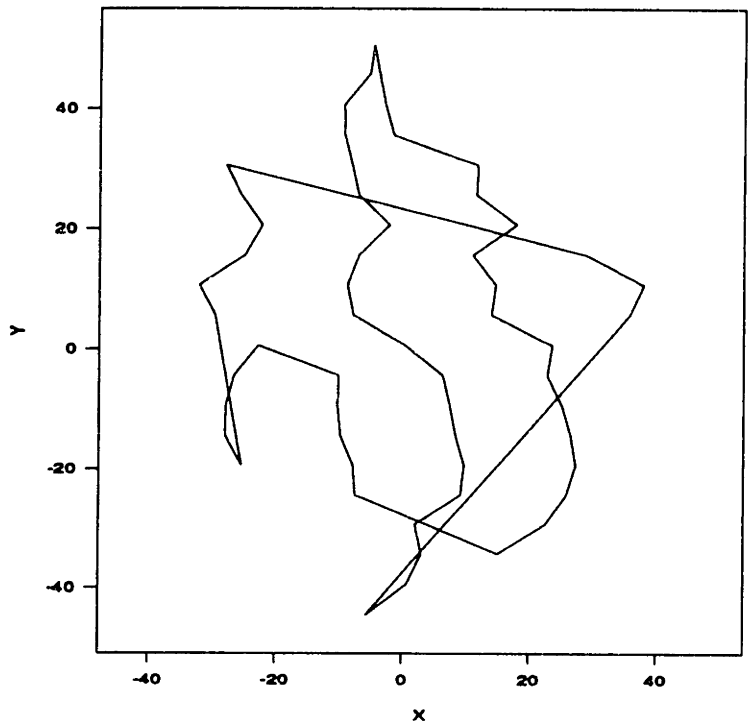

Figure 3-19: Tour formed by nearest neighbor heuristic with *no* local optimization.

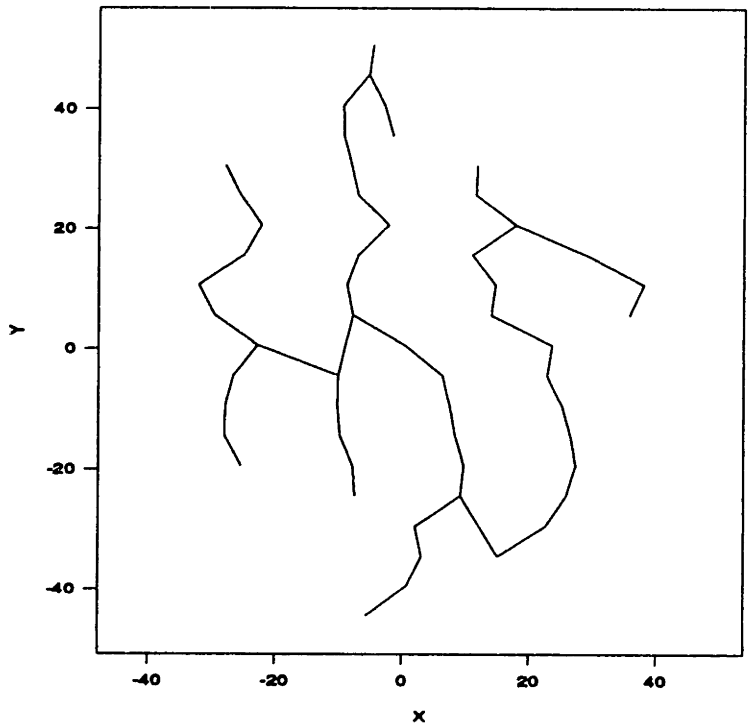

Figure 3-20: Minimum spanning tree for scan points.

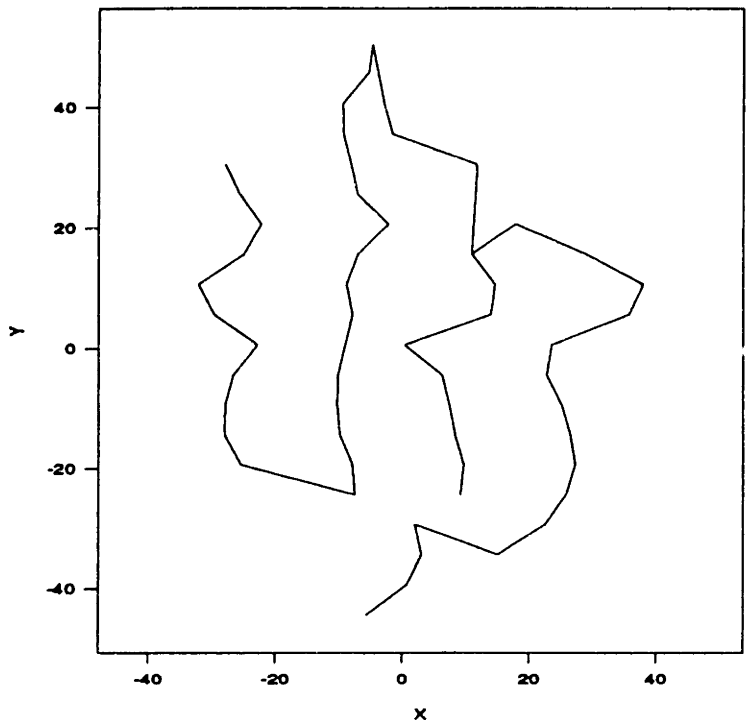

Figure 3-21: Final MST from tree relaxation with Lagrangian objective function.

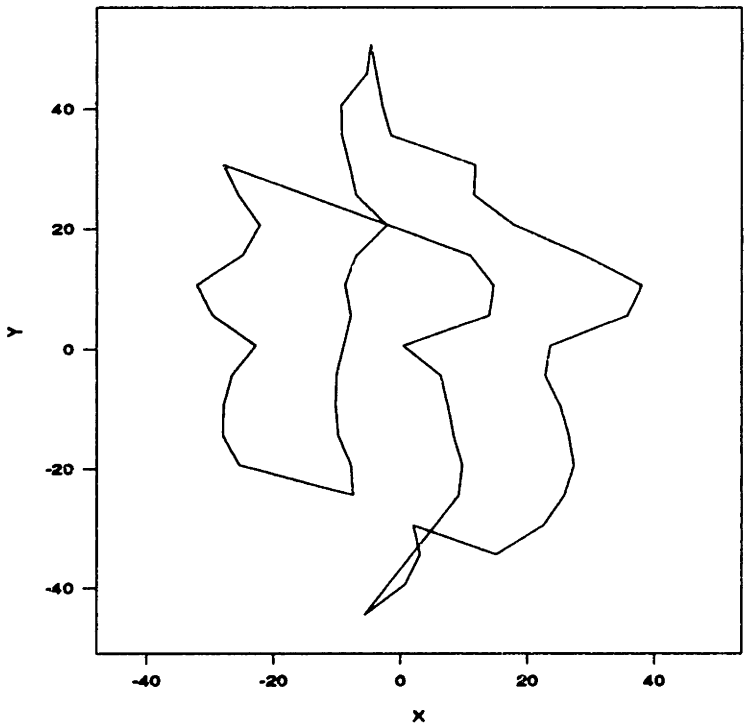

Figure 3-22: Feasible solution from Christofides' heuristic that uses the final MST from **TR** as input.

Table 3.6 displays the average gap, in kilometers, for 100 problem instances with a scan depth of 5.0 km. The scan depth refers to the along-track distance that the sensor will image before repointing; it essentially results in the placement of 5 km×24 km rectangles for HIRIS. The parameters used in generating the regions that constitute a problem instance are the same as those that produced Figure 3-17: circle radius of 40 km with a potential range of ±15 km, and 36 features. Scan points were placed using the algorithms of section 3.6.1 as modified in section 3.8.1, an example of which is displayed in Figure 3-18. Table 3.7 presents the same results with the gaps expressed in percentages rather than distance. Figure 3.5 specifies the abbreviations used in these and subsequent tables and figures.

Legend	
IFS	Initial Feasible Solution
NN	Nearest Neighbor
MF	Multiple Fragment
LLB	Lagrangian Lower Bound
LO	Local Optimization
CFS	Christofides Feasible Solution

Table 3.5: Legend for tables and figures.

IFS	Local Opt (LO) Heuristic	Init Feas Soln− Lagrangian LB	Christofides Feasible Solution		
			+No LO − LLB	+2-Opt−LLB	+2H-Opt−LLB
NN	2-Opt	23.78041	19.00723	20.48472	17.27395
	2H-Opt	19.27257	18.13398	20.02544	17.30529
	No Local Opt	48.93607	20.20401	22.16178	18.41962
MF	2-Opt	21.15284	18.61599	21.41143	17.52738
	2H-Opt	17.91947	19.09654	21.77544	17.78759
	No Local Opt	21.91366	20.43237	21.06920	17.32557

Table 3.6: Summary of solution gaps, expressed in km, for a scan depth of 5.0.

Similarly, Tables 3.8 and 3.9 contain the average gaps, expressed in kilometers and percentages, respectively, for problem instances that are parameterized by a scan depth of 1.0 km (but that are otherwise analogous).

IFS	Local Opt (LO) Heuristic	Init Feas Soln– Lagrangian LB	Christofides Feasible Solution		
			+No LO – LLB	+2-Opt–LLB	+2H-Opt–LLB
NN	2-Opt	5.655579	4.488320	4.841463	4.084706
	2H-Opt	4.569976	4.277951	4.741894	4.094392
	No Local Opt	11.591122	4.788114	5.266107	4.369063
MF	2-Opt	5.023462	4.398191	5.059161	4.150057
	2H-Opt	4.253908	4.500003	5.164872	4.211773
	No Local Opt	5.154425	4.828965	4.970517	4.084878

Table 3.7: Summary of solution gaps, expressed in %, for a scan depth of 5.0.

IFS	Local Opt (LO) Heuristic	Init Feas Soln– Lagrangian LB	Christofides Feasible Solution		
			+No LO – LLB	+2-Opt–LLB	+2H-Opt–LLB
NN	2-Opt	45.16682	28.47246	27.90070	24.94015
	2H-Opt	36.23844	29.51486	29.96878	27.25093
	No Local Opt	100.71020	30.39483	32.65111	28.50908
MF	2-Opt	31.97162	29.02252	30.72352	28.54207
	2H-Opt	27.20955	27.38481	29.93658	26.68985
	No Local Opt	37.76638	29.27886	30.33885	27.25459

Table 3.8: Summary of solution gaps, expressed in km, for a scan depth of 1.0.

IFS	Local Opt (LO) Heuristic	Init Feas Soln– Lagrangian LB	Christofides Feasible Solution		
			+No LO – LLB	+2-Opt–LLB	+2H-Opt–LLB
NN	2-Opt	8.983606	5.640020	5.548835	4.965086
	2H-Opt	7.223588	5.836850	6.003211	5.440751
	No Local Opt	20.012523	6.048354	6.537386	5.704409
MF	2-Opt	6.367048	5.752318	6.163840	5.709083
	2H-Opt	5.440552	5.403467	5.963251	5.323017
	No Local Opt	7.455047	5.783272	6.051835	5.450134

Table 3.9: Summary of solution gaps, expressed in %, for a scan depth of 1.0.

3.8.3.2 Analysis

Better insight into the relative performance of the different combinations of initial and final heuristics is provided by the charts in Figures 3-23 and 3-24 for a scan depth of 5.0 and Figures 3-25 and 3-26 for a scan depth of 1.0; they are a graphical display of the information in Tables 3.7 and 3.9. Considering first Figure 3-23, the average solution gaps (in percentages) are grouped by the heuristic used to construct the initial feasible solution. Within each grouping, then, is given the ultimate gap produced by the different final heuristics. Note that the label "Initial Feasible Solution−LLB" refers to the gap between the grouping heuristic and the Lagrangian lower bound. Thus, for example, "Init Feas Soln−LLB" in the "MF + 2H-Opt" group refers to the gap for the "MF + 2H-Opt − LLB" combination. Similarly in Figure 3-24, where the grouping is done by the final heuristic employed, the individual entries in the "Init Feas Soln−LLB" category are for gaps between the 6 possible initial heuristics and the Lagrangian lower bound.

Most telling are Figures 3-24 and 3-26. Regardless of the initial heuristic employed, if the Christofides-type algorithm is then used and followed by two-H-Opt, the solution gap will be small. Among the heuristics used to construct an initial feasible solution, the best alternative is the multiple fragment heuristic followed by two-H-opt. The one clear poor choice is to construct a nearest neighbor path and then perform no local optimization. The charts also show that after computing a Christofides feasible solution, there is relatively little value added to then processing it with two-opt, and for certain combinations two-opt worsens the the gap. The reason that this can occur is because the Christofides feasible solution constructed is first a tour, and two-opting it serves to remove long crossing (non-planar) arcs that would otherwise be removed in extracting a path.

Boxplots of the distribution of solution gaps, in percent, for 50 problem instances with a scan depth of 5.0 are shown in Figures 3-27, 3-28, 3-29, and 3-30. They are grouped by the final heuristic used, with Figure 3-27 reflecting the gap between the

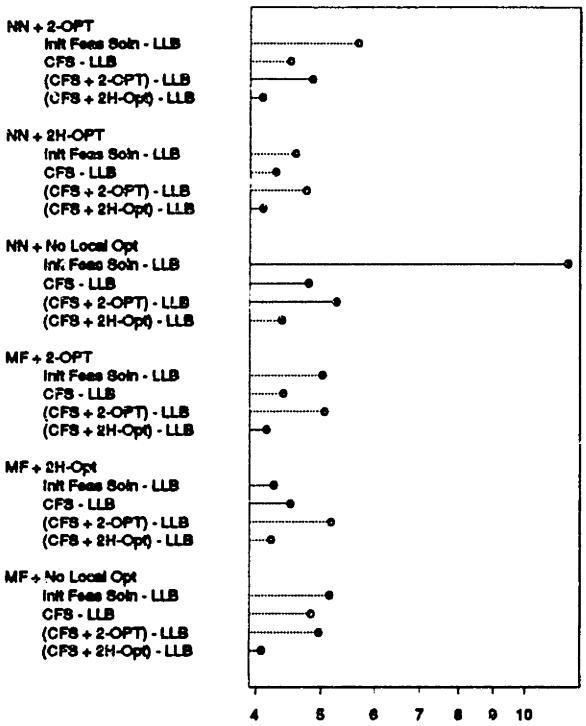

Figure 3-23: Solution gaps in %, grouped by initial heuristic; scan depth = 5.0.

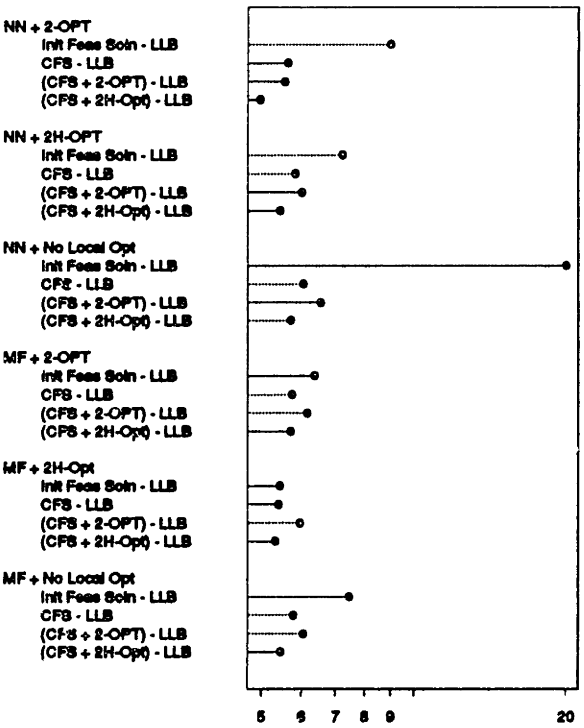

Figure 3-24: Solution gaps in %, grouped by final heuristic; scan depth = 5.0.

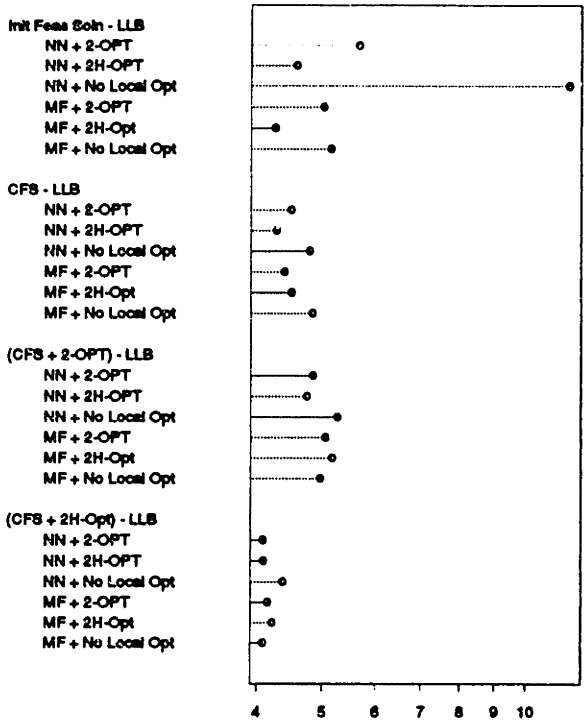

Figure 3-25: Solution gaps in %, grouped by initial heuristic; scan depth = 1.0.

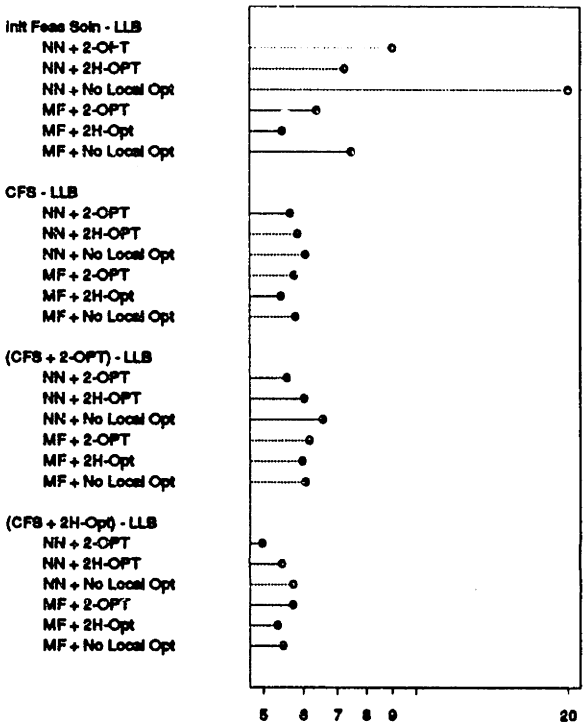

Figure 3-26: Solution gaps in %, grouped by final heuristic; scan depth = 1.0.

initial feasible solution and the Lagrangian lower bound. In comparing the figures, the *scale* should be noted, especially for the first one. The gaps produced by the initial heuristics are large in comparison to the others, with one instance being on the order of 40%. So although the spread in Figure 3-27 may initially *look* tighter, that isn't actually the case. The distributions for all of the heuristics involving a Christofides-type feasible solution are generally comparable, although the one producing the best solutions, the Christofides-type heuristic followed by two-H-opt local optimization, also has the closest spreads.

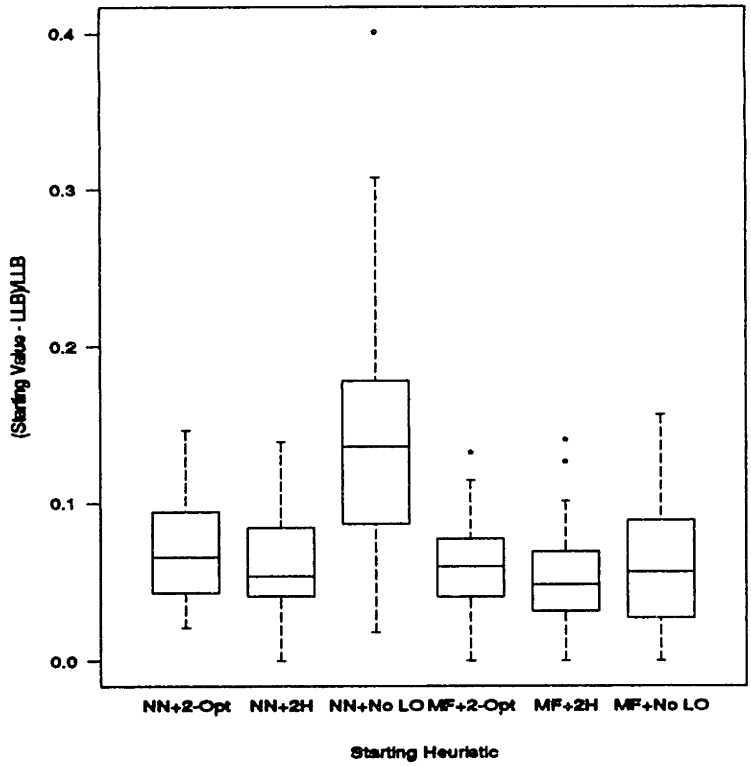

Figure 3-27: Solution gap boxplot for 50 instances of starting heuristics.

As can be seen in the boxplots, there are problem instances for which the solution gap is zero—this corresponds to when the Lagrangian lower bound problem produces the optimal solution. For the scan depth of 5.0, there were from 13 to 16 instances in which the optimal solution was found, depending on the heuristic used to calculate the initial upper bound. When the scan depth dropped to 1.0, though, there was at most one problem for which the shortest path was determined. Given that the regions

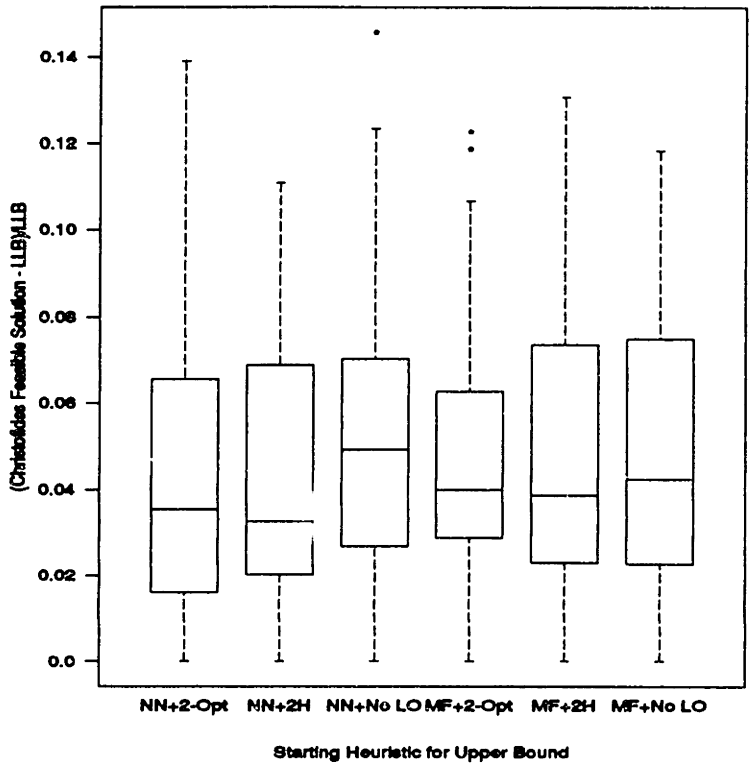

Figure 3-28: Solution gap boxplot for 50 instances of CFS final heuristic.

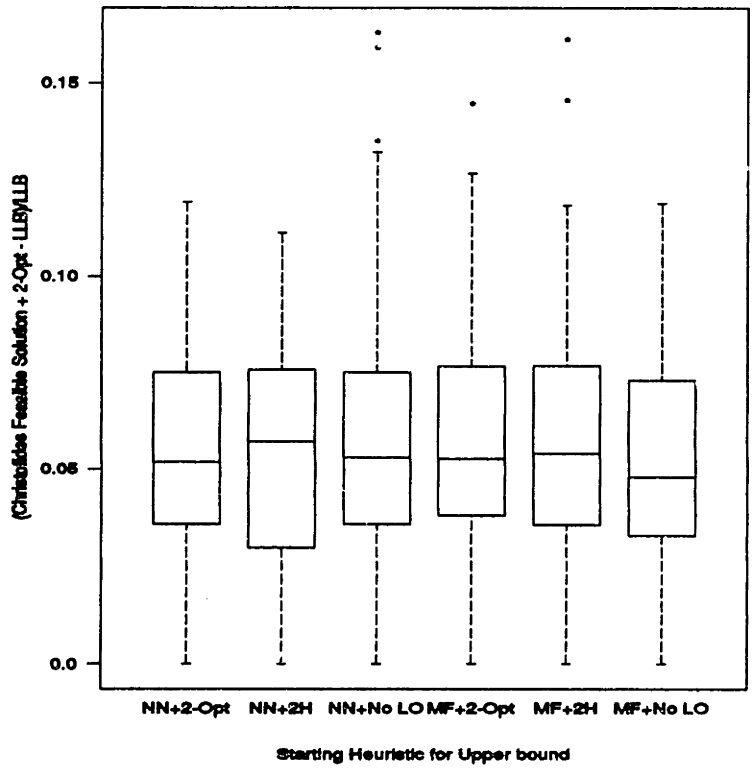

Figure 3-29: Solution gap boxplot for 50 instances of CFS followed by 2-Opt.

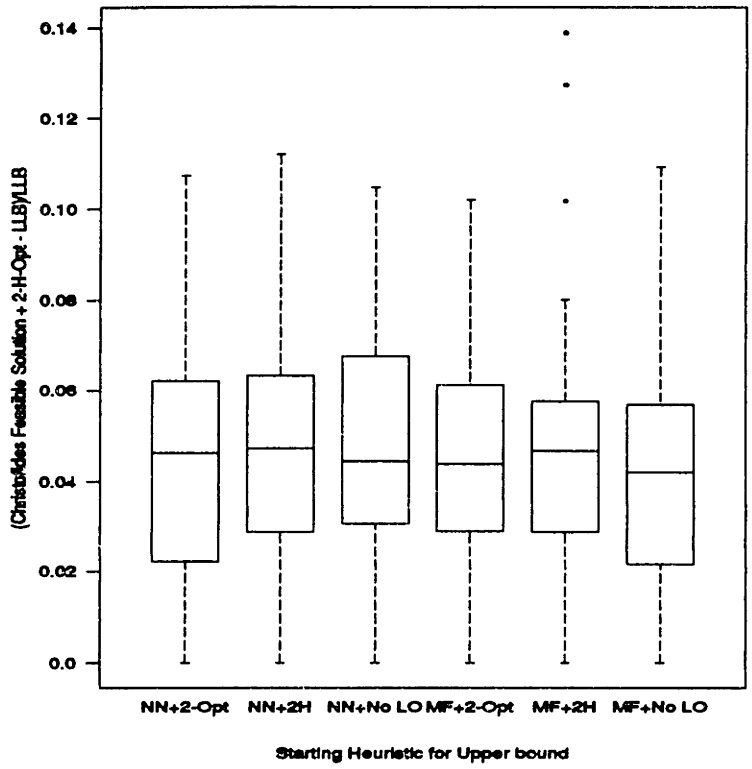

Figure 3-30: Solution gap boxplot for 50 instances of CFS followed by 2H-Opt.

scanned were the same for both scan depths, a smaller scan depth yields more scan points: an average of 61 for the 100 instances of 5.0, and 282 for the same instances with 1.0. We established a rough upper bound of 300 on the number of scan points that could be sequenced in a single project. Considering settling time alone, $300 \times .5$ seconds per movement requires 2.5 minutes, and five minutes is about as long as an imaging time window will last.

Chapter 4

Conclusions & Future Directions

4.1 Summary

In Chapters 2 and 3 we studied two classes of models that can be used to manage satellite-based sensors. A static, coverage oriented model that focusses on footprint placement was treated first, and then scan sequencing for area imaging projects was considered. In both cases the issue of practical concern was modeled, algorithms were developed, and the solution approaches then computationally tested against representative problem instances.

It was also discussed in Chapter 2 how the static coverage model could be used in a rolling horizon context to determine dynamic sensor assignments. Current assignments based on the best known values of targeted objects would be made by the static model, with future assignments (based on current estimates of demand values) being held tentative. In the next period, assignments for that period would be updated and made firm, with a new set of tentative assignments made for the time period added to the horizon. Tentative assignments for intervening periods could be updated.

The scan sequencing models of Chapter 3 can similarly be used in a broader context: project selection and scheduling for an instrument. In the following section we discuss the issues of interest, using as an example the Eos sensor HIRIS, and give a

possible formulation that extends an existing model so as to capture a greater number of resource constraints.

4.2 Project Selection and Scheduling

4.2.1 Background

The various projects that prospective users will request for any kind of space-based sensor will easily exceed the observation capacity of a reasonable planning horizon. This will remain true for Eos sensors such as HIRIS, which are intended to acquire at least a decade of overlapping observations. As a result, efficient management of such sensors so as to maximize scientific use is an obvious goal.

The development of observation schedules has traditionally been a large and complex task. Only a subset of all the requests that are submitted can be performed, and the ones selected must comply with myriad constraints. The restrictions include such things as platform orbital characteristics, instrument power requirements, thermal dissipation, sensor capabilities, viewing conditions, guidance requirements, limited data bandwidth, and preferences for when and how measurements of certain phenomena should be made. Constraints arise from the activities necessary to perform an observation, resource sharing considerations with other instruments on the satellite, and mission objectives. Scheduling constraints imposed by science objectives are an example of the latter. Numerous relationships among observations may be specified, including partial orderings over sets of observations, temporal separation constraints between ordered observations, temporal grouping constraints over unordered observations, coordinated parallel observations with different viewing instruments, same sensor orientation constraints for repeat observations, and conditional execution of dependent observations [82].

It may well not be possible to satisfy all problem constraints, in which case some may have to be selectively relaxed, or specific projects or project combinations de-

ferred to a later schedule. Some constraints, such as time windows for measurements, must be met. The general objective for the project selection and scheduling problem is to perform as much "science" as possible, subject to insuring feasibility with respect to a complex set of constraints involved with instrument operation, image execution, and overall resource allocation objectives.

Specific instrument scheduling is similar to space mission scheduling [4], [17],[18], [38], [62], in which the same basic decisions of selecting and scheduling science projects must be made. The Jet Propulsion Laboratory (JPL) has performed mission scheduling for many years for a variety of deep space flight projects. The effort in scheduling an entire project such as Voyager can be measured in man-centuries. Because of the cost and time involved, JPL and NASA have recently used a variety of knowledge-based scheduling engines that have gone by names such as *Deviser, Plan-It, Switch,* and *Ralph.* There are also other program specific scheduling systems, such as the Science Operations Ground System (SOGS), which supports scheduling for the Hubble space telescope. It is a $70 million fortran-based software system developed by TRW. Generating a workable schedule using such systems, in addition to being an extremely long and costly process, still cannot consistently produce an agenda that is "optimal"—whatever that may mean in this context. Unanticipated constraints cannot easily be accommodated, and many sources of scheduling flexibility that might be permitted by a less comprehensive approach cannot be exploited.

The claim is made that classical optimization techniques are unable to deal with scheduling problems for which the scope is this large and complex [18]. An alternative modeling approach, which would use such techniques, considers only a subset of the constraints involved [57]. Again using the example of HIRIS, we discuss elements of a problem formulation that address scheduling projects during the appropriate time window, and that include restrictions on data bandwidth, power, and thermal dissipation—these being the limiting resources in the operation of HIRIS that JPL has identified. Given that adaptive scheduling of HIRIS is desirable so that targets of

opportunity such as volcanos or fires can be imaged, a model that can quickly generate high quality solutions with respect to changing objectives and possibly resource constraints is potentially quite useful.

4.2.2 Modeling Issues

Rather than give a mathematical formulation of project selection and scheduling for an instrument, we quantify somewhat the salient concerns that must be addressed, introducing notation and showing how the issues can be captured analytically.

A simplified version of the project selection and scheduling problem has aspects that are similar to machine scheduling, a classic operations research problem. Each project j takes time p_j to complete, a "processing" time. The processing time would be determined by using a scan sequencing algorithm to compute how long it takes to perform a specified job—at the detailed scan level, as was done in Chapter 3. In addition, a project will be characterized by the amount of data it produces, as well as its power requirements. It is assumed that the number of proposed projects n cannot all be performed over the planning horizon of interest due to time and resource restrictions.

As stated earlier, the general objective is to accomplish as much "science" as is possible in the time available. For a sensor such as HIRIS, the planning horizon would probably be on the order of 30 days. This objective can be mapped into the specific goal of maximizing the weighted value of the projects that are selected and scheduled. If all jobs are considered to be of equal value, the scheduling component reduces to minimizing the maximum completion time amongst the jobs selected. Because the EOS platforms will be in low-earth, sun-synchronous orbits (705 km), various projects can only be performed during specific time windows, which can be designated as $[r_j, d_j]$ for project j. Thus each project can begin only during the interval $[r_j, d_j - p_j]$.

Over a thirty day period, there will be at least 15 opportunities to conduct each imaging project, and possibly more, depending on the latitude of the target (equato-

rial latitudes are least frequently available, and those greater than $\pm 40^{\circ}$ or more most frequently available). Not all viewing opportunities will be equally desirable, though, because they may present different viewing angles. As an example, viewing vegetation from an off-nadir angle can provide a better estimate of the total albedo of the canopy. This is because the angular distribution of reflectance from vegetation is sensitive to leaf density, leaf orientation, and the distance between plants [51]. Different windows for the same project can represent different jobs; this allows the assignment of various weights to the disparate opportunities, reflecting the preferences between viewing angles. A constraint that only one of these imaging tasks is to be scheduled can be imposed. Current JPL planning calls for each project to be scheduled *twice*, to increase the chance that acceptable viewing conditions are obtained.

It may also be the case that projects will consist of linked or repeat observations. The frequency requirements of coverage flow directly from the nature of the science area to be studied. Water and snow/ice studies need frequent measurements, while vegetation studies require less frequent ones. Geological studies are a distant third, but must acquire data under optimal lighting and visibility conditions and when the area of interest is neither covered by vegetation nor obscured by clouds. A penalty function could be employed to drive imaging assignments toward meeting observation frequency preferences.

Additional notation should help clarify these ideas. It is intended only as a means of representing elements of project selection and scheduling problem in a coherent fashion; these are not propositions that are likely to be solved directly as mathematical programs. Some of our notation is adopted from [57]. We define the following variables:

$$y_j = \begin{cases} 1 & \text{if job } j \text{ is performed, } j = 1, \ldots, n \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if job } i \text{ is performed before job } j, \\ & \text{or job } j \text{ is performed and job } i \text{ is not, } i,j = 1,\ldots,n,\ i \neq j \\ 0 & \text{otherwise} \end{cases}$$

t_j = the start time of job j if job j is performed, $j = 1,\ldots,n$.

The objective in optimizing the selection and scheduling of scans for HIRIS becomes:

$$\text{Maximize} \sum_{j=1}^{n} w_j y_j \,, \tag{4.1}$$

where w_j is the weight of project j, $j = 1,\ldots,n$. This, of course, is subject to many constraints, the most rigid one being that of time windows. Those constraints that can be selectively relaxed may also be placed in the objective (4.1), using a penalty or a barrier function approach [75]. These are methods for approximating a constrained optimization problem with an unconstrained or less constrained one. The approximation is accomplished with a penalty function by adding to the objective function a high cost for violating the constraints, and in the case of a barrier method by adding a term that favors solutions in the interior of the feasible region over those near the boundary.

A project can only begin during an interval within which the necessary geometry between the earth and the satellite is achieved, and must begin early enough so that it can be completed. This can be stated as

$$t_j \geq r_j y_j, \quad j = 1,\ldots,n \tag{4.2}$$

$$t_j \leq (d_j - p_j)y_j + D(1 - y_j), \quad j = 1,\ldots,n \,, \tag{4.3}$$

where D is the planning horizon. We assume that only *one* project can be performed at a time. If two requests are combinable, than they could be merged before the planning process began. Thus when two projects i and j are scheduled, there must be

sufficient time between between their starts so that the first one can be completed, and then whatever repositioning of the sensor that needs to take place can be performed:

$$t_i - t_j \geq (p_j + c_{ij})y_j - 2Dx_{ij}, \quad i = 1, \ldots, n-1,\ j > i \tag{4.4}$$

$$t_j - t_i \geq (p_i + c_{ji})y_i - 2Dx_{ji}, \quad i = 1, \ldots, n-1,\ j > i\,, \tag{4.5}$$

where c_{ij} is the minimum transit time needed if project i follows project j. Only one of the two constraints (4.4) and (4.5) can be active; there are standard ways of enforcing this.

The constraints of data bandwidth, power consumption, and thermal dissipation can all be handled in a similar manner. Essentially, the instrument will have a budget for each such resource available to it; we'll let BW be the bandwidth available during the planning time horizon. Then

$$\sum_{j=1}^{n} b_j y_j \leq BW\,, \tag{4.6}$$

where b_j is the bandwidth required by project j, is the appropriate budget constraint. This can be done analogously for thermal dissipation and power. Equation (4.6) is a knapsack constraint, and several such constraints form a multidimensional knapsack. As an alternative to posing available bandwidth BW as a strict bound, we could also take the approach of considering it a relaxable limit and attaching a penalty, say b, to exceeding it. Then the term

$$-\max\left(b \times \left(BW - \sum_{j=1}^{n} b_j y_j\right),\ 0\right) \tag{4.7}$$

could be added to the objective function. For HIRIS, the budget set by NASA on the amount of data that can be produced is very limiting. The long-term average of 3 mbps is less than a third of what the instrument could reasonably generate, even after accounting for both viewing opportunities and sensor movements. The 3 Mbps data

rate limitation is due to HIRIS having to share the 300 Mbps Eos Data Information System with numerous other instruments.

User preferences for multiple viewing opportunities or linked observations are also easily depicted. When at most a limited number of several project opportunities are to be scheduled, the constraint

$$\sum_{j \in A_k} y_j \;\leq\; \xi_k, \quad k = 1, \ldots, K\,, \tag{4.8}$$

captures this. ξ_k is the upper bound on the number selected; A_k is a set containing the indices j of the projects that represent multiple opportunities to perform an imaging request, and for which there are $k = 1, \ldots, K$, different such groupings. A penalty function that subtracts $\bar{\beta}_j$ from w_j probably best shows linked observations as being preferential, rather than strict. If we let

A_l = set of project indices that constitute a group of linked observations, $l = 1, \ldots, L$,

ρ_l = periodicity of the set A_l, $l = 1, \ldots, L$,

β_l = penalty for violating the periodicity preference for set A_l, $l = 1, \ldots, L$,

t_l^* = time of last scan from set A_l, $l = 1, \ldots, L$,

then

$$\bar{\beta}_j = \beta_l \,|\, t_j - t_j^* - \rho_l|, \; j \in A_l, \; l = 1, \ldots, L\,, \tag{4.9}$$

represents the penalty for deviating from the stated preference ρ_l. The objective function (4.1) is now $\sum_{j=1}^{n}(w_j - \bar{\beta}_j)y_j$. As $\bar{\beta}_j$ is defined, there is a penalty for observing a site either too often or too infrequently; this term could be redefined as

$$\bar{\beta}_j = \beta_l \times \left(\max\left(t_j - t_j^* - \rho_l\right), 0\right), \quad j \in A_l, \; l = 1 \ldots, L\,, \tag{4.10}$$

so that a penalty is incurred only for imaging too infrequently.

The modeling ideas embraced by (4.1)-(4.10) follow the same vein as the approach taken by Hall and Magazine [57]. In their work they considered only the resource of time and allowed each project to be performed solely during a single time window, while we include additional resources such as data bandwidth and provide for multiple opportunities to conduct an imaging project. In addition, science imposed preferences regarding the temporal distribution of linked observations are considered. The model of [57] has been shown to be strongly $\mathcal{NP}$-complete by transformation from 3-PARTITION. Thus by restriction, the model we propose is $\mathcal{NP}$-complete and unless $\mathcal{P} = \mathcal{NP}$, it is unlikely that there exists an optimal polynomial time algorithm for its solution.

There has been work done on other, similar scheduling problems. Traditional paradigms include the problems of minimizing the total time needed to complete n jobs within available time intervals on parallel processors, minimizing the number of processors needed to complete all jobs, and vehicle routing and scheduling problems with time windows, which we referred to at the end of Chapter 2. In all cases, however, every task must be completed; there is no notion of choosing amongst those requested.

4.2.3 Model Analysis

Numerous algorithms have been developed for finding solutions to various scheduling problems. Several of the same approaches that we used earlier have been exploited to routinely deliver solutions that are very close to optimal. The challenge here lies in specifying a sufficiently tractable problem formulation that combines increased modeling fidelity with responsiveness to analysis. As an indication that the added scheduling dimensions are manageable, we show how the interval pricing heuristic of [57] can be extended to the problem we have framed. Further analysis, such as upper bounding procedures, Lagrangian relaxation, dynamic programming, or the computational testing of any of these approaches, should be the subject of future

research.

The interval pricing heuristic essentially computes a "price" for each discrete interval of time, and then selects a project based on whichever offers the best value versus cost benefit. The project chosen depends upon both scientific weight and resource consumption. If two projects are of equal value, it would be better to choose first the one that consumes less of the resources available (here we are considering such things as time, bandwidth, and power). In the case of a single resource problem, a natural choice for an index of aggregate consumption is merely the amount of that resource used. The choice is less clear when there are a multiplicity of resources.

An approach for computing a "penalty" factor for resource consumption is described by Loulou and Michaelides [74]. It is argued that project selection should be governed by the worst effect that selection of project j would have on the following:

1. total consumption of resource q if project j is added to the set of projects already selected.

2. amount of resource q remaining if project j is added to the set of projects already selected.

3. future potential demand for resource q if project j is added to the set of projects already selected.

A penalty factor for project j that incorporates these [74] is:

$$V_j = \max_{q=1,\ldots,Q} \left\{ (\lambda_q + \pi_{qj})(\sum_{s \in \mathcal{S}} \pi_{qs} - \pi_{qj})/(1 - \lambda_q - \pi_{qj}) \right\} \tag{4.11}$$

where

Q = number of resources

λ_q = fraction of resource q consumed so far, $q = 1, \ldots, Q$

π_{qj} = fraction of resource q that project j would consume, $q = 1, \ldots, Q$, $j = 1, \ldots, n$

S = set of candidate projects.

An alternative to (4.11) that has been used is

$$V_j = \sum_{q \in Q} \pi_{qj} \lambda_q / \| \lambda \| \tag{4.12}$$

where λ is the Q dimensional vector of λ_q's, and $\| \cdot \|$ is the L_2 norm.

A shortcoming of (4.11) is that as a resource nears depletion, V_j can become numerically unstable; the denominator can near zero while the numerator remains large. There are two approaches to reducing this problem. The first decreases the importance of the ratio in (4.11) by taking its square root (or some other fractional power):

$$V_j = \max_{q=1,\ldots,Q} \left\{ (\lambda_q + \pi_{qj})(\sum_{s \in S} \pi_{qs} - \pi_{qj})^{1/2} / (1 - \lambda_q - \pi_{qj})^{1/2} \right\}. \tag{4.13}$$

A second approach is to use (4.11) or (4.13) until $\max_q \lambda_q$ is within some specified tolerance of 1, and from then on select projects based upon their weight alone. This is referred to as a *switch* in [74].

We incorporate (4.11) into a marginal return heuristic for the time intervals.

Interval Pricing Heuristic :

1. Initialize.

   (a) $S := \{1, \ldots, n\}$. (The set of unselected projects.)

   (b) $T_j := \{1, \ldots, D\}$, for $j = 1, \ldots, n$. (Time intervals available for project j.)

   (c) $J := \phi$. (Set of projects that have been scheduled.)

2. Compute costs.

   (a) $\lambda_1 := \sum_{j \in J} p_j / D$; $\lambda_2 := \sum_{j \in J} b_j / BW$.

   (b) $\pi_{1j} := p_j / D$, for all $j \in S$; $\pi_{2j} := b_j / BW$, for all $j \in S$.

(c) $V_j := \max_{q=1,2} \{(\lambda_q + \pi_{qj})(\sum_{s \in \mathcal{S}} \pi_{qs} - \pi_{qj})/(1 - \lambda_q - \pi_{qj})\}$, for all $j \in \mathcal{S}$.

(d) $U_j := w_j/V_j$, for all $j \in \mathcal{S}$. ("Utility" of project j.)

(e) $\Gamma_i := \sum_{j \in \mathcal{S} | r_j \leq i, d_j > i} U_j$, for all $i \in T_j$. (Value of time interval i.)

(f) $\mathcal{F}_j := \min_{\substack{\tau \in [r_j, d_j - p_j] \\ \{\tau, \ldots, \tau + p_j - 1\} \in T_j}} \sum_{i=\tau}^{\tau + p_j - 1} \Gamma_i$, for all $j \in \mathcal{S}$. (Value of the minimum cost time slot for project j.)

(g) $\mathcal{Z}_j := \operatorname{argmin}_{\substack{\tau \in [r_j, d_j - p_j] \\ \{\tau, \ldots, \tau + p_j - 1\} \in T_j}} \sum_{i=\tau}^{\tau + p_j - 1} \Gamma_i$, for all $j \in \mathcal{S}$. (Start time for the minimum cost time slot for project j; its cost is $\mathcal{F}_j$.)

3. Select next project: η.

(a) $\Omega := \mathcal{S}$. (Set of unselected projects.)

(b) $\eta := \operatorname{argmax}_{j \in \Omega} U_j/\mathcal{F}_j$. (Project with the highest utility to cost ratio.)

(c) If $\{\mathcal{Z}_\eta - c_{j\eta}, \ldots, \mathcal{Z}_\eta - 1;\ \mathcal{Z}_\eta + p_\eta, \ldots, \mathcal{Z}_\eta + p_\eta + c_{\eta j} - 1\} \in T_\eta$ for all $j \in J$, then select η and go to step (4); otherwise $\Omega := \Omega \backslash \{\eta\}$ and $T_\eta := T_\eta \backslash \{\mathcal{Z}_\eta\}$.

(d) If $\Omega \neq \phi$, go to step 3b.

(e) If $\Omega = \phi$, got to step 4e.

4. Update and iterate.

(a) $\mathcal{S} := \mathcal{S} \setminus \{\eta\}$. (Remove selected project.)

(b) $T_j := T_j \backslash \{\mathcal{Z}_\eta, \mathcal{Z}_\eta + 1, \ldots, \mathcal{Z}_\eta + p_\eta - 1\}$ for all $j \in \mathcal{S}$. (Delete newly assigned time intervals.)

(c) $J := J + \{\eta\}$.

(d) If $\eta \in A_k$ for some k^*, then $\mathcal{S} := \mathcal{S} \setminus \{j \mid j \in A_{k^*}\}$. (Drop any projects that are multiple opportunities for the same request.)

(e) If there exists no $\mathcal{Z}_j \in [r_j, d_j]$ such that $\{\mathcal{Z}_j - c_{ji}, \ldots, \mathcal{Z}_j, \ldots, \mathcal{Z}_j + p_j - 1, \ldots, \mathcal{Z}_j + p_j + c_{ij} - 1\} \in T$ then $\mathcal{S} := \mathcal{S} \setminus \{j\}$, for all $j \in \mathcal{S}$, $i \in J$.
(Check to see if any projects are now infeasible.)

(f) If $\mathcal{S} \neq \phi$, return to step 2 and iterate.

(g) If $\mathcal{S} = \phi$, stop. (There are no projects left.)

Project weights that vary with the time interval (e.g., $w_j - \bar{\beta}_j$, where $\bar{\beta}_j$ is a function of the time interval) can be incorporated by computing U_{ij} rather than U_j for all $i \in T_j$, $j \in \mathcal{S}$. Γ_i, $\mathcal{F}_j$, and $\mathcal{Z}_j$ would then be calculated as before. An alternative to selecting the next project in step 3b based on the ratio of project weight to index of resource consumption is to use the difference between the two: $\eta := \text{argmax}_{j \in \Omega}(U_j - \mathcal{F}_j)$. In addition, computational experience may indicate a need to use (4.13) for V_j in step (2c) of the heuristic, or to employ a switch to selection based on pure project weights [74].

4.3 Conclusions

In this thesis we have considered several classes of models for the management of satellite-based sensors. In particular, we have studied sensor coverage and scan sequencing problems. Although there are many potential applications, the ones that motivated our work are those of strategic surveillance and environmental observation. Our intent was to provide an intuitive understanding of the problems and their solutions. A similar approach can be used to gain insight for the further research directions that have been identified. To the extent that such models and their analysis provide usable solutions to realistic problems, or even a better understanding of how to construct them, then they are a contribution to the effective use of resources that are likely to be even more heavily employed in the future than they have been in the past.

Appendix A

HIRIS

A.1 Introduction

Traditionally, scientists within different disciplines related to the study of the Earth have worked largely independent of one another. No real attempt has been made to integrate the various models that have been developed through the study of different components such as the atmosphere, soils, vegetation, and oceans. A recent thrust, however, is to view the Earth as an integrated system. Such a perspective requires a remote sensing system that can collect data about the Earth's surface, oceans, and atmosphere over a range of scales. An integrated measurement capability, as will be provided by the Earth Observing System (Eos), can acquire data throughout the electromagnetic spectrum for the purpose of characterizing various Earth system processes. An understanding of the dependencies involved between different disciplines, particularly between those of different scales, can enhance our ability to predict environmental change. The study of specific processes will require the collection of remote sensing data at a range of spectral, spatial, and temporal resolutions.

The High-Resolution Imaging Spectrometer (HIRIS) [36], [53], [83], [89] is one of the highest spatial and spectral resolution instruments that will be deployed as part of the Earth Observing System . It will obtain simultaneous images in 192 spectral

bands in the range .25–4.0 μm, with a sampling interval of 10 nanometers. The spatial resolution is 30 meters, and an instantaneous image will consist of 800 adjacent pixels which produces a 24 kilometer wide swath. This image can be made within the region coverable by pointing the sensor up to $\pm 45°$ cross-track (i.e., orthogonal to the direction of travel of the satellite), and $+60°/-30°$ along-track. Cross-track pointing allows for the measurement of reflectance properties under different atmospheric attenuation conditions, as well as providing for more frequent opportunities to image a particular spot.

Imaging spectrometry is the measurement of electromagnetic radiation from the sun that is reflected by the surface of the earth. The ability to acquire data with the precision offered by HIRIS will improve our ability to make a whole host of measurements that are not currently possible with the current generation of broadband space-based sensors. Examples are the surface mapping of geological deposits based upon the spectral signature of constituent minerals and soils, and the characterization of vegetation, coastal and inland waters, and alpine snow and ice.

HIRIS represents the finest scale of measurements that are to be made among the suite of instruments envisioned for the Earth Observing System. That, however, places it in an intermediate position in a comprehensive data collection scheme between *in situ* measurements made by a person on the ground and those made by moderate and broad band sensors that map the surface of the globe every few days. HIRIS will be complemented by several instruments in the Eos program, such as the Moderate-Resolution Imaging Spectrometer (MODIS) in the visible, near-infrared, and thermal infrared wavelengths, as well as by the passive microwave instruments that compose the High-Resolution Multi-frequency Microwave Radiometer (HMMR). There are other instruments whose high-resolution measurements will add to those made by HIRIS in the visible and near-infrared wavelengths. They include a Synthetic Aperture Radar (SAR) in the active microwave and a Thermal Infrared Multispectral Scanner (TIMS). Such instruments permit analysis and interpretation of data from

specific regions covered by broader swath instruments.

A.2 Heritage of HIRIS

Previous instruments provide the scientific and engineering experience necessary to develop an instrument such as HIRIS with an acceptable degree of technical risk. These instruments fall into two categories: previous spaceborne sensors with high spatial resolution but low spectral resolution, and airborne imaging spectrometers that possess both high spatial and spectral resolution characteristics.

As discussed earlier, the Landsat Thematic Mapper (TM), which was first fielded in 1982, has relatively high spatial resolution in seven spectral bands. It has been of considerable scientific benefit, primarily in the areas of cloud studies, snow-cloud discrimination in mountainous areas, detection of stress in vegetation due to dehydration, and the identification of some minerals whose reflection characteristics fall within the spectral bands of the TM. The High Resolution Visible Instrument on the French satellite SPOT has 20 meter spatial resolution in three spectral bands, and 10 meter resolution in one panchromatic band. It is capable of being pointed cross-track up to 27°, which allows for multiple views within a single orbital revisit cycle.

There are two sets of airborne instruments, the first of which is the Airborne Imaging Spectrometers (AIS-1 and AIS-2) [109] that were used from 1983 to 1987. They had 10 meter spectral resolution in 128 spectral bands for images that were 32 and 64 pixels wide. Their principal use was for the testing of area-array infrared detectors, but some scientific results in geology and vegetation were provided. The successor to the AIS program is the Airborne Visible and Infrared Imaging Spectrometer (AVIRIS) [110], which has been in use since 1987. It covers the same spectral range as is proposed for HIRIS, with spatial resolution of 20 meters. Testing with AVIRIS will establish the potential benefits of HIRIS.

Potential scan sequencing strategies for the airborne imaging spectrometers are

much broader than they are for space-borne sensors. Fewer mechanical restrictions due to sensor maneuvering are imposed on the selection and ordering of images, and there are no time windows with which to be concerned. It also isn't necessary to point the sensor cross-track for multiple views, since all that is needed is to fly back over the desired area. A space-based sensor is much more highly constrained in terms of its movements and the timing of its images. It is likely that future generations of space-based sensors will incorporate more flexibility as technology permits; thus the strategies that we consider encompass a wider spectrum of sequencing options than are strictly permitted for HIRIS.

A.3 Science Objectives

Imaging spectrometry measures the solar radiation that is reflected from the surface of the earth. With high spectral resolution, it is possible to characterize a variety of earth phenomena, several of which we will describe [36], [53], [89].

A.3.1 Mineral Mapping

Over 1000 minerals have unique spectral signatures in the range .25–4.0 μm. As a result, imaging spectrometry can be used to identify and map these minerals in surface rocks and soils. This permits the location of nonrenewable resources, and provides for a better understanding of the processes that have produced these landforms.

The cumulative properties of the combination of minerals, organic material, and water contained in soil permit it, too, to be characterized by its reflectance. The spectral signature will often contain components due to both mineral and vegetation, which it may be possible to separate.

A.3.2 Oceans and Inland Waters

HIRIS has better spectral resolution than do other existing instruments that have been used to study coastal zones and inland waters. It should be possible to separate the different components of near-surface waters. In addition, the near-infrared bands can be used to compensate for atmospheric effects that occur over water with high levels of sedimentation and in nearby land scenes. It may also be possible to study near-shore waters which contain material unrelated to chlorophyll pigments. Increased spatial resolution will aid in the examination of inland waters, which have several distinctive characteristics: relatively small size; long and complex boundaries with land; widely varying optical conditions; and strong horizontal and vertical gradients in their geological, chemical, and physical conditions.

A.3.3 Vegetation

Analysis of vegetation reflectance focusses on a variety of biogeochemcial processes, to include water content of leaves, photosynthetic activity, productivity, and transpiration. Stress associated with excesses or deficiencies in soils that lead to shifts in the chlorophyll absorption spectrum can be measured. On a large scale, the health of regions of vegetation can be related to such parameters as total leaf area or total foliar nitrogen. This is an area where the measurements benefit from HIRIS's ability to point cross-track. Off-nadir viewing provides better information on the total reflectance characteristics of vegetation.

A.3.4 Atmosphere and Snow & Ice

HIRIS is intended primarily for use in observing the surface, however, it can also collect information on both the atmosphere and alpine snow and ice conditions. In the atmosphere this instrument can measure such features as aerosol loading and composition, which is of research use both directly and for correcting images taken over

nearby land masses. Snow reflectance is a function of its grain size and absorbing impurities, attributes that can be estimated by HIRIS measurements. Off-nadir viewing and selected spectral bands will be used to evaluate these characteristics, which are necessary for calculating the surface radiation balance.

A.4 Instrument Design

The design characteristics of HIRIS will reflect the science requirements established in the last section. Because the instrument is intended for targeting rather than for continuous data acquisition, it will obtain data at specific times and places. From earlier experience with the Landsat Thematic Mapper, a spatial resolution of 30 meters was chosen. This will permit study of vegetation systems where changes can occur due to gaps created by tree death, windfall, and land use changes. While even greater resolution for use in mineral mapping is desirable, the disadvantage of an even higher data rate rules it out.

Spectral coverage is determined by science objectives and technological capabilities. The bandwidth selected contains almost all the diagnostic information that can be gleaned from sensing reflected solar energy. Spectral resolution is determined primarily by the needs of water and vegetation studies in the visible and near-infrared bands and mineral mapping in the short-wave infrared. Most minerals, for instance, can be detected by 10 nanometer sampling whereas 20 nanometer sampling would significantly corrupt the ability to identify certain materials.

The frequency with which any given region can be imaged is driven by the character of the phenomena being considered. Water studies call for the greatest frequency of observation and geological studies the least; vegetation studies fall in the middle. All observations, however, require favorable conditions, to include appropriate lighting and the desired presence or absence of vegetation cover. The pointing capabilities of HIRIS, $\pm 45°$ cross-track and $+60°/-30°$ along-track will allow for the re-targeting

of any point on the globe in no more than two days. If the site is located at 40° north or south latitude it could be imaged eight or more times within the 16-day orbital repeat cycle. In addition to permitting more frequent observation, measurements made by off-track pointing are used to correct for atmospheric attenuation and to estimate other reflectance characteristics.

HIRIS's potential raw internal data rate is massive:

$$192 \frac{\text{bands}}{\text{image}} \times 12 \frac{\text{bits/pixel}}{\text{band}} \times 800 \frac{\text{pixels}}{\text{image}} \times \left(4.49 \frac{\text{ms}}{\text{image}}\right)^{-1} = 410 \text{ Mbps/image}$$

Twelve bits per pixel was selected based on the dynamic range and smallest reflectance interval to be observed. Snow is one of the brightest natural substances, so large dynamic range is necessary. It is also desirable to be able to distinguish between very small differences in reflectance, so fine quantization is needed. The signal processing needed for water studies also calls for 12-bit encoding.

The current HIRIS design plan can be summarized in table A.1 (adapted from [53, p. 139]). A more detailed discussion of relevant scan related technical parameters is provided in section 3.1.

A.5 Summary

HIRIS is an environmental sensor that has been developed for high spatial and spectral resolution. Its use will enable the collection of data that cannot currently be gathered using present sensors. It will be a complement to moderate and low resolution sensors by sampling at specific sites within the regions they cover. HIRIS will capture transient processes in the context of a global, multistage remote sensing strategy.

Orbit altitude	705 kilometers
Swath width	24 kilometers
Ground instantaneous field-of-view	30 meters
Spectral Coverage	0.4-2.5 μm, 192 bands
Sampling Interval	
0.4-1.0μm	9.4 nanometers
1.0-2.5μm	11.7 nanometers
Pointing	
Along-track	$+60°/-30°$
Cross-track	$\pm 45°$
Encoding	12 bits/pixel
Data Rate	
Internal	410 Mbps
Instrument to Platform	100 Mbps
Long term	3 Mbps

Table A.1: HIRIS Instrument Design.

Appendix B

Figures of Starting Heuristics

To provide a general sense of how the different starting heuristics produce feasible tours (from which we then extract the shortest path), we show figures that display each of the six different combinations of path forming and local optimization heuristics. Table 3.3 is reproduced here to recapitulate the heuristics used.

Initial Feasible Solutions for OSSP
Nearest Neighbor, followed by 2-Opt
Nearest Neighbor, followed by 2H-Opt
Nearest Neighbor, with no Local Optimization
Multiple Fragment, followed by 2-Opt
Multiple Fragment, followed by 2H-Opt
Multiple Fragment, with no Local Optimization

Table B.1: Combinations of path-forming and local optimization heuristics that provide initial feasible solutions whose length can be used in the subgradient optimization step of the tree relaxation with Lagrangian objective function.

Figure B-1 shows the nearest neighbor heuristic followed by 2-Opt; Figure B-2 nearest neighbor followed by 2H-Opt; Figure B-3 nearest neighbor followed by *no* local optimization; Figure B-4 multiple fragment followed by 2-Opt; Figure B-5 multiple fragment followed by 2H-Opt; and Figure B-6 multiple fragment followed by *no* local optimization.

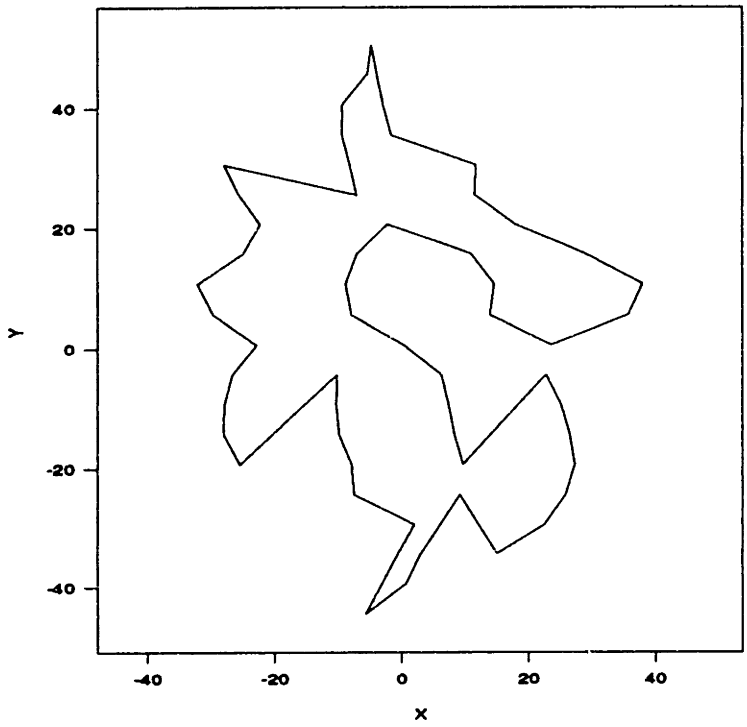

Figure B-1: Tour formed by nearest neighbor heuristic followed by 2-Opt.

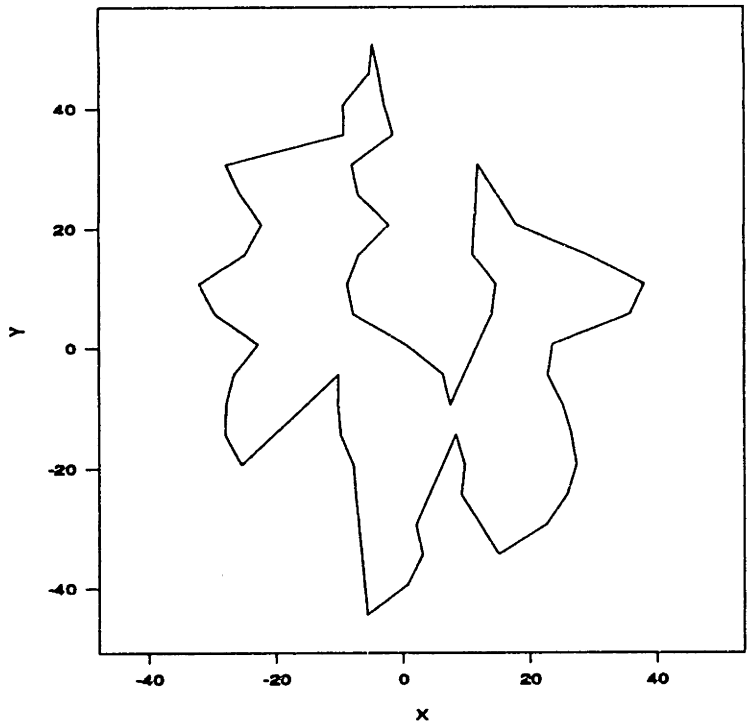

Figure B-2: Tour formed by nearest neighbor heuristic followed by 2H-Opt.

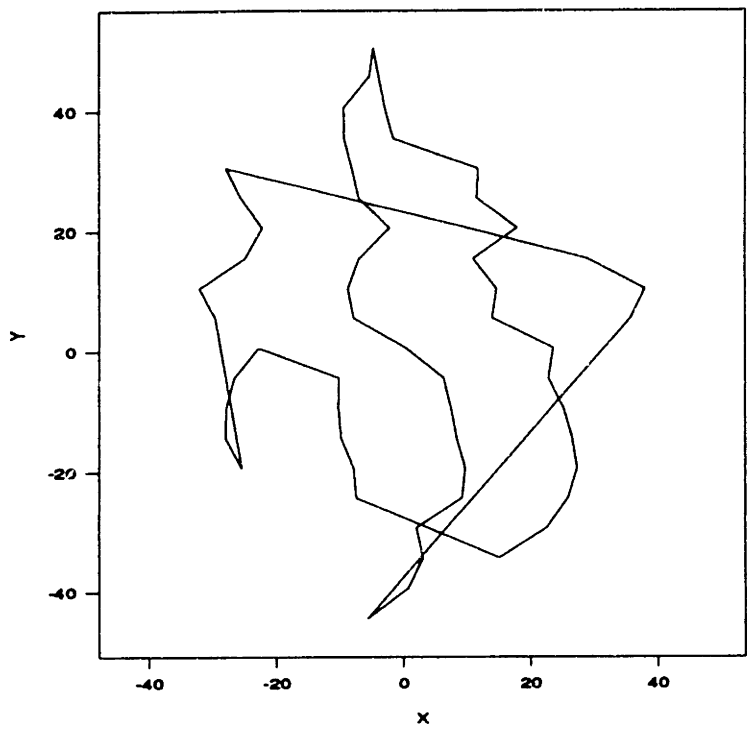

Figure B-3: Tour formed by nearest neighbor heuristic with *no* local optimization.

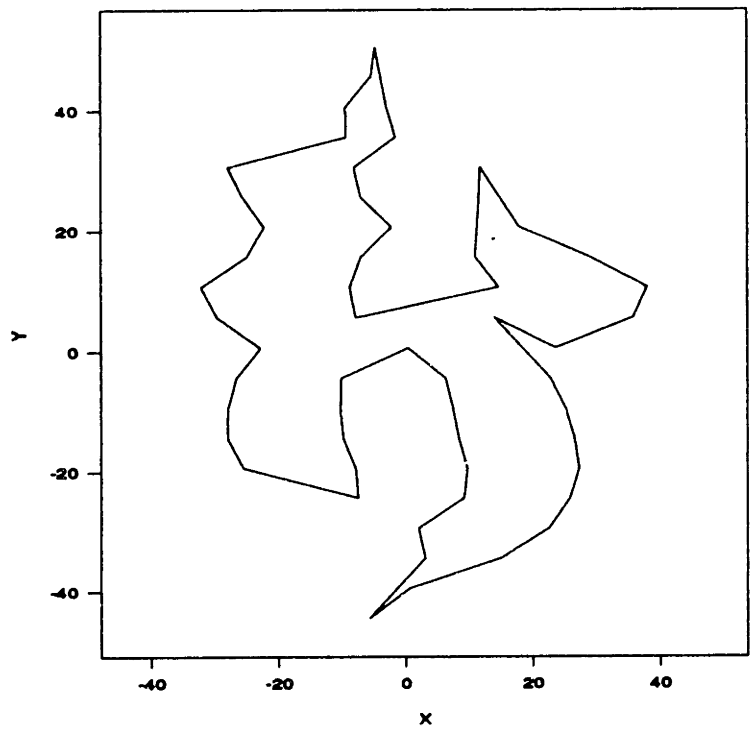

Figure B-4: Tour formed by multiple fragment heuristic followed by 2-Opt.

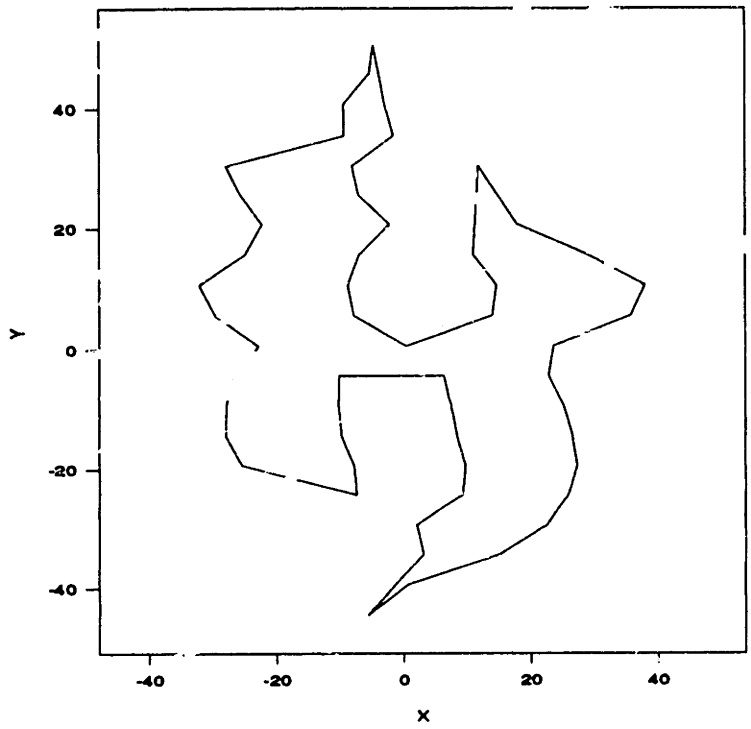

Figure B-5: Tour formed by multiple fragment heuristic followed by 2H-Opt.

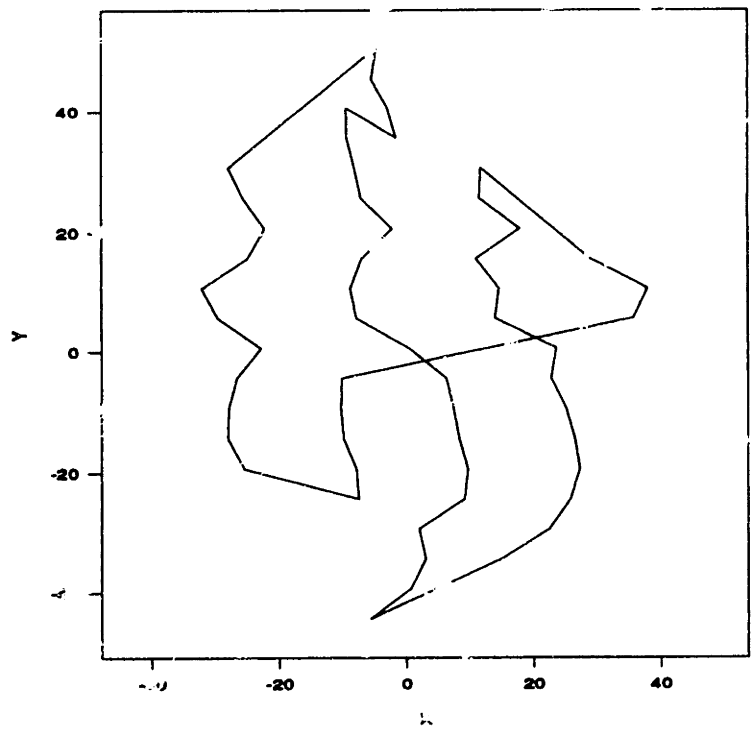

Figure B-6: Tour formed by multiple fragment heuristic with *no* local optimization.

Bibliography

[1] Tetsuo Asano and Takao Asano. Minimum partition of polygonal regions into trapezoids. In *24th Symposium on Foundations of Computer Science*, pages 233–241. IEEE Computer Society, November 1983.

[2] James R. Asker. NASA reveals scaled back plan for six Eos spacecraft. *Aviation Week and Space Technology*, 136(9):61, March 1992.

[3] D. Avis, B. Davis, and J. M. Steele. Probabilistic analysis of a greedy heuristic for Euclidean matching. *Probability in the Engineering and Information Sciences*, 2:143–156, 1988.

[4] K. A. Bahrami, E. Biefeld, L. Costello, and J. W. Klein. *Space Power System Scheduling using an Expert System.* Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109.

[5] E. Balas. Cutting planes from conditional bounds: A new approach to set covering. *Mathematical Programming Study*, 12:19–36, 1980.

[6] E. Balas and A. Ho. Set covering algorithms using cutting planes, heuristics and subgradient optimization: A computational study. *Mathematical Programming*, 12:37–60, 1980.

[7] E. Balas and S. M. Ng. On the set covering polytope: I. All the facets with the coefficients in {0,1,2}. *Mathematical Programming*, 43:57–69, 1989.

[8] E. Balas and S. M. Ng. On the set covering polytope: II. Lifting the facets with coefficients in {0,1,2}. *Mathematical Programming*, 45:1–20, 1989.

[9] A. Ballard. Rosette constellations of earth satellites. *IEEE Transactions on Aerospace and Electronic Systems*, AES-16(5):656–673, September 1980.

[10] J. E. Beasley. An algorithm for set covering problems. *European Journal of Operational Research*, 31:85–93, 1987.

[11] J. E. Beasley. A Lagrangian heuristic for set-covering problems. *Naval Research Logistics*, 37:151–164, 1990.

[12] M. Bellmore and G. L. Nemhauser. The traveling salesman problem: A survey. *Operations Research*, 16:538–558, 1968.

[13] M. Bellmore and H. D. Ratliff. Set covering and involutory bases. *Management Science*, 18:194–206, 1971.

[14] J. L. Bentley. K-d trees for semidynamic point sets. In *Sixth Annual ACM Symposium on Computational Geometry*, pages 187–197, Berkeley, California, June 1990.

[15] J. L. Bentley and J. B. Saxe. An analysis of two heuristics for the Euclidean travelling salesman problem. In *6th Annual Allerton Conference on Communication, Control, and Computing*, pages 41–49, October 1980.

[16] Jon Louis Bentley. Experiments on geometric traveling salesman heuristics. Technical Report 151, AT&T Bell Laboratories, Murray Hill, NJ 07974, August 1990.

[17] Eric W. Biefeld. *PLAN-IT: Knowledge-Based Mission Sequencing.* Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, MS 301/250D, Pasadena, California 91109.

[18] Eric W. Biefeld and Lynne P. Cooper. *Replanning & Iterative Refinement in Mission Scheduling.* Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109.

[19] L. B. Bodin, B. Golden, A. Assad, and M. Ball. Routing and scheduling of vehicles and crews: The state of the art. *Computers and Operations Research*, 10:69–211, 1984.

[20] M. L. Brandeau and S. S. Chiu. An overview of representative problems in location research. *Management Science*, 35(6):645–674, June 1989.

[21] V. Cerny. A thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41–51, 1985.

[22] Seth Chaiken, Daniel J. Kleitman, Michael Saks, and James Shearer. Covering regions by rectangles. *SIAM Journal on Algebraic and Discrete Methods*, 2(4):394–410, December 1981.

[23] S. Chand. Rolling horizon procedures for the facilities in series inventory model with nested schedules. *Management Science*, 29(2):237–249, February 1983.

[24] N. Christofides. The shortest Hamiltonian chain of a graph. *SIAM Journal of Applied Mathematics*, 19:689–696, 1970.

[25] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, 1976.

[26] N. Christofides. The travelling salesman problem. In N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors, *Combinatorial Optimization*, pages 131–149. John Wiley & Sons, Inc., Chichester, 1979.

[27] N. Christofides and S. Korman. A computational survey of methods for the set covering problem. *Management Science*, 21:591–599, 1975.

[28] R. L. Church and C. S. Revelle. The maximal covering location problem. *Papers of the Regional Science Association*, 32:101–118, 1974.

[29] R. L. Church and C. S. Revelle. Theoretical and computational links between the p-median, location set-covering, and maximal covering location problems. *Geographical Analysis*, 8:406–415, 1976.

[30] G. Cornuejols and A. Sassano. On the 0,1 facets of the set covering polytope. *Mathematical Programming*, 43:45–55, 1989.

[31] G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6:791–812, 1958.

[32] M. S. Daskin. A maximum expected covering model: Formulation, properties, and heuristic solution. *Transportation Science*, 17:48–70, 1983.

[33] M. S. Daskin and E. H. Stern. A hierarchical objective set covering model for emergency medical service vehicle deployment. *Transportation Science*, 15(2):137–152, May 1981.

[34] R. H. Day. On optimal extracting from a multiple file data storage system: An application of integer programming. *Operations Research*, 13:482–494, 1965.

[35] P. B. DiDomenico. A phase-based approach to satellite constellation analysis and design. Master's thesis, M.I.T., June 1991.

[36] Jeff Dozier and Alexander F. H. Goetz. HIRIS—Eos instrument with high spectral and spatial resolution*. *Photogrammetria*, 43:167–180, 1989.

[37] B. Dueck and T. Scheuer. Threshold accepting: A general purpose algorithm apearing superior to simulated annealing. Technical Report TR 88.10.011, IBM Scientific Center Heidelberg, 1988.

[38] William C. Eggemeyer and Alan Bowling. *Deep Space Network Resource Scheduling Approach and Application.* Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109.

[39] J. Elzinga and D. W. Hearn. Geometrical solutions for some minimax location problems. *Transportation Science*, 6:379–394, 1972.

[40] J. Elzinga and D. W. Hearn. The minimum covering sphere problem. *Management Science*, 19:96–104, 1972.

[41] D. Erlenkotter. A dual-based procedure for uncapacitated facility location. *Operations Research*, 26:992–1009, 1978.

[42] J. Etcheberry. The set covering problem: A new implicit enumeration algorithm. *Operations Research*, 25:760–772, 1977.

[43] M. L. Fisher. The Lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18, January 1981.

[44] M. M. Flood. The traveling salesman problem. *Operations Research*, 4(1):61–75, 1956.

[45] R. L. Francis and J. A. White. *Facility Layout and Location: An Analytical Approach.* Prentice-Hall, Englewood Cliffs, NJ, 1974.

[46] D. S. Franzblau and D. J. Kleitman. An algorithm for constructing regions with rectangles: Independence and minimum generating sets for collections of intervals. *J. Association for Computing Machinery*, pages 167–174, 1984.

[47] B. Fritzke and P. Wilke. Flexmap—a neural network for the traveling salesman problem with linear time and space complexity. Research report, Universität Erlangen-Nürnberg, 1991.

[48] M. R. Garey, R. L. Graham, and D. S. Johnson. Some NP-complete geometric problems. In *Proceedings of the Eighth Annual ACM Symposium on Theory of Computing*, pages 10–22, Hershey, Pennsylvania, May 1976. The ACM Special Interest Group for Automata And Computability Theory.

[49] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York, 1979.

[50] R. S. Garfinkel and G. L. Nemhauser. Optimal set covering: A survey. In A. Geoffrion, editor, *Perspectives on Optimization.* Addison-Wesley, Reading, MA, 1972.

[51] S. A. W. Gerstl, C. Simmer, and B. J. Powers. The canopy hot-spot as a crop identifier. In M. C. J. Damen, G. Sicco Smit, and H. Th. Verstappen, editors, *Proceedings of the Seventh International Syumposium on Remote Sensing for Resources Development and Environmental Management*, pages 261–263. A. A. Balkema, Rotterdam, 1986.

[52] F. Glover. Artificial intelligence, heuristic frameworks and tabu search. *Managerial and Decision Economics*, 11:365–375, 1990.

[53] Alexander F. H. Goetz and Mark Herring. The high resolution imaging spectrometer (HIRIS) for Eos. *IEEE Transactions of Geoscience and Remote Sensing*, 27(2), March 1989.

[54] B. L. Golden and C. C. Skiscim. Using simulated annealing to solve routing and location problems. *Naval Research Logistics Quarterly*, 33:261–279, 1986.

[55] S. L. Hakimi. Optimal locations of switching centers and the absolute centers and medians of a graph. *Operations Research*, 12:450–459, 1964.

[56] S. L. Hakimi. Optimal distribution of switching centers in a communications network and some related graph theoretic problems. *Operations Research*, 13:462–475, 1965.

[57] Nicholas B. Hall and Michael J. Magazine. *Maximizing the Value of a Space Mission*, August 1991. Draft.

[58] K. Helbig-Hansen and J. Krarup. Improvements of the Held-Karp algorithm for the symmetric traveling-salesman problem. *Mathematical Programming*, 7:87–96, 1974.

[59] M. Held and R. M. Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18:1138–1162, 1970.

[60] M. Held, P. Wolfe, and H. P. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6:62–88, 1974.

[61] J. Hopcroft and R. Karp. An $n^{\frac{5}{2}}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

[62] Craig D. Johnson and David G. Werntz. *A New Methodology for Resource Allocation and Planning.* Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109, November 1987.

[63] D. S. Johnson. The column of NP-completeness: An ongoing guide. *Journal of Algorithms*, 3:182–195, 1982.

[64] D. S. Johnson. Local optimization and the traveling salesman problem. In *Proceedings of the 17th Colloquium on Automata, Languages and Programming*, pages 446–461. Springer Verlag, 1990.

[65] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems. II: The p-centers. *SIAM Journal of Applied Mathematics*, 37:513–538, 1979.

[66] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems. II: The p-medians. *SIAM Journal of Applied Mathematics*, 37:539–560, 1979.

[67] J. B. Kruskal, Jr. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematics Society*, 7:48–50, 1956.

[68] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors. *The Traveling Salesman Problem.* John Wiley & Sons, Inc., 1985.

[69] C. Y. Lee and E. V. Denardo. Rolling planning horizons: Error bounds for the dynamic lot size model. *Mathematics of Operations Research*, 11(3):423–432, August 1986.

[70] C. E. Lemke, H. M. Saikin, and K. Spielberg. Set covering by single branch enumeration with linear programming subproblems. *Operations Research*, 19:998–1022, 1971.

[71] S. Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44:2245–2269, 1965.

[72] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21:498–516, 1973.

[73] A. Lingas. The power of non-rectilinear holes. In M. Nielsen and E. M. Schmidt, editors, *Ninth International Colloquium on Automata, Languages and Programming*, pages 369–383. Springer-Verlag, Berlin, July 1982.

[74] R. Loulou and E. Michaelides. New greedy-like heuristics for the multidimensional 0-1 knapsack problem. *Operations Research*, 27(6):1101–1114, 1979.

[75] D. G. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley Publishing Company, Reading, Massachusetts, second edition, 1984.

[76] N. Megiddo, E. Zemel, and S. L. Hakimi. The maximum coverage location problem. *SIAM Journal for Algebraic and Discrete Methods*, 4:253–261, 1983.

[77] E. Minieka. The m-center problem. *SIAM Review*, 12:138–139, 1970.

[78] P. B. Mirchandani and R. L. Francis. *Discrete Location Theory*. John Wiley & Sons, Inc., 1990.

[79] I. D. Moon and S. S. Chaudry. An analysis of network location problems with distance constraints. *Management Science*, 30:290–307, 1984.

[80] G. C. Moore and C. S. Revelle. The hierarchical service location problem. *Management Science*, 28:775–780, 1982.

[81] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer. Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7:65–85, 1988.

[82] Nicola Muscettola, Stephen F. Smith, Gilad Amiri, and Dhiraj Pathak. Generating space telescope observation schedules. Technical Report CMU-RI-TR-89-28, The Robotics Institute, Carnegie Mellon University, November 1989.

[83] NASA, Goddard Space Flight Center. *EOS Reference Handbook*, May 1991.

[84] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1988.

[85] P. Nobili and A. Sassano. Facets and lifting procedures for the set covering polytope. *Mathematical Programming*, 45:111–137, 1989.

[86] Tatsuo Ohtsuki. Minimum dissection of rectilinear regions. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 1210–1213, Rome, 1982.

[87] T. Ohya, M. Iri, and K. Murota. Improvements of the incremental method for the Voronoi diagram with computational comparison of various algorithms. *Journal of the Operations Research Society of Japan*, 27:306–337, 1984.

[88] M. W. Padberg and G. Rinaldi. A branch and cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33:60–100, 1991.

[89] HIRIS Instrument Panel. HIRIS—High-Resolution Imaging Spectrometer: Science Opportunities for the 1990s. In *Earth Observing System Reports, vol. 2c.* National Aeronautics and Space Administration, Washington, D.C., 1987.

[90] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1982.

[91] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[92] Gerhard Reinelt. Fast heuristics for large geometric traveling salesman problems. *ORSA Journal on Computing*, 4(2):206–217, Spring 1992.

[93] C. Revelle, D. Marks, and J. C. Liebman. An analysis of private and public sector facilities location models. *Management Science*, 16:692–707, 1970.

[94] D. Rockey. Personal communication, March 1992.

[95] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis II. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal of Computing*, 6:563–581, 1977.

[96] G. T. Ross and R. M. Soland. Modeling facility location problems as generalized assignment problems. *Management Science*, 24:345–357, 1977.

[97] R. Y. Rubinstein. *Simulation and the Monte Carlo Method.* John Wiley & Sons, New York, 1981.

[98] P. Ruján. Searching for optimal configurations by simulated tunneling. *Zeitschrift Physik B Condensed Matter*, 73:391–416, 1988.

[99] H. M. Salkin and R. D. Koncal. Set covering by an all-integer algorithm: Computational experience. *Journal of The Association for Computing Machinery*, 20:189–193, 1973.

[100] M. E. Salveson. The assembly line balancing problem. *Journal of Industrial Engineering*, 6:18–25, 1955.

[101] C. Sandi. Subgradient optimization. In N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors, *Combinatorial Optimization*, pages 73–91. John Wiley & Sons, Inc., Chichester, 1979.

[102] A. Sassano. On the facial structure of the set covering polytope. *Mathematical Programming*, 44:181–202, 1989.

[103] R. Sedgewick. *Algorithms in C.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1990.

[104] M. I. Shamos and D. Hoey. Closest-point problems. In *Proceedings of the 16th IEEE Symposium on Foundations of Computer Science*, pages 151–162, October 1975.

[105] T. H. C. Smith and G. L. Thompson. A LIFO implicit enumeration search algorithm for the symmetric traveling salesman problem using Held and Karp's 1-tree relaxation. *Annals of Discrete Mathematics*, 1:479–493, 1977.

[106] K. Steiglitz and P. Weiner. Some improved algorithms for computer solution of the traveling salesman problem. In *6th Annual Allerton Conference on Circuit and Systems Theory*, pages 814–821, October 1968.

[107] B. C. Tansel, R. L. Francis, T. J. Lowe, and M. L. Chen. Duality and distance constraints for the nonlinear p-center problem and covering problem on a tree network. *Operations Research*, 30:725–743, 1982.

[108] C. Torregas, C. Revelle, and L. Bergman. The location of emergency service facilities. *Operations Research*, 19:1363–1373, 1970.

[109] G. A. Vane. Early imaging experiments with the new Airborne Visible Imaging Spectrometer (AVIRIS). In *Procedings of the 21st International Symposium for Remote Sensing of the Environment.* Environmental Research Institute of Michigan, Ann Arbor, Michigan, 1987.

[110] G. A. Vane, A. F. H. Goetz, and J. B. Wellman. Airborne imaging spectrometer: A new tool for remote sensing. *IEEE Transactions on Geoscience and Remote Sensing*, GE-22:546–549, 1984.

[111] T. Volgenant and R. Jonker. A branch and bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation. *European Journal of Operations Research*, 9:83–89, 1982.

[112] W. Walker. Application of the set covering problem to the assignment of ladder trucks to fire houses. *Operations Research*, 22:275–277, 1974.

[113] J. R. Weaver and R. L. Church. A comparison of solution procedures for covering location problems. In *Modeling and Simulation: Proceedings of the Annual Pittsburgh Conference*, volume 14, pages 1417–1422, 1983.