OPTICAL SENSOR TASKING OPTIMIZATION FOR SPACE SITUATIONAL AWARENESS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Bryan D. Little

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Carolin E. Frueh, Chair

School of Aeronautics and Astronautics

Dr. Kathleen C. Howell

School of Aeronautics and Astronautics

Dr. David A. Spencer

School of Aeronautics and Astronautics

Dr. Richard G. Cobb

Air Force Institute of Technology, Department of Aeronautics & Astronautics

Approved by:

Dr. Weinong Chen

Aeronautics and Astronautics Associate Head for Graduate Education

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Little, Bryan D. Ph.D., Purdue University, August 2019. Optical Sensor Tasking Optimization for Space Situational Awareness. Major Professor: Carolin E. Frueh.

As the number of Resident Space Objects continues to increase, the need for efficient sensor tasking strategies, to support Space Situational Awareness, continues to be of great importance. This dissertation investigates the optimization of the sensor tasking problem for ground-based and space-based optical sensors, observing objects in the Geosynchronous Earth Orbit (GEO) region. In this work, sensor tasking refers to assigning the times and pointing directions for a sensor to collect observations of cataloged objects, in order to maintain the accuracy of the orbit estimates. Sensor tasking must consider the dynamics of the objects and uncertainty in their positions, the coordinate frame in which the sensor tasking is defined, the timing requirements for observations, the sensor capabilities, the local visibility, and constraints on the information processing and communication. This research focuses on finding efficient ways to solve the sensor tasking optimization problem. First, different coordinate frames are investigated, and it is shown that the observer fixed Local Meridian Equatorial (ground-based) and Satellite Meridian Equatorial (space-based) coordinate frames provide consistent sets of pointing directions and accurate representations of orbit uncertainty for use by the optimizers in solving the sensor tasking problem. Next, two classical optimizers (greedy and Weapon-Target Assignment) which rely on convexity are compared with two Machine Learning optimizers (Ant Colony Optimization and Distributed Q-learning) which attempt to learn about the solution space in order to approximate a global optimal solution. It is shown that the learning optimizers are able to generate better solutions, while the classical optimizers are more efficient to run and require less tuning to implement. Finally,

the realistic scenario where the optimization algorithm receives no feedback before it must make the next decision is introduced. The Predicted Measurement Probability (PMP) is developed, and employed in a two sensor optimization framework. The PMP is shown to provide effective feedback to the optimization algorithm regarding the observations of each sensor.[1]

[1]The views expressed in this article are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

1. INTRODUCTION

Space Situational Awareness (SSA) is a broad classification of the information necessary to ensure safe operations in space. While there is no common, worldwide definition of all the types of information required to maintain SSA, the current state and ephemeris of resident space objects are considered a fundamental element [1, 2]. Satellite owner/operators may receive information about the dynamic states of their satellites through Telemetry, Tracking and Command (TT&C) links, but the vast majority of the tracked objects are not operational satellites with monitored TT&C links; additionally, data from these links is not typically shared by the owner/operators [3,4]. The primary source of SSA information about the dynamic states of the resident space objects comes from sensors such as radars and optical telescopes [3–7]. Whether an object is monitored by an operator or tracked by SSA sensors, the non-linear nature of orbit dynamics causes the understanding of the dynamic state to continuously degrade in the absence of measurements. Therefore, maintaining knowledge of object dynamics requires regularly re-observing the objects with independent sensors [3,7–9].

The number of resident space objects in orbit around Earth has been constantly increasing since the beginning of the space age, and will be subject to continued increase for the foreseeable future [5,6,10]. U.S. Strategic Command (USSTRATCOM) maintains a catalog of over 23,000 objects larger than 10 centimeters in diameter, including operational satellites, in orbit satellite spares, spent rocket bodies, and other debris [6, 11]. Sensor capabilities continue to improve, which is expected to lead to more objects being able to be detected [5, 6]. If a catalog were extended to include all objects greater than one centimeter, the number of objects included would be on the order of hundreds of thousands [5–7]. Additionally, the National Aeronautics and Space Administration (NASA) and the European Space Agency (ESA) estimate that if a catalog could be extended to include all objects greater than 1 millimeter in

diameter, the catalog would increase to more than 100 million objects; objects this small are not generally observed by current sensing capabilities [5, 6].

The number of sensors available for observing resident space objects is considerably smaller (on the order of hundreds) than the number of known and unknown resident space objects [7,12,13]. Further complicating this problem is the dual nature of the sensors used to observe resident space objects; these sensors are tasked with both, regular re-observation of all known objects (tracking) and detection unknown or lost objects (survey) [3,7,9,14]. Survey of unknown objects requires collecting multiple observations in order to generate candidate orbits through initial orbit determination methods [3, 9]. These candidates are then scheduled for follow-up observations to refine the candidates, with the goal of adding a new object to the catalog [7]. Tracking is the process of regularly re-observing the cataloged objects to maintain accurate orbits [3, 9]. Even under perfect dynamic modelling, the positional uncertainty of orbiting objects increases over time due to the non-linear nature of the orbit dynamics [7, 9]; combined with un-modeled perturbations, objects that are not observed on a regular basis will eventually become impossible to associate to their previously cataloged orbit [3, 9].

Both survey and tracking must be performed on a continuous basis in order to ensure the most complete understanding of the near Earth space and to maintain SSA. As a result, sensor tasking strategies that best use sensor resources are crucial. The networks and systems for observing these objects consist of various types of heterogeneous sensors, each with limitations due to location, weather, other observational constraints (e.g. night for optical observations), and sensor availability [3, 13]. Furthermore, the detectability of each object is time-dependent, based on observation geometry and object specific properties (e.g. attitude motion, shape, surface material reflectivity) [3, 7–9, 15]. This leads to a complex sensor tasking optimization problem.

1.1 Research Objectives

This work investigates the optimization of the sensor tasking problem for ground-based and space-based optical sensors, observing objects in the Geosynchronous Earth Orbit (GEO) region. Frueh et al. (2018) presented a formulation for the sensor tasking problem, which is used as the basis for the optimizations in this dissertation [7]. The analysis seeks to generate a framework for solving the formulation by meeting the following objectives:

1. Determine a coordinate frame representation that allows for fast and accurate transformation of the object probability density function from the propagation space to the measurement space in support of the optimization.

2. Compare the effectiveness and efficiency of convex and non-convex optimizers for solving the sensor tasking problem as applied in this research.

3. Generate a computationally efficient solution for optimizing the sensor tasking problem in the absence of feedback.

The sensor tasking problem is solved by generating a grid in the chosen measurement space and weighting the grid fields based on the likelihood of objects appearing in the grid. Initially, the objects are modeled as point masses in the classic two-body problem and immediate feedback is available to the optimization algorithms. The problem is extended to include uncertainty in the object states with immediate feedback, and then without feedback provided to the optimization algorithm. Common astronomical coordinate frames (where possible) and well known optimization techniques are employed to generate and test the framework for optimization of the sensor tasking problem.

1.1.1 Assumptions

The research objectives will be tested through simulations of the sensor tasking problem. The ground-based sensors are based on known telescope locations and the

space-based sensor is modeled on a real world Low Earth Orbit (LEO); the initial states of the GEO objects are drawn from the publicly available Two-Line Element Catalog. The following assumptions are applied to all simulations performed in support of this dissertation:

1. Object Specific:
    (a) All objects are modeled as one meter diameter spheres for probability of detection calculations
    (b) Orbital uncertainties are initialized as Gaussian
    (c) All objects begin with equal need to be observed
    (d) All observations are correctly correlated to the catalog objects (no mis-tagging)
2. Sensor Specific:
    (a) Only optical sensors are considered
    (b) Sensors positions are perfectly known at every observation time
    (c) Sensor pointing directions are perfectly known (no pointing errors)
    (d) Sensor operating times are known and fixed (e.g. re-positioning time)
    (e) Measurement noise is included based on known sensor models

Any additional assumptions used in the simulations will be expressed when those simulations are analyzed.

1.2 Outline of Dissertation

This work uses the formulation of Frueh, et al. (2018) for the cost function used in the optimization; four different optimizers are applied to generate the solutions to the problem. The problem is presented as a maximization of the total value of viewing directions chosen, where the values are directly related to the catalog of

objects [7]. Results are presented for three simulations in this work: a) A single ground-based sensor with immediate feedback, b) a single space-based sensor with immediate feedback, and c) two ground-based sensors with different field of view sizes and no feedback.

Chapter 2 introduces necessary background information pertinent to the sensor tasking problem. Orbits and orbital perturbations, descriptions of the coordinate frames used in sensor tasking, determination of the observer position, a description of the Two-Line Element sets describing the target objects, an introduction to optical sensors, and further descriptions of survey and tracking are provided. Previous work in the area of sensor tasking optimization is also discussed.

Chapter 3 begins with a discussion of orbit uncertainty, its causes, and its propagation. The description of the field of regard for the ground-based and space-based sensors, in terms of the measurement angles for the different coordinate frames is provided. The transformation of the orbit uncertainty into the measurement spaces is presented, along with the criteria for choosing the best measurement frame for each sensor. The analysis of the uncertainty transformation leads to the decision of which measurement spaces are best for use in optimizing the tasking of the ground-based and space-based sensors.

Chapter 4 discusses the challenges of optimizing the sensor tasking problem. The formulation of Frueh et al. (2018) is introduced, and the simplifications that are used in this work are discussed. The discretization of the field of regard for each sensor is described, followed by a description of how the combined cumulative distribution functions are calculated in the discretized field of regard. Convexity is discussed and the sensor tasking problem is analyzed, using solutions generated by the greedy algorithm, to assess if the sensor tasking problem is convex or not. Finally, the four optimizers that will be applied to the problem are introduced: a) a simple greedy optimizer that finds the local optimal solution at every time step, b) the Weapon–Target Assignment (WTA) method that assesses regional optimality to determine the assignment at each time step, c) the Ant Colony Optimization (ACO) method that

uses agents and a heuristic to iterate through the problem and arrive at an optimal solution, and the Distributed Q-Learning (DQL) algorithm that uses agents to assess the optimal solution based on rewards received.

Chapter 5 presents the results of simulations performed using the four optimizers. The simulations test the sensor tasking for a ground-based sensor and a space-based sensor, separately. Three scenarios are investigated for each sensor: a) absolute knowledge of the object positions and perfect detection, b) uncertain state estimates for the objects and perfect detection, and c) uncertain state estimates for the objects and realistic probability of detection. In each scenario, it is assumed that the optimizers receive immediate feedback of which objects are observed by the sensor at each observation.

Chapter 6 introduces the realistic constraint that the optimization algorithms do not receive immediate feedback of what the sensor observes, during the sensor tasking problem. A method for providing probabilistic feedback, based on the expected positions of the target objects is developed. The Predicted Measurement Probability method is then applied to the optimization of the tasking of two heterogeneous ground-based sensors.

Chapter 7 presents final conclusions for this work. Additional areas for study are also presented.

2. BACKGROUND

The sensor tasking problem for Space Situational Awareness requires an understanding of the orbital motion, the coordinate frames and the transformations between them, how the sensors of interest operate, as well as many other aspects. This chapter provides an overview of some of these areas and introduces much of the terminology that will be used in later chapters.

2.1 Orbital Motion

The fundamental motion of objects in orbit is described by Kepler's three laws [3, 8, 16]. Kepler's laws were stated in terms of planetary motion about the Sun, but are readily adapted to any object orbiting a primary body:

1. The orbits of objects in space are ellipses with the primary body at one focus.

2. For a constant time interval, equal areas are swept out by the radius vector (from the primary to the object).

3. The square of the object's orbit period is proportional to the cube of the semi major axis of the object's orbit.

Kepler's laws describe how orbiting bodies move, but not why they move that way. Newton's laws of motion and of gravity provide the understanding for the dynamics that govern the motion [3,8,16]. Figure 2.1 shows the interaction of two bodies in an inertial reference frame, where the mutual gravitation is the only force acting on the bodies. Each body, and the center of mass, are represented by their three dimensional inertial position vectors (e.g. $\bar{r}_1 = [x_1, y_1, z_1]^T$).

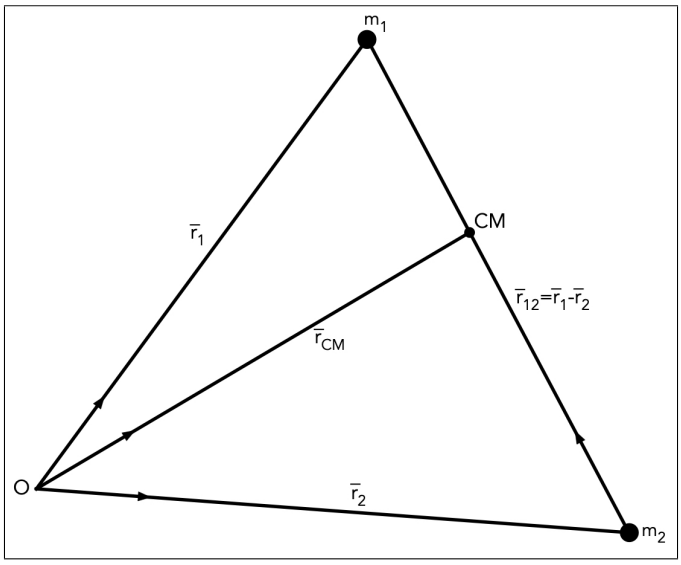


Figure 2.1. The problem of two bodies in an inertial frame.

The equations of motion for each object, and the system center of mass, follow directly from Newton's laws and are given by:

$$m_1 \ddot{\bar{r}}_1 = -G\frac{m_1\ m_2}{|\bar{r}_{12}|^3}\bar{r}_{12} \tag{2.1}$$

$$m_2 \ddot{\bar{r}}_2 = G\frac{m_1\ m_2}{|\bar{r}_{12}|^3}\bar{r}_{12} \tag{2.2}$$

$$m_1 \ddot{\bar{r}}_1 + m_2 \ddot{\bar{r}}_2 = 0 \tag{2.3}$$

where G is the universal gravitational constant [16]; m_2 is taken to be the primary (larger) body. The center of mass is moving with constant speed and direction which can be found by integrating equation 2.3 twice. Additionally, the energy and momentum of the system are known to be conserved [3, 8, 16].

For Earth orbiting satellites, the mass of the satellite is negligible as compared to the mass of the Earth. This leads to the equation of motion for the satellite:

$$\ddot{\bar{r}} = -\frac{\mu}{r^3}\bar{r} \tag{2.4}$$

where $\mu = G\ m_{\oplus}$ is Earth's gravitational constant, and r is the distance from the center of the Earth to the object in orbit. Equation 2.4 is a set of three second order, non-linear scalar differential equations. It is commonly referred to as the Two-Body Problem, where the Earth is assumed to be a perfect sphere of uniform density, and no other celestial bodies are effecting the satellite; the negative sign indicates that gravity is an attractive force, pulling the object toward the center of the Earth [3,16].

In general, celestial bodies are neither perfect spheres, nor of uniform density. The result is that a true orbit is perturbed from the two-body motion described by equation 2.4. The first step is to write the equation of motion in terms of a disturbing potential:

$$\ddot{\bar{r}} = \nabla U \tag{2.5}$$

where U is the potential function of the Earth and ∇ is the gradient operator. For the two-body problem, the potential function is simply $U = \mu/r$, which leads to equation 2.4.

To generate a potential function for the non-symmetric, non-uniform spheroid, the body is represented as a collection of mass elements, m_q. Figure 2.2 shows the geometric relationship of a point Q within the Earth and the orbiting object P [3]. Each of these mass elements provides some gravitational attraction on the orbiting body, and the full potential results from the integral:

$$U = G \int_{\text{body}} \frac{1}{\rho_q} \mathrm{d}m_q \tag{2.6}$$

Using spherical geometry and the Legendre Polynomials ($P_{\ell,m}[\cdot]$), the potential function can be written as a sum of spherical harmonics [3, 8]:

$$U = \frac{\mu}{r} + \frac{\mu}{r} \sum_{\ell=2}^{\infty} \sum_{m=0}^{\ell} \left(\frac{R_{\oplus}}{r} \right)^{\ell} P_{\ell,m} \left[\sin \phi_{gc,sat} \right] \left(C_{\ell,m} \cos(m\lambda_{sat}) + S_{\ell,m} \sin(m\lambda_{sat}) \right) \tag{2.7}$$

Equation 2.7 provides the potential function for the full gravity potential; $R_{\oplus}$ is the Earth's mean radius, while λ_{sat} and $\phi_{gc,sat}$ are the longitude and geocentric latitude

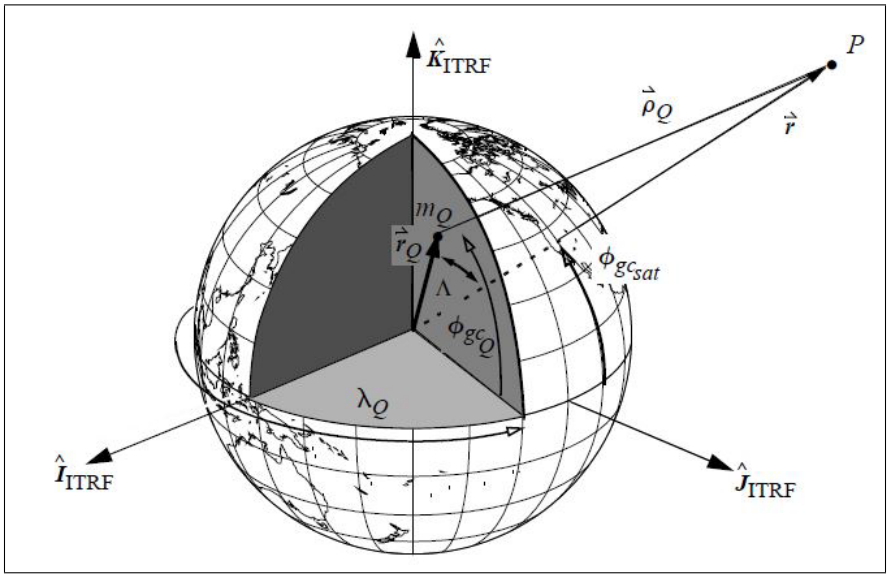


Figure 2.2. The gravitational attraction of the mass element at Q on the object P (From Ref. [3], Figure 8-3, pg. 537).

of the satellite, respectively. The zero degree/order ($\ell = m = 0$) term is simply the two-body motion about a spherically symmetric body, while all of the first degree terms ($\ell = 1$) can be shown to be zero [3].

The orbit will also be perturbed by other celestial bodies, such as the Moon and the Sun. Derivation of these terms begins by adding in the forces from the additional bodies on the primary body and the orbiting object. Again, the mass of the orbiting body can be neglected for Earth orbiting satellites, leading to:

$$\ddot{\bar{r}} = -\frac{\mu}{r^3}\bar{r} - \sum_{j=1}^{n} \mu_j \left(\frac{\bar{r} - \bar{r}_j}{|\bar{r} - \bar{r}_j|^3} + \frac{\bar{r}_j}{r_j^3} \right) \tag{2.8}$$

The positions of the perturbing bodies, $\bar{r}_j$, are given in the reference frame centered on the primary body, and μ_j represents the gravitational constant of each perturbing body.

Through the application of spherical geometry and Legendre Polynomials, we can also write the potential function for each additional body, $\mathcal{R}_j$, as:

$$\mathcal{R}_j = \frac{\mu_j}{\rho_j}\Big(1 + \sum_{k=2}^{\infty}\left(\frac{r}{\rho_j}\right)^k P_k[\cos\alpha]\Big) \tag{2.9}$$

where, ρ_j is the distance from the primary to the disturbing body and α_j is the angle between the positions of the orbiting object and the disturbing body [8]. The equation of motion can be found from equation 2.5 by substituting $\mathcal{R}_j$ for U.

The perturbations from the non-spherical primary and the third-bodies are conservative forces, causing no change in the energy or momentum of the overall system. However, there are other, non-conservative forces that are present. Two common non-conservative forces, drag and solar radiation pressure (SRP), are dependent on the size, shape, orientation, and orbital regime of the orbiting object.

Drag is a non-conservative force that is especially important for Low Earth Orbits. The resulting acceleration, given by:

$$\bar{a}_{drag} = -\frac{1}{2}\frac{c_D\ A}{m}\rho(r)v_{rel}^2\frac{\bar{v}_{rel}}{|\bar{v}_{rel}|} \tag{2.10}$$

and is dependent on satellite specific characteristics (coefficient of drag c_D, cross sectional area A, satellite mass m) and atmospheric conditions (air density at the given altitude $\rho(r)$). Drag always acts in the direction opposite the velocity of the satellite as given by the term $-\bar{v}_{rel}/|\bar{v}_{rel}|$ in equation 2.10. Drag causes secular changes in the semi-major axis and eccentricity of the orbit, which, without maneuvers, causes an orbit to decay to the point of re-entry [3].

Solar Radiation Pressure becomes important for long term propagation of objects in the Medium Earth Orbit (MEO) and GEO regimes. The form of the SRP acceleration equation is dependent on the materials used on the satellite, its shape, and its orientation (if not spherical). Complex models of satellites to produce accurate SRP effects are an area of ongoing research [17, 18]. The cannon ball model is often used

as a rough estimate of the solar radiation pressure for satellites. The acceleration for the cannon ball is:

$$\bar{a}_{SRP,sph} = -\frac{A}{m}\frac{\Phi}{c}\frac{AU^2}{|\bar{r}-\bar{r}_{\odot}|^2}\cdot\left(\frac{1}{4}+\frac{1}{9}C_d\right)\cdot\frac{\bar{r}-\bar{r}_{\odot}}{|\bar{r}-\bar{r}_{\odot}|} \tag{2.11}$$

The first term is the area to mass ratio, which tends to be small for operational satellites, but can be very large for debris objects [19]. The second term is the solar constant (Φ) divided by the speed of light (c), which provides the solar pressure per unit area [3]; because the solar constant is based on the mean Earth-Sun distance, the squared Astronomical Unit (AU^2) and distance from the satellite to the Sun ($|\bar{r}-\bar{r}_{\odot}|^2$) provide a scaling factor for the solar pressure per unit area at the satellite. The coefficient of diffuse reflection (C_d) is based on the modeled surface material; for a sphere the net specular reflection is zero. Finally, the direction is based on the unit vector from the satellite to the Sun, $\bar{r}-\bar{r}_{\odot}/|\bar{r}-\bar{r}_{\odot}|$.

Determining which perturbations to include is dependent on many factors, including the orbital regime of the satellite under analysis, the level of desired accuracy, and the duration of the propagation. For this work, only the two-body motion is considered during the simulations.

2.2 Coordinate Systems

Coordinate systems, or reference systems, are required for describing the motion of objects in space, and for defining the measurements of those objects when they are observed. In this work, all coordinate systems considered are orthogonal systems, meaning that the coordinate axes are orthogonal. Additionally, coordinate systems may be either right-handed or left-handed. Figure 2.3 shows a right-handed system; the third axis (z) is positive above the plane formed by rotating counter clockwise from the primary to secondary axis (x to y). Figure 2.3 also depicts how the same point may be described by different coordinates, such as the Cartesian coordinates (x, y, z) or spherical coordinates $(\rho, \varphi, \vartheta)$; these coordinates are related to each other

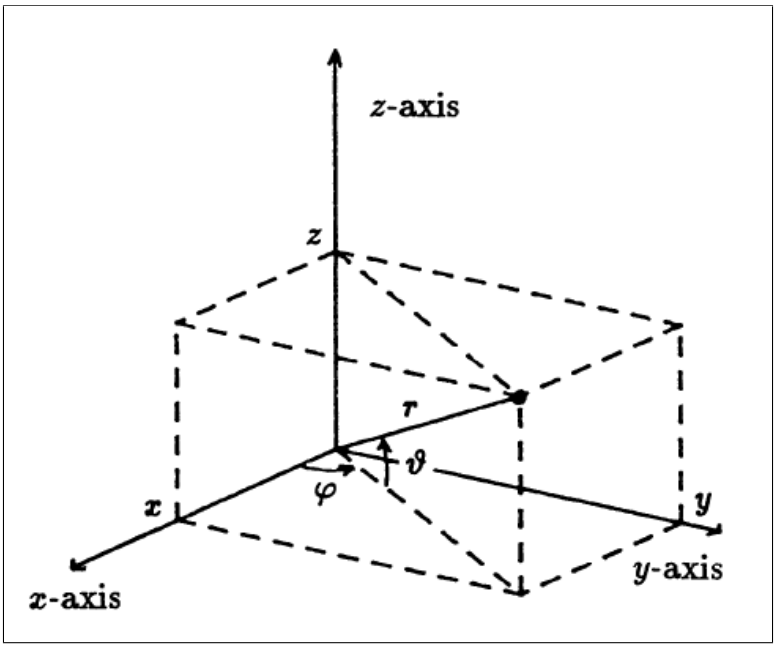


Figure 2.3. A right-handed coordinate system showing the relation between Cartesian and spherical coordinates (From Ref. [20], Figure 2.1, pg. 10) [20]

through transformation equations. This section will introduce additional coordinate systems, coordinate transformations, and the measurement angles which are used to describe the orbiting objects as seen by an observer.

Propagation of Earth orbiting objects is generally performed in an *inertial* reference frame based on a given epoch. In this work, the Earth Centered Inertial (ECI) reference frame, terrestrial equivalent of the International Celestial Reference Frame (ICRF/J2000.0), is used for orbit propagation [3]. The ECI reference plane is the Earth's equatorial plane, the primary direction ($\hat{\imath}$-axis) is toward the mean vernal equinox, the normal to the reference plane ($\hat{k}$-axis) is through the Earth's rotation axis, and the $\hat{\jmath}$-axis completes the right-handed system [3].

The states of the objects and the space-based sensor are generated in the ECI frame, while the position of the ground-based sensor is defined the Earth Centered, Earth Fixed (ECEF) coordinate frame and rotated to the ECI frame. The ECEF

coordinate frame is a rotating frame, where the primary direction ($\hat{e}_1$-axis) is defined to be through the Greenwich meridian, the second axis ($\hat{e}_2$-axis) is through 90° East, and the normal ($\hat{e}_3$-axis) is through the Earth's rotation axis [3].

Rotation of a position vector from the ECEF coordinate frame to the ECI coordinate frame is accomplished by a rotation about the Earth's rotation axis. The angle of rotation is the Greenwich Mean Sidereal Time (GMST, Θ), which defines the rotation of the prime meridian past the $\hat{\imath}$-axis. The position rotation is given as:

$$\bar{R}_{ECI} = \mathbf{R}_3(\Theta)\bar{R}_{ECEF} \tag{2.12}$$

where $\bar{R}$ is a position vector, and $\mathbf{R}_3(\cdot)$ is the matrix representation for a rotation about the third axis.

While the propagation of the objects is done in the ECI frame, some of the perturbations discussed above are more readily calculated in Earth fixed frames such as ECEF. For example, the effects of the non-sphericity of the Earth are typically calculated in an Earth fixed frame, because they depend on the objects position with respect to the mass distribution of the Earth [3,21]. The effects can then transformed back to the ECI frame and used in the propagation.

2.2.1 Ground-Based Observers

There are four common coordinate systems used in celestial mechanics to describe the location of an object observed by a ground-based optical (or electro-optical) telescope; each can be related back to the ECI position of the object. The first is the Geocentric Equatorial System (GES), which is based on the Earth Centered Inertial frame. The angles in the GES are the geocentric right ascension (α_g) and declination (δ_g), which are related to the ECI position vector of the object ($\bar{r} = [r_x\ r_y\ r_z]^T$). The

position of an orbiting object can be written in terms of its distance from the center of the Earth ($r = |\bar{r}|$) and its geocentric angles, as:

$$\bar{r} = \begin{bmatrix} r_x\hat{\imath} \\ r_y\hat{\jmath} \\ r_z\hat{k} \end{bmatrix} = \begin{bmatrix} r\ \cos\alpha_g \cos\delta_g \\ r\ \sin\alpha_g \cos\delta_g \\ r\ \sin\delta_g \end{bmatrix} \tag{2.13}$$

Likewise, if the position vector is known in ECI coordinates, the angles may be derived through trigonometric relationships, performing all necessary quadrant checks[1].

For objects that are very far from Earth, the difference between the center of the Earth and the observer position becomes negligible and the GES is adequate for observing the object. For near Earth objects, the difference between the object's position with respect to the center of Earth and with respect to an observer on Earth's surface, is significant. For these near Earth objects, the Equatorial Vernal Equinox (EVE) system is introduced. EVE is closely related to GES, with the angles based on the range vector of the object with respect to the observer position:

$$\bar{\rho} = \bar{r} - \bar{R} \tag{2.14}$$

where, $\bar{r}$ is the object position as given in equation 2.13, and $\bar{R}$ is the observer's position.

For the EVE system, the coordinate frame is centered at the observer's Earth fixed location (topocenter) with the reference plane parallel to the Earth's equator and the primary direction always toward the mean vernal equinox. The angles in EVE are the topocentric right ascension (α), measured positively from the primary direction and within the reference plane ($0^\circ < \alpha \leq 360^\circ$), and declination ($\delta$), which

[1]Quadrant checks will be an important step in the determination of angles based on each of the following coordinate systems.

is measured above and below the reference plane ($-90° \leq \delta \leq 90°$). The range vector is found from the scalar range and the measurement angles as:

$$\bar{\rho} = \begin{bmatrix} \rho \ \cos\alpha\cos\delta \\ \rho \ \sin\alpha\cos\delta \\ \rho \ \sin\delta \end{bmatrix} \tag{2.15}$$

Optical sensors cannot measure the range to an object. Dividing equation 2.15 by ρ results in the pointing vector $\bar{u} = \bar{\rho}/|\bar{\rho}|$, as defined in the ECI frame. The angles are then derived from the pointing vector as:

$$\alpha = \tan^{-1}\left(\frac{u_y}{u_x}\right) \tag{2.16}$$

$$\delta = \sin^{-1}(u_z) \tag{2.17}$$

Since the primary direction in the EVE system is fixed (to the vernal equinox) in space while the topocenter is rotating with the Earth, the angle pairs within the observer's field of regard (visible sky) are constantly changing throughout the night.

A coordinate system where the axes are fixed with respect the observer, results in fixed pointing angles within the field of regard. The Local Meridian, Local Horizon (LMLH) system is such a fixed coordinate system. The reference plane for LMLH is the local horizontal plane, and the primary direction is South along the local meridian of the topocenter[2]. The angles in LMLH are the azimuth (β), measured positively from South to West ($0° < \beta \leq 360°$), and the elevation (h), which is measured above the local horizon ($0° \leq h \leq 90°$). LMLH is a left-handed system, which results in distant objects (e.g., stars) always appearing to move positively in the azimuth direction.

[2]Other definitions place the primary direction to the North, but this work will use South.

The pointing vector (based on the ECI position vectors of the object and observer) is related to the angles β and h as:

$$\begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} = \mathbf{S}_2\mathbf{R}_3(\theta)\mathbf{R}_2(\frac{\pi}{2} - \phi) \begin{bmatrix} \cos\beta\cos h \\ \sin\beta\cos h \\ \sin h \end{bmatrix} \tag{2.18}$$

where, the matrix $\mathbf{S}_2$ makes the system left-handed by switching the direction of the y-axis, θ is the Local Mean Sidereal Time (LMST), and ϕ is the geographic latitude of the observer [20]. In equation 2.18, the LMST rotates the local meridian to the mean vernal equinox direction, and is the combination of the geographic longitude and the GMST ($\theta = \text{GMST} + \lambda$). Rearranging equation 2.18, the azimuth and elevation are given in terms of the pointing vector components as:

$$\beta = \tan^{-1}\left(\frac{u_x\sin\theta - u_y\cos\theta}{u_x\cos\theta\sin\phi + u_y\sin\theta\sin\phi - u_z\cos\phi}\right) \tag{2.19}$$

$$h = \sin^{-1}\left(u_x\cos\theta\cos\phi + u_y\sin\theta\cos\phi + u_z\sin\phi\right) \tag{2.20}$$

The last common coordinate system for ground-based observers is the Local Meridian Equatorial (LME) system. LME uses the same reference plane as EVE, parallel to the equator and through the topocenter, but the primary direction is away from the Earth's rotation axis in the direction of the local meridian. The angles used in LME are Hour Angle (τ) and Declination (δ) [20]. Similar to LMLH, LME is a left-handed system, measuring τ positively from the local meridian toward the West ($0^\circ < \tau \leq 360^\circ$); declination is measured in the same way in EVE and LME. The relation of the LME angles to the pointing vector in ECI is:

$$\begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} = \mathbf{S}_2\mathbf{R}_3(\theta) \begin{bmatrix} \cos\tau\cos\delta \\ \sin\tau\cos\delta \\ \sin\delta \end{bmatrix} \tag{2.21}$$

where, again, the LMST is used in equation 2.21 to rotate the topocenter to align with the vernal equinox. Likewise, the angles are given in terms of the ECI pointing vector components as:

$$\tau = \tan^{-1}\left(\frac{u_x \sin\theta - u_y \cos\theta}{u_x \cos\theta + u_y \sin\theta}\right) \tag{2.22}$$

$$\delta = \sin^{-1}(u_z) \tag{2.23}$$

Note, equation 2.23 is the same as equation 2.17 for the EVE system.

For ground-based sensors, the EVE, LME, and LMLH systems may all be used by optical sensors for measuring Earth orbiting objects. LMLH provides the most intuitive angles for pointing sensors as the elevation is always above the local horizon and azimuth can be related to compass directions. Because the orientation of LMLH and LME are fixed to the topocenter, the field of regard is static with respect to the observer, simplifying the determination of pointing directions at every time. The EVE has the most direct relation to the object inertial position coordinates, providing the most direct relation between the frame of propagation and the frame in which measurements are taken. The benefits and drawbacks to these systems will be further investigated in Chapter 3.

2.2.2 Space-Based Observers

Space-based sensors provide many benefits for observing Earth orbiting objects. The sensors orbit outside the atmosphere, relieving issues related to the atmospheric aberration and terrestrial weather with which ground-based sensors must contend. They are also not restricted to observing only at night, though they must account for not pointing the sensor too close to the Sun or the illuminated side of the Moon to avoid sensor saturation.

One difficulty of space-based sensors is the definition of a coordinate system in which to take measurements. This author was not able to find any literature that discussed such coordinate systems. Instead, two coordinate systems based on frames

used in other scenarios have been defined for use by space-based sensors. The first is called the Satellite Orbit Radial (SOR) system and is based on the RSW coordinate frame (see Ref. [3], pg 165), and the second is analogous to the LME discussed above, and is called the Satellite Meridian Equatorial (SME) system.

The SOR uses the sensor's orbital plane as the reference plane for the system and the sensor's radius vector as the principal direction. Rotating vectors into this coordinate system requires the 3-1-3 Euler angle rotation about the right ascension of the ascending node (Ω), the inclination (i), and the true longitude ($\lambda = \omega + f$) of the space-based sensor's orbit. Figure 2.4 shows the relation of the ECI and the RSW coordinate frames through the angles Ω, i, ω (argument of perigee), and f (true anomaly). The measurement angles for the SOR are given in terms of the ECI

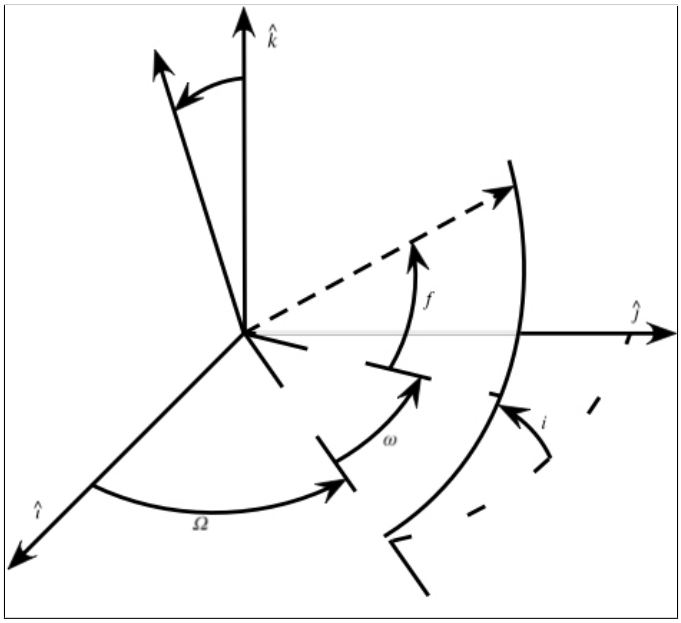


Figure 2.4. Geometry of coordinate transformation from ECI to RSW.

pointing direction coordinates as:

$$\vartheta_s = \tan^{-1}\left(\frac{\mathrm{S}_1\ u_x + \mathrm{S}_2\ u_y + \mathrm{S}_3\ u_z}{\mathrm{R}_1\ u_x + \mathrm{R}_2\ u_y + \mathrm{R}_3\ u_z}\right) \tag{2.24}$$

$$\varphi_s = \sin^{-1}\left(\mathrm{W}_1\ u_x + \mathrm{W}_2\ u_y + \mathrm{W}_3\ u_z\right) \tag{2.25}$$

Where:

$$\mathrm{R}_1 = \cos\Omega\cos\lambda - \sin\Omega\sin\lambda\cos i$$

$$\mathrm{R}_2 = \sin\Omega\cos\lambda + \cos\Omega\sin\lambda\cos i$$

$$\mathrm{R}_3 = \sin\lambda\sin i$$

$$\mathrm{S}_1 = -\cos\Omega\sin\lambda - \sin\Omega\cos\lambda\cos i$$

$$\mathrm{S}_2 = -\sin\Omega\sin\lambda + \cos\Omega\cos\lambda\cos i$$

$$\mathrm{S}_3 = \cos\lambda\sin i$$

$$\mathrm{W}_1 = \sin\Omega\sin i$$

$$\mathrm{W}_2 = -\cos\Omega\sin i$$

$$\mathrm{W}_3 = \cos i$$

The longitudinal angle, ϑ_s, is measured positively in a right-handed sense from the radius vector, and the latitudinal angle, φ_s, is measured about the orbital plane ($-90^\circ \leq \varphi_s \leq 90^\circ$). The terms R_i, S_i, W_i, are the elements of the rotation matrix formed by the 3-1-3 Euler angle rotation from ECI to RSW:

$$\begin{bmatrix} \mathrm{R}_1 & \mathrm{R}_2 & \mathrm{R}_3 \\ \mathrm{S}_1 & \mathrm{S}_2 & \mathrm{S}_3 \\ \mathrm{W}_1 & \mathrm{W}_2 & \mathrm{W}_3 \end{bmatrix} = \mathbf{R}_3(\lambda)\mathbf{R}_1(i)\mathbf{R}_3(\Omega) \tag{2.26}$$

The SME has the reference plane parallel to the equator and passing through the sensor's orbital position. The principal direction is given by projecting the sensor's radius vector onto the reference plane, extending out from the orbit position; this is

analogous to the local meridian for the ground-based sensor, but is defined based on the angle θ_{sbs} which is related to the sensor's ECI x_{sbs} and y_{sbs} components by:

$$\theta_{sbs} = \tan^{-1}\left(\frac{y_{sbs}}{x_{sbs}}\right) \tag{2.27}$$

where the position coordinates of the observer are assumed to be accurately known. The measurement angles are given as:

$$\tau_s = \tan^{-1}\left(\frac{u_x \sin\theta_{sbs} - u_y \cos\theta_{sbs}}{u_x \cos\theta_{sbs} + u_y \sin\theta_{sbs}}\right) \tag{2.28}$$

$$\delta_s = \sin^{-1}(u_z) \tag{2.29}$$

where τ_s is measured positively in the left-handed sense and δ_s is measured about the reference plane.

For the ground-based sensor coordinate systems, the angles are limited by the physical nature of the topocenter, primarily the minimum elevation. Theoretically, the space-based sensor FOR could include all directions that do not point through the Earth; pointing at the Sun or Moon are time varying and dealt with separately. However, this work assumes that the satellite maintains a fixed orientation with respect to the measurement space (SOR or SME coordinate system), and the camera is fixed to the satellite face that is directed away from Earth. The FOR for the space-based sensors are limited to $\pm 75^\circ$ for the in-plane angles (ϑ_s and τ_s) and $\pm 90^\circ$ for the out of plane angles (φ_s and δ_s).

2.3 Observer Position

Expressing the observer's position is another critical aspect of SSA modelling. For the space-based sensor, the observer position is expressed in the same way as the target objects. However, the ground-based sensor's position is generally fixed to a point on the Earth. If the Earth were spherical, determining the ECEF position would be a simple matter of using the radius (R), the longitude (λ) and the latitude (ϕ) of the observer. However, as previously discussed, the Earth is non-spherical.

The geocentric latitude (ϕ_{gc}) and the radius from the Earth's center to the observer position must be adjusted based on the flattening of the Earth at the equator [3]. The ECEF (non-rotating) position of the observer is given by:

$$\bar{R}_{ECEF} = \begin{bmatrix} R\ \cos\lambda\cos\phi_{gc} \\ R\ \sin\lambda\cos\phi_{gc} \\ R\ \sin\phi_{gc} \end{bmatrix} \tag{2.30}$$

where R and ϕ_{gc} are the corrected radius and geocentric latitude, respectively. The true location of the observer is affected by other forces as well.

The orientation of Earth in the ICRF is not fixed, but experiences three perturbing effects: Precession is the secular motion of the Earth's celestial North pole around the celestial North pole of the ecliptic, Nutation is the periodic motion of the Earth's North pole towards and away from the celestial North pole of the ecliptic, and Polar Motion is the variation of Earth's rotation axis about Earth's celestial North pole. Precession and Nutation are due to torques from other celestial bodies, primarily the Sun and Moon [3]. Polar Motion is understood to be the motion of the rotation axis with respect to the Earth's crust, i.e., the rotation axis is not a fixed direction [3]. Figure 2.5 shows the motion of Earth's axes with respect to the celestial sphere.

The Precession comes from treating the Earth as a rotationally symmetric gyroscope and calculating the torque on the axis of rotation. The torque is due to the inclination of the Sun and Moon with respect to the Earth's equator. This luni-solar torque on Earth's rotation axis is expressed as [23]:

$$\bar{\mathbf{D}} = \frac{3}{2}(I - I_{eq})\sin(\epsilon)\cos(\epsilon)\Big(n_{\odot}^2 + n_{\mathfrak{D}}^2\Big)\hat{\imath} \tag{2.31}$$

The I and I_{eq} are the moments of inertia for the axially symmetric, oblate spheroid approximation for Earth; ϵ is the obliquity of the ecliptic; $n_{\odot}$ and $n_{\mathfrak{D}}$ are the mean motions of the Sun and Moon, respectively.

The resulting Precession defines the change from Mean Equinox of Date (MOD) for the epoch, T, to MOD at J2000.0. Precession is a secular effect that is determined

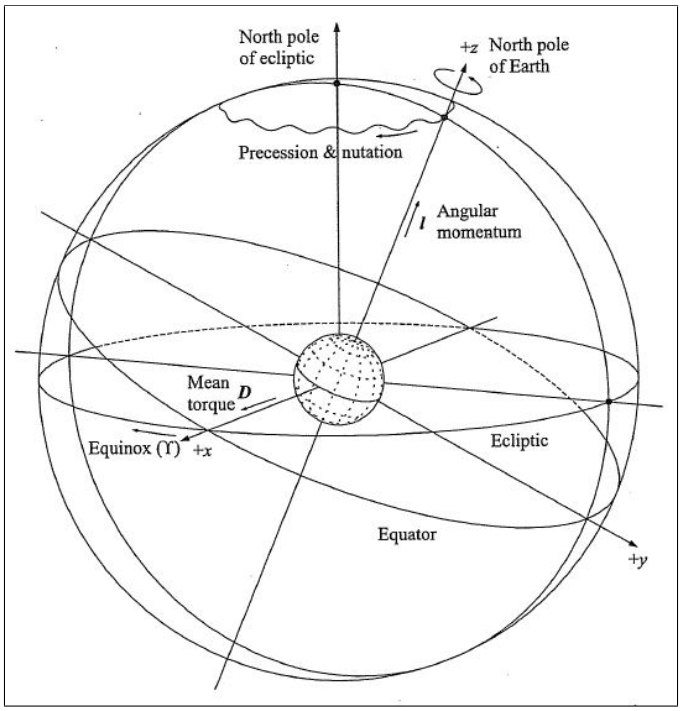


Figure 2.5. The motion of Earth's axes as a result of the luni-solar torques (From Ref. [22], Fig. 5.7, pg. 173).

by calculation of the 3-2-3 Euler angle rotation (ζ, θ, and z). The angles are given as:

$$\zeta = 2306.2181'' \cdot T + 0.30188'' \cdot T^2 + 0.017998'' \cdot T^3 \tag{2.32}$$

$$\theta = 2004.3109'' \cdot T - 0.42665'' \cdot T^2 - 0.041833'' \cdot T^3 \tag{2.33}$$

$$z = 2306.2181'' \cdot T + 1.09468'' \cdot T^2 + 0.018203'' \cdot T^3 \tag{2.34}$$

where T is epoch in Julian centuries past J2000.0 ($T = (JD - 2451545.0)/36525.0$), as evaluated in terrestrial time [3]. The matrix representation is given by $\mathbf{P}$ as:

$$\mathbf{P} = \mathbf{R}_3(\zeta)\mathbf{R}_2(-\theta)\mathbf{R}_3(z) \tag{2.35}$$

In addition to Precession, Earth's rotational axis undergoes small periodic motions called Nutation, which arises from variations of the luni-solar torque that are monthly

and annual, and are mainly due to changes in the orientation of the Moon's orbit with respect to the Earth's equator [3]. Addressing these variations allows for transforming from MOD to the True Equinox of Date (TOD) [3].

The Nutation can be expressed as a 1-3-1 Euler angle rotation which depends on $\bar{\epsilon}$, the mean obliquity of the ecliptic at T; $\Delta\Phi$, the difference in longitude between the mean and true vernal equinox; and $\Delta\epsilon$, the difference between the mean and true obliquity of the ecliptic. $\bar{\epsilon}$ in terms of the epoch T is given by [3]:

$$\bar{\epsilon} = 84381.448'' - 46.8150'' \cdot T - 0.00059 \cdot T^2 + 0.001813 \cdot T^3 \tag{2.36}$$

The values for $\Delta\Phi$ and $\Delta\epsilon$ are dependent on the mean anomalies of the Sun ($M_{\odot}$) and Moon ($M_{\leftmoon}$), the mean longitude of the Moon ($u_{M_{\leftmoon}}$), the mean distance to the Sun (D), and the mean longitude of the ascending node of the Moon ($\Omega_{\leftmoon}$) [3, 24]; each of these are related to the epoch T (see Vallado, pg 230 [3]). The values for $\Delta\Phi$ and $\Delta\epsilon$ are then calculated from trigonometric series. The true obliquity of the ecliptic is given by $\epsilon = \bar{\epsilon} + \Delta\epsilon$, and the Nutation is expressed in matrix form as [3]:

$$\mathbf{N} = \mathbf{R}_1(-\bar{\epsilon})\mathbf{R}_3(\Delta\Phi)\mathbf{R}_1(\epsilon) \tag{2.37}$$

The final perturbation of Earth's orientation is the Polar Motion which can be understood by treating the Earth as an axially symmetric gyroscope, where the rotation axis is moving around a fixed axis in the absence of torques. Unfortunately, the Polar Motion cannot be described by analytic formulas, but is understood through observations; the values derived from these observations may be found in look-up tables [3]. The parameters x_p and y_p, from the look-up tables, are used to define the matrix form of the Polar Motion as:

$$\mathbf{\Pi} = \mathbf{R}_2(-x_p)\mathbf{R}_1(-y_p) \tag{2.38}$$

In order to generate very accurate understanding of the observer object geometries over long time spans, these effects and the perturbations on the orbiting objects should

be modelled. In this work, the time spans are assumed to be short enough to ignore these effects.

2.4 Two-Line Element Catalog

The objects used in the simulations for this work all come from the publicly available Two-Line Element set (TLE) catalog, which is published by United States Strategic Command (USSTRATCOM). The TLE catalog is maintained by the 18th Space Control Squadron, using the Space Surveillance Network assets, and all of the data is generated with SGP4 [3, 11, 25]. While some changes to the TLE format have occurred since its inception, the TLE is still based on the punch cards that were used at the start of the space age [25]. Figure 2.6 shows an example TLE.

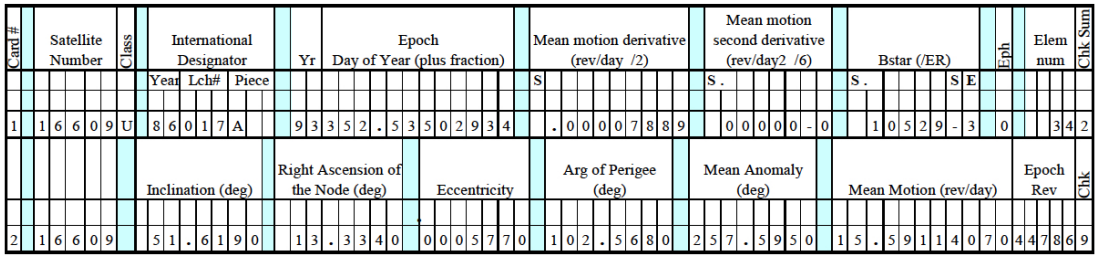


Figure 2.6. An example of the TLE format used in the publicly available catalog (From Ref. [25], Figure 11, pg. 30).

In the first line of Figure 2.6, the Satellite Number is a five digit identifier given to each object in the catalog, while the International Designator is a combination of the two digit year of launch, the launch number for the given year, and a description of which "piece" of the launch the TLE belongs to (e.g. Payload) [25]. The Epoch provides the timing of the latest orbit update in terms of the two digit year, the day of year (i.e. 1-365), and the time in fractions of a day (epoch is in Universal Time Coordinated) [3, 25]. According to Vallado et al. (2006), the derivatives of the mean motion and the Ephemeris type (Eph) are not used in SGP4, and the element set

number is incremented with each update [25]. The Bstar is a "drag-like" coefficient used in SGP4 [25].

The second line contains the orbital elements that describe the size, shape, and orientation of the orbit. The inclination (i), right ascension of the ascending node (Ω), argument of perigee (ω), and mean anomaly (M) are given in degrees, to four decimal places. The eccentricity (e) is given to seven decimal places; the orbits are assumed to be closed about Earth, so $e < 1$ and the decimal on the left is assumed. The last value that is used in determining the orbit of the object is the mean motion (n), from which the semi-major axis may be calculated ($a = \sqrt[3]{\mu/n^2}$).

While errors are inherent in any orbit estimation, the TLE does not provide any measure of the uncertainty in the orbit estimate. Vallado et al. (2006) state that the number of decimals used in the TLE format limits their accuracy, though they assert that the accuracy is generally "about a kilometer or so at epoch and it quickly degrades" (pg. 30) [25]. Dong and Chang-yin (2010) showed that the accuracy of orbits propagated with SGP4 degrade differently for different orbital regimes; the errors between SGP4 propagations and "high accuracy" propagations, for objects in the GEO region, approach 40 kilometers after 15 days of propagation [26]. Additionally, Flohrer et al. (2009) showed that the combined along-track and out-of-plane position errors found when comparing optical observation to TLE predictions, could be as much as 70 kilometers [27]. The limited information about the accuracy of the orbits in the TLE catalog is considered when generating the object uncertainties for the simulations in Chapters 5 and 6.

2.5 Optical Sensors

The coordinate systems allow for the definition of the angles to measure the objects observed by the sensor, but they do not describe the sensor itself. Sensors are not only limited by where they can be pointed, but by the amount of sky they can see (field of view), the brightness of the objects that can be detected, the time it takes

to re-position the sensors, and many other aspects. This section will introduce some characteristics of optical sensors that influence their ability to observe objects in orbit.

For a ground-based observer, the field of regard (FOR) is the amount of sky that could theoretically be observed by a sensor at the observer location, taking into account obstructions (e.g., local mountains); space-based observers tend to have larger FORs, where the primary limitation is pointing directions that intersect the Earth. Optical sensors are further limited by their field of view (FOV), which may be defined as the amount of sky visible within the aperture of the sensor. FOV is generally given as an angular value that describes the diameter (circular) or the edge lengths (rectangular) of the FOV; in this work, the FOVs used are rectangular or square.

The shape of the FOV is dependent on the shape of the detector, such as a Charged Couple Device (CCD). The CCD collects signal from all celestial light sources, as well as sources of noise [28, 29]. Figure 2.7 provides an example of signals being tracked (dots), while other sources are moving with respect to the sensor (streaks). Masking, averaging, and other techniques may be used to reduce some of the unwanted signals, but there is no way to remove all noise sources.

The signal strength of an object, at the detector, is given as:

$$E(S_{sig,obj}) = (D-d)\frac{\bar{\lambda}}{\hbar c} \cdot \exp\big(-\tau(\bar{\lambda}) \cdot R(\zeta)\big) \cdot I_{sig}(\bar{\lambda}) \tag{2.39}$$

Equation 2.39 is an approximation using the mean wavelength, $\bar{\lambda}$, of the energy being reflected. It is dependent on the amount of energy received at the observer (I_{sig}), the area of the primary (D) and secondary (d) mirrors, the atmospheric function (R), the atmospheric extinction coefficient (τ), Planck's constant ($\hbar$), and the speed of light (c). The I_{sig}, assuming spherical objects, is given as:

$$I_{sig}(\bar{\lambda}) = I_{0,Sun}\frac{A}{\rho^2} \cdot \Big(\frac{2}{3}\frac{C_d(\bar{\lambda})}{\pi}(\sin\alpha + (\pi-\alpha)\cos\alpha)\Big) \tag{2.40}$$

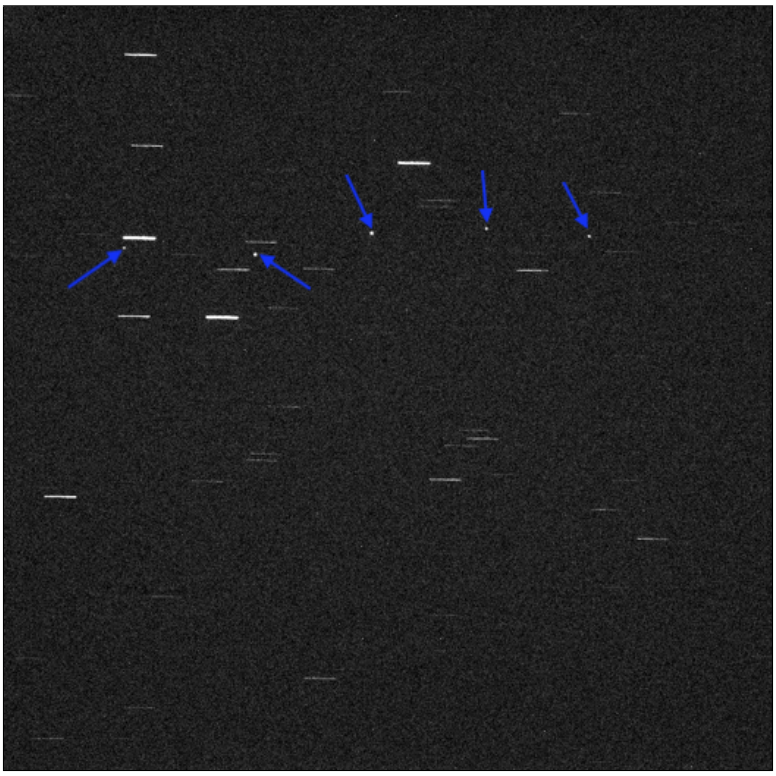


Figure 2.7. Example of a CCD image showing five objects. This image was taken with the Purdue Optical Ground Station, observing five satellites in GEO; rate tracking causes the stars to appear as streaks, while the objects appear as points.

where α is the Sun-object-observer phase angle, $C_d(\bar{\lambda})$ is the coefficient of diffuse reflection (average wavelength), A is the cross-sectional area of the object, and ρ is the object-observer range.

The signal strength must rise above the noise level at the detector in order to be detected, but detection is only guaranteed if the Signal-to-Noise ratio (SNR) is sufficiently high [28]. Sources of noise include, but are not limited to, celestial background sources (e.g. stars), spurious electron emitted by the CCD (dark noise), errors in the

read out process, and sensor gain. The variance of the object signal also contributes to the SNR, which is given as:

$$\text{SNR} = \frac{\sum_i^{n_{pix}} \lambda_{obj,i}}{\sqrt{\lambda_{obj,i} + n_{pix}\left(1 + \frac{1}{n_b}\right)\left(\lambda_{S,i} + \lambda_{D,i} + \sigma_{R,i}^2 + \frac{g^2}{24}\right)}} \tag{2.41}$$

This is the modified Merline equation for a CCD, where the number of electrons per pixel of the signal ($\lambda_{obj,i}$), sky background ($\lambda_{S,i}$), and dark noise ($\lambda_{D,i}$) are Poisson random variables, the readout noise ($\sigma_{R,i}^2$) is a Gaussian random variable and the gain ($g^2/24$) is a uniform random variable [28].

Sanson and Frueh present a probability of detection formula for observations of orbiting objects from ground-based sensors, which takes into account a user defined SNR threshold [28]:

$$\begin{aligned} p_d =& 1 - \frac{1}{2}\sum_{-\infty}^{\infty} \frac{\Gamma(n - \frac{g}{2}, \lambda_{\text{obj,i}} + \lambda_{S,i} + \lambda_{D,i})}{n!} \\ & \cdot \left(\text{erf}\left(\frac{n+1-t-\mu_{B,i}}{\sqrt{2g(\sigma_{B,i}^2+\sigma_{R,i}^2)}}\right) - \text{erf}\left(\frac{n-t-\mu_{B,i}}{\sqrt{2g(\sigma_{B,i}^2+\sigma_{R,i}^2)}}\right)\right) \end{aligned} \tag{2.42}$$

where, t is the SNR threshould above which detection is assured. In this work, when p_d is not assumed to be one, it will be calculated using equation 2.42.

2.6 Sensor Tasking Techniques

The sensor tasking problem is generally described in terms of the two methods for employment, a) survey and b) tracking [3, 7, 14, 30]. A key difference in the two methods is the knowledge, however imperfect, of the objects intended to be seen. Survey is the method of scanning for unknown or un-tracked objects, collecting multiple observations to generate initial orbits, and refining the orbit until the object can be tracked with some level of confidence. Surveys are often conducted based on the population of the known objects, with the intent to find unknown objects (like debris) that exist in nearby orbits [31–35]. Tracking is the process of obtaining

new observations for objects with known (cataloged) orbits in order to maintain the accuracy of the orbit estimate [30, 33, 35–37].

2.6.1 Survey

Survey methods are an important aspect of SSA. They are essential for generating orbit estimates for debris objects generated by degradation of other objects or after collision events. Once an orbit estimate is generated through survey operations, or by an owner/operator during launch and orbit insertion, the orbit estimate must be maintained by regular tracking of the object.

Survey strategies may be very different for objects in different orbital regimes. For objects in Low Earth Orbit (LEO), radar systems have long been used to observe objects and generate candidate orbits [3, 13, 38]. Optical sensors are better suited for surveillance of objects in higher altitudes where relative velocities are smaller, and objects spend longer periods within a sensor's field of view [3,12,39]. For such objects, especially near GEO, sidereal tracking is used to generate observations which can be combined to produce initial orbit determinations. Figure 2.8 shows a simulated observation using sidereal tracking; the stars appear as points, while any objects appear as streaks. The high relative velocities of LEO objects with respect to ground-based observers makes surveillance of these objects with optical sensors difficult; however, optical sensors may generate un-correlated tracks, during tracking missions, to combine with surveillance observations for performing initial orbit determination [3, 9].

Early survey strategies using optical sensors to observe debris objects in GEO were introduced by Schildnecht, Hugentobler, & Verdun (1995), with continued development occurring ever since [7, 12, 31, 32, 39, 40]. A common sensor tasking method for surveying GEO with optical sensors is known as the stripe scanning method, which uses stripes of pointing directions that share the same right ascension and are spaced in declination to cover at least $\pm 15^{\circ}$ [7, 12, 31, 40]. When properly constructed with respect to the field of view (FOV) of the sensor and the relative angular velocities of

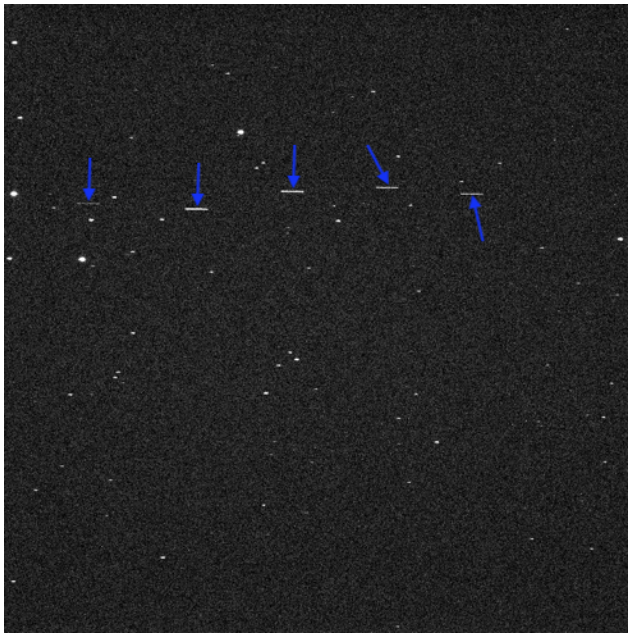


Figure 2.8. During sidereal tracking (tracking the stars) the relative velocities of Earth orbiting objects causes them to appear as streaks. This image is taken with the Purdue Optical Ground Station, observing the same GEO satellites as Figure 2.7.

the target objects, a single stripe can generate a leak-proof strategy for understanding the debris population [12, 31, 32, 32]. Such strategies are able to model the debris populations in GEO, in order to understand the risks to operational satellites and begin to inform solutions for the debris problem [31, 34, 41].

A single stripe method will only produces one observation per object per night, but multiple observations are required for generating object orbits [12, 31]. This led to the development of a two stripe method, which is generally not leak-proof, that can be used to generate at least two observations per night [12, 40]; Figure 2.10 shows an example of the two stripe method. Traditional initial orbit determination methods require at least three observations, however, the admissible regions method of Milani, et al. (2004) allows for generating sets of candidate orbits from limited information [42].

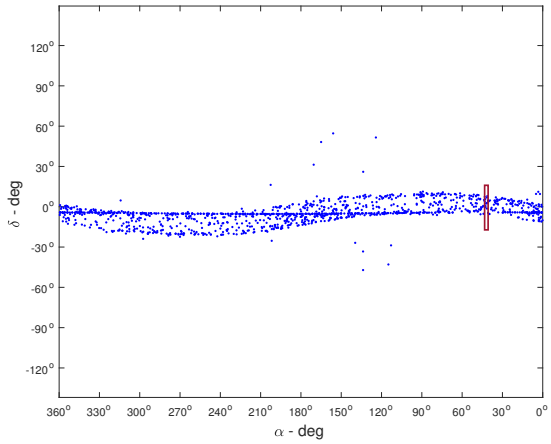


Figure 2.9. Properly constructed single stripe scanning provides leak-proof surveillance of the target population.

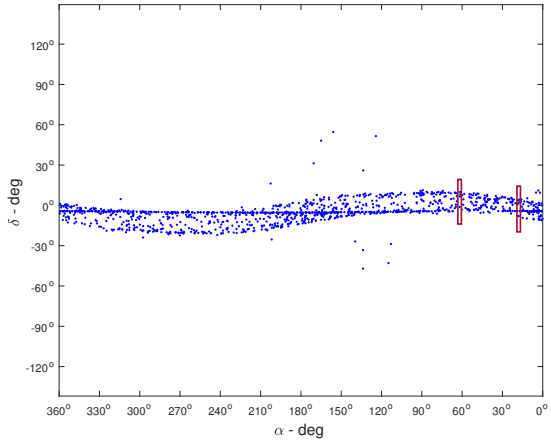


Figure 2.10. Two stripe scanning provides multiple observations of the visible GEO population in order to generate initial orbit candidates.

Paul, Frueh, & Fiedler (2019) have proposed a hypothesis surface method for tasking sensors to perform survey operations [19]. The surface is generated by determining the average combined cumulative distribution functions for a hypothesized

population, within a set of pre-defined viewing directions. The surface values for the viewing directions, indicate which viewing direction is most likely to obtain measurements of target objects, such as debris. By cycling through these viewing directions, multiple observations of desired objects should be possible, leading to the creation of candidate orbits.

2.6.2 Tracking

Tracking is the employment of SSA sensors in order to maintain viable orbit estimates for resident space objects already entered in a catalog. Objects in LEO may be tracked with radar or optical sensors, while objects at GEO are typically tracked with optical sensors [7, 31]; this work focuses on optical tracking of GEO objects. Because tracking is focused on known objects, it will often include decisions about when to make an observation based on some additional measure such as observability [15,30,35,37]. Effective tracking methods ensure the custody of objects in current catalogs is maintained. Loss of custody causes an object to have to be rediscovered through survey operations. Thus, the tracking and survey missions of SSA are interdependent.

Traditionally, optical tracking is performed on individual objects, with the sensor pointing determined by maximizing the probability that the object is within the FOV at the time of the observation. Ideally, this places the target object at the center of the FOV as in Figure 2.11, where the target object is indicated by the green dot and the FOV is the red square; it may be possible for additional objects, in similar orbits, to be observed along with the target object, as indicated by the blue dots. If the covariance of the target object is small enough to fit within a sensor's field of view, and the illumination conditions are good, then the object should be observed [35,43].

The tracking problem is critical to maintaining custody of cataloged objects in the near Earth space; if custody is lost, the object must be re-acquired through

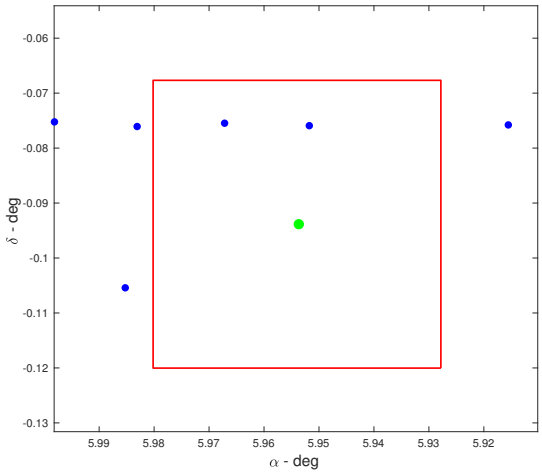


Figure 2.11. Classical tracking attempts to place the desired object (green) at the center of the FOV to maximize the chances of its observation.

survey operations. Additionally, the survey and tracking missions often require use of the same resources (sensors), which puts stress on the sensor utilization rates. As the population of the RSOs continues to grow, the stress on the resources will also continue to increase. This dissertation focuses on optimizing the sensor tasking problem for optical tracking of GEO objects.

2.7 Sensor Tasking Optimization

Optimizing the sensor tasking problem is focused on assigning sensors in the most efficient manner possible. To this end, various methods have been explored, ranging from heuristic assumptions to rigorous mathematical modeling [14,30,31,33,35,36,41, 43–48]. Linares and Furfaro (2017) have shown the benefits of using Reinforcement Learning by employing an Asynchronous Advantage Actor-Critic (A3C) method for solving the sensor tasking problem [45]. The actor generates a policy for the sensor tasking that attempts to maximize the reward received; the reward used is an uncertainty threshold for the RSOs, but other examples are noted [45]. To support

the final solution, the critic learns an estimate of the action-value function for the problem [45]. The critic provides the updated estimate of the value function, which the actor uses to generate the updated policy. This balance is able to generate an optimal solution more quickly than the optimizing based on the value function directly, while preventing the policy from converging to a sub-optimal local maxima [45]. Additionally, the Linares and Furfaro (2017) employ neural networks for both the actor and the critic, helping to decrease the computation time for the problem [45]. The authors were able to show that for populations of 100 and 300 RSOs, the method reduces the trace of the position covariances, to below a predefined level [30].

Hill et al. (2010) presented a method for scheduling a network of sensors by considering the orbit covariance of each RSO given a centrally generated task list [43]. In this case, the sensor tasking was based on the orbit error covariance of the target object, and the observation geometry between the object and the sensors in the network [43]. Hill et al. (2010) introduce an observation effectiveness metric, which is based on the expected reduction of the position error variance from an observation of a given RSO by a given sensor [43]. They were able to show that individual observations could be scheduled based on this metric, and the overall median position uncertainty of the catalog could be effectively reduced. This method is theoretically able to handle large catalogs of objects and multiple sensors, but it assigns tasking based on individual objects being observed by each sensor at each observation time.

Frueh et al. (2018) introduce a sensor tasking formulation that is not based on a single targeted object being observed at each observation time [7]. As shown in Figure 2.7, it is not uncommon for multiple objects to be observed within a sensor's FOV at a given pointing direction. Therefore, the sensor tasking problem should be able to be solved by considering all objects which are visible to the sensor. Additionally, the method allows for the uncertainty in the target orbits to be considered, incorporates the probability of detection for the objects, and provides for including other factors such as an uncertainty threshold in order to optimize the sensor tasking problem. Frueh et al. (2010) also allows for optimizing both the survey and tracking problems

in a combined cost function [7]. This formulation will be further discussed in Chapter 4.

Patel et al. (2108) present a method to search for a known target object based on the expected information gain present in candidate orbits [49]. A single target object is represented by a set of candidate orbits through a Monte Carlo sampling of the target object's probability density function. The sensor then uses the Kullback-Leibler divergence and Minimum Mean Conditional Entropy measures to determine the best measurement for the sensor to attempt; these Information-theory measures provide an assessment of the reduction in the probability distribution based on the hypothetical observations, which is then used to assign the next sensor tasking step [49]. The authors show that the information theoretic methods are able to eliminate unlikely candidate orbits faster than a maximum probability method that assigns sensor tasking based only on the candidates that are most likely to represent the true object.

Each of these methods provide different methods for determining where a sensor should be pointed to capture the target objects. Often the focus of sensor tasking is on observations of single objects, though that is not a strict constraint for optical sensors. This work seeks to further investigate the combination of object position and sensor pointing angles as part of the optimization of the sensor tasking problem.

2.8 Summary

This work focuses on optimizing the employment of optical sensors for the re-observation (tracking) of objects in the Geosynchronous region. To perform the optimization, the most advantageous coordinate system will be selected and multiple optimization techniques will be applied. Both ground-based and space-based sensors will be modeled to show how the tracking problem changes in each case: ground-based sensors are the most common sensors applied to observation of objects at GEO, while use of space-based sensors to observe objects in orbit is an area of ongoing research

with fewer data points [47, 50, 51]. The objects to be observed will be taken from the publicly available Two-Line Element (TLE) catalog, from which an expected position will be determined. Due to many factors, the true state of an orbiting object is only probabilistically known, with the uncertainty being represented by a probability density function (PDF). As the orbits are propagated, the volume and shape of the PDFs are subject to change; this is especially true for the position uncertainty, where the non-linear nature of the orbit problem causes in-track error growth [3, 9, 27, 52, 53]. Taking all of this into account, the sensor tasking will seek to re-observe as many of the objects in the catalog as possible during an observation interval.

This chapter introduced orbits, orbit perturbations, coordinate frames for both ground-based and space-based observers, the variability of ground-based observer positions, the Two-Line Element sets that define the objects to be observed, the optical sensors that are commonly used for observing Earth orbiting objects, and sensor tasking problem definition and some previous solutions. The effects discussed in sections 2.1 and 2.3 are not considered further in this work. The coordinate systems introduced in section 2.2 are discussed further in the following chapter, and the p_d from equation 2.42 will be employed in Chapters 5 and 6.

3. ORBIT UNCERTAINTY FOR SENSOR TASKING

The exact states of orbiting objects are not perfectly known, as discussed in section 2.1. Additionally, any un-modelled orbital perturbations that effect the true states, lead to uncertainty which grows over time. This chapter covers some of the sources of the uncertainty, the effect of propagation on the uncertainty, and how the uncertainty can be transformed into the measurement spaces of the observer. The information from this chapter will be used to support the optimization of the sensor tasking for ground-based and space-based sensors, observing objects in the GEO region.

3.1 Sources of Orbit Uncertainty

For most objects in orbit, the only way to estimate the state is through imperfect observations. Generally, radar is used for LEO objects, while optical sensors provide better information for objects at MEO and GEO. Unfortunately, all sources of observations on orbiting objects contain noise and other errors that increase the uncertainty in the estimated states of the objects [3, 9]. However, the observations are not the only source of uncertainty in the orbits. The non-linearity of the orbit problem itself causes the uncertainties to change over time, while any un-modelled perturbations will cause the true state to diverge from the estimated state.

3.1.1 Uncertainty from Sensors

Since this work uses optical sensors to observe objects at GEO, the sources of uncertainty for those sensors will be discussed. Potential errors in the measurements from optical sensors can arise from several sources, including uncertainty in the pointing angles, CCD errors, and atmospheric refraction. When a measurement is taken,

the observer reports the angles which define the pointing direction of the sensor, such as:

$$\bar{u} = \begin{bmatrix} u_{\hat{x}} \\ u_{\hat{y}} \\ u_{\hat{z}} \end{bmatrix} = \begin{bmatrix} \cos\alpha\cos\delta \\ \sin\alpha\cos\delta \\ \sin\delta \end{bmatrix} \tag{3.1}$$

in the EVE coordinate system. These reported angles are themselves measured, and thus subject to errors, $d\alpha$ and $d\delta$, from the true angles. Calibrations are used to ensure these error values are as small as possible, but it is impossible to eliminate them completely [9].

Optical sensors have a FOV that allows them to observe an area of the sky. The effective FOV is focused onto the CCD, where the energy is measured. The CCD is an array of pixels, and measurements of a single object are often spread across more than one pixel as shown in Figure 3.1; the spreading of the signal makes the determination of the exact position more difficult. A low SNR will also add uncertainty into the determination of the position on the CCD. Even if the object is contained on only one pixel, it is generally not possible to discriminate where exactly on the pixel the energy is most concentrated. This combines with the noise sources discussed in section 2.5, to generate uncertainty about the measurement.

Finally, as energy passes through the atmosphere it is affected in various ways. This includes the refraction of the light as it transitions from space to the atmosphere [20]. Figure 3.2 shows the effect of atmospheric refraction on the light from a star, as observed from Earth. This bending results in errors in the actual measurements [20].

The uncertainty in the measurements are incorporated into the orbit estimates when the observations are used to generate new estimates. As technology has improved, observers have been able to reduce the errors in the measurements through better calibration techniques and more refined sensors. Generally, the resulting errors

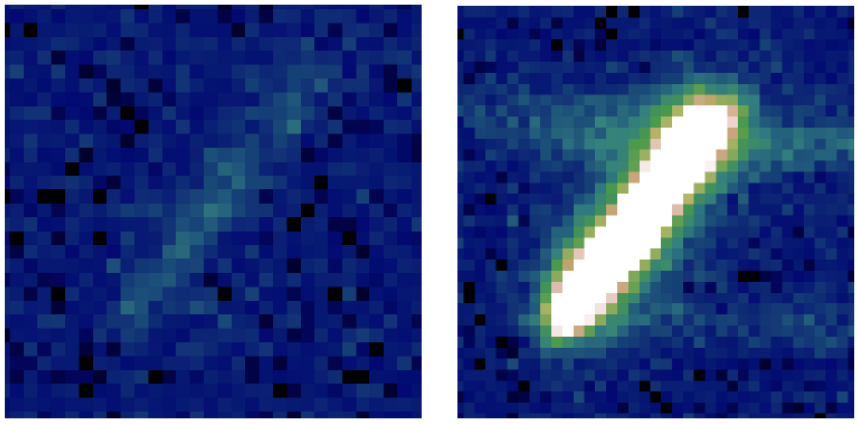


Figure 3.1. Examples of object images spread over multiple images. The image on the left has a low SNR, while the image on the right has a high SNR (From Ref. [29], Fig. 3.11, pg. 31).

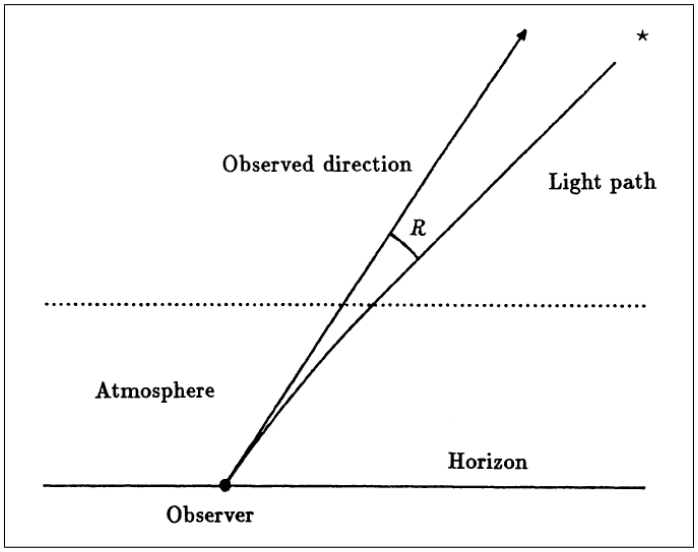


Figure 3.2. Atmospheric refraction bends the path of light toward the vertical (From Ref. [20], Fig. 3.3, pg. 46).

are expected to be smaller than the uncertainty in the orbit estimates before an orbit update, so the updated orbit has a smaller PDF volume.

3.1.2 Uncertainty in Propagation and Orbit Updates

This work uses an Extended Kalman Filter (EKF) framework to propagate the states and uncertainties of the objects through their orbits, using two-body dynamics; all objects have initial orbit estimates and no Initial Orbit Determination is performed. The system and measurement models for the EKF are given as [54]:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) + \mathbf{M}(t)\mathbf{w}(t) \tag{3.2}$$

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{L}_k\mathbf{v}_k \tag{3.3}$$

where, in the system model (equation 3.2) $\mathbf{f}(\mathbf{x}(t))$ represents the dynamics, $\mathbf{x}(t)$ is the state, $\mathbf{w}(t)$ is the noise in the system, and $\mathbf{M}(t)$ maps the noise to the dynamics [54]. In the measurement model (equation 3.3) $h(\mathbf{x}_k)$ is non-linear mapping of the state to the measurement space, $\mathbf{x}_k$ is the state at time t_k, $\mathbf{v}_k$ is the measurement noise, and $\mathbf{L}_k$ maps the noise to the measurement [54].

The true state, $\mathbf{x}(t)$, is generally unknown, while the mean state, $\mathbf{m}(t)$, is defined as the expected value of the true state, and is given by [3, 9, 54]:

$$\mathbf{m}(t) = \mathrm{E}\Big\{\mathbf{x}(t)\Big\} \tag{3.4}$$

where the noise term, $\mathbf{w}(t)$, from the system model is zero mean, white noise, so it drops out of equation 3.4 when the expected value is taken. The covariance is the expected value of the squared error of the mean state from the true state as given by [3, 9, 54]:

$$\mathbf{P}(t) = \mathrm{E}\Big\{\big(\mathbf{x}(t) - \mathbf{m}(t)\big) \cdot \big(\mathbf{x}(t) - \mathbf{m}(t)\big)^T\Big\} \tag{3.5}$$

Equations 3.4 and 3.5 provide the definition of the mean and covariance (first and second moments of the PDF) used in the EKF; the EKF cannot generate an initial mean and covariance, but requires that they be provided, e.g. as the output of a

batch least squares [9, 54]. The propagation of the mean and covariance are achieved by:

$$\dot{\mathbf{m}}(t) = \mathbf{f}(\mathbf{m}(t)) \tag{3.6}$$

$$\dot{\mathbf{P}}(t) = \mathbf{F}(\mathbf{m}(t))\mathbf{P}(t) + \mathbf{P}(t)\mathbf{F}^T(\mathbf{m}(t)) + \mathbf{M}(t)\mathbf{Q}_s(t)\mathbf{M}^T(t) \tag{3.7}$$

where the noise in the system is included by using the shape matrix, $\mathbf{M}(t)$, to incorporate the process noise spectral density, $\mathbf{Q}_s(t)$, into the covariance propagation. This has the effect of inflating the covariance over time, and helps to keep the uncertainty large enough to contain the true state when measurements are not available. However, if measurements are not provided to regularly update the PDF, the uncertainty may become large enough that the object would be considered "lost".

When the measurement of an object is available, the Kalman update equations are used to produce an updated mean and covariance [54]:

$$\hat{z}_k = \mathbf{h}(\mathbf{m}_k^-) \tag{3.8}$$

$$\mathbf{W}_k = \mathbf{H}(\mathbf{m}_k^-)\mathbf{P}_k^-\mathbf{H}^T(\mathbf{m}_k^-) + \mathbf{L}_k\mathbf{R}_k\mathbf{H}_k^T \tag{3.9}$$

$$\mathbf{C}_k = \mathbf{P}_k^-\mathbf{H}^T(\mathbf{m}_k^-) \tag{3.10}$$

$$\mathbf{K}_k = \mathbf{C}_k\mathbf{W}_k^{-1} \tag{3.11}$$

$$\mathbf{m}_k^+ = \mathbf{m}_k^- + \mathbf{K}_k(z_k - \hat{z}_k) \tag{3.12}$$

$$\mathbf{P}_k^+ = \mathbf{P}_k^- - \mathbf{C}_k\mathbf{K}_k^T - \mathbf{K}_k\mathbf{C}_k^T + \mathbf{K}_k\mathbf{W}_k\mathbf{K}_k^T \tag{3.13}$$

where $\hat{z}_k$ is the expected measurement based on the mean state, z_k is the actual measurement recorded by the sensor, $\mathbf{H}(\mathbf{m}_k^-)$ is the Measurement Jacobian evaluated at the *a priori* mean, $\mathbf{W}_k$ is called the Innovations Covariance, $\mathbf{R}_k$ is the measurement noise covariance, $\mathbf{C}_k$ is the Cross Covariance, and $\mathbf{K}_k$ is the Kalman Gain; the superscripts on $\mathbf{m}_k$ and $\mathbf{P}_k$ denote the mean and covariance before the update (-) and after the update (+). The Kalman update equations result in a new estimate of the position and covariance for an object, and these updated moments of the PDF are used in equations 3.6 and 3.7 moving forward.

Because of the non-linear nature of the orbit problem, even when the dynamics are assumed to be perfectly known, the position covariance changes during propagation. Figure 3.3 shows the result of this phenomenon. An object position is defined in GEO and it is fitted with an initial covariance; the 3-σ interval is plotted in magenta. The object is then propagated for 25 hours, just over one full orbit, and the 3-σ interval of the covariance is plotted again. Despite the assumption of perfect dynamics and with zero process noise, the covariance has stretched in the in-track direction.

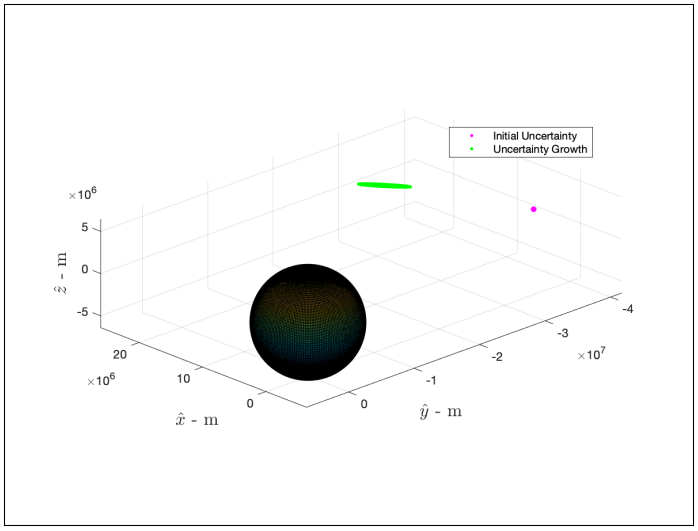


Figure 3.3. An object ($a = 42142\ km, e = 4.2 \times 10^{-4}, i = 0.7°, \Omega = 48°, \omega = 140°$) is plotted with an initial covariance ($\sigma_{pos} = 50\ km, \sigma_{vel} = 1\ m/s$) and then propagated over more than one orbit to show the position uncertainty growth.

In practice, the dynamics are not perfectly known which will cause additional growth in the covariance; the primary changes still occur in the in-track direction based errors in the semi-major axis of the orbit, though errors in other elements will also add to the uncertainty growth [27, 53]. While the measurements will operate to decrease the overall uncertainty, they still contain errors which will be present in the updated uncertainty for the object PDF after the Kalman update.

Additionally, the perturbation forces discussed in Chapter 2 that are not modeled in the system dynamics, will drive additional uncertainty in the orbit estimate as the PDF is propagated. For example, the non-sphericity of Earth causes secular changes in the RAAN and argument of perigee for an orbit. Not modelling these effects causes the expected state and the true state to diverge, which must be accounted for by increasing the uncertainty around the expected state. Other perturbations change the orbits in different ways, but the effect is still an increase in the uncertainty about the expected state of the object. The EKF accounts for un-modeled dynamics by increasing the covariance volume with the process noise spectral density, $\mathbf{Q}_s(t)$, in equation 3.7.

3.2 Uncertainty and Measurement Spaces

Before discussing the transformation of the uncertainty, the measurement spaces to which the uncertainty is transformed are discussed in detail. As previously stated, the only information provided by the optical sensors used in this work, is the angle pair that defines the geometry between the observer and the object as described by the vector relationship in equation 2.14, and the pointing vector, $\bar{u}$, given in equation 3.1; the range vector is related to the pointing vector by $\bar{\rho} = \rho \cdot \bar{u}$. The angle pair reported for an observation is dependent on the coordinate system chosen for the given sensor, as discussed in sections 2.2.1 and 2.2.2. The allowable angle pairs to which a sensor can be pointed are a function of the field of regard (FOR) for the sensor. The ground-based measurement spaces will be introduced first, followed by the space-based measurement spaces.

3.2.1 Ground-Based Sensors

The LMLH coordinate system provides for a simple angular definition of the FOR. In Figure 3.4 the FOR is shown as a rectangle in azimuth and elevation, (β, h). The bottom edge of the FOR is defined to be at the minimum elevation, h_{min}, with the

remaining edges obeying the spans defined in section 2.2.1. In this coordinate system, the FOR is fixed to the observer.

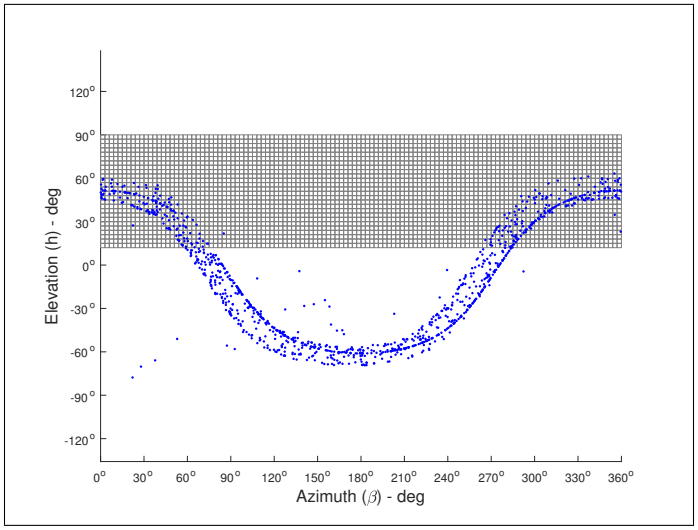


Figure 3.4. The FOR as represented in the LMLH measurement space. The GEO objects are plotted to show how they relate to the FOR in this measurement space.

In Figure 3.4, everything below the FOR represents pointing directions to which the sensor is not allowed to be commanded. The top edge represents the pointing directions about the observer's zenith direction ($h = 90°$). There is a singularity at zenith where the azimuth direction can no longer be determined. The physical result is that the pointing directions near the top of the FOR have significant overlap. For the LMLH coordinate system, this can be problematic for observers at lower latitudes (e.g. $|\phi| < 30°$), where the zenith direction is likely to intersect orbits of the target objects. Singularities are common spherical coordinate frames, but the choice of reference plane can limit the impact on the problem.

The angles in the EVE coordinate system (α, δ) are derived from the pointing vector using equations 2.16-2.17. Because the coordinate frame is fixed in space, the angles that define the FOR are not static with respect to the observer, and must be calculated at every time. Figure 3.5 shows the FOR in the EVE system, at one

moment in time. The bottom edge of the FOR is again restricted by h_{min}, and the white space below the FOR represents the pointing directions that are blocked by the Earth. The curve representing the bottom of the FOR is caused by the fact that the

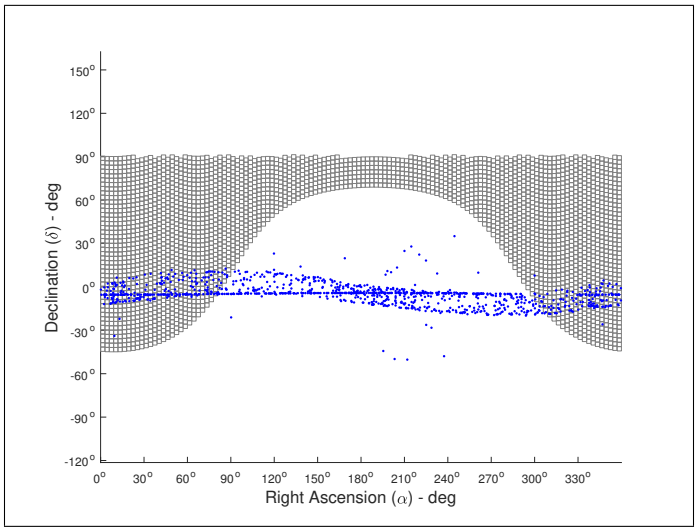


Figure 3.5. One instance of the FOR as represented in the EVE measurement space. The target objects are plotted, and the geostationary objects form a horizontal line at $\delta \approx 6°$.

minimum declination, δ_{min}, is a function of h_{min}, the geographic latitude (ϕ), and an azimuth angle (β):

$$\delta_{min} = \sin^{-1}\left(\sin\phi \sin h_{min} - \cos\phi \cos h_{min} \cos\beta\right) \tag{3.14}$$

Unfortunately, β is also a function of the desired declination (δ_{min}), and equation 3.14 is not easily solved.

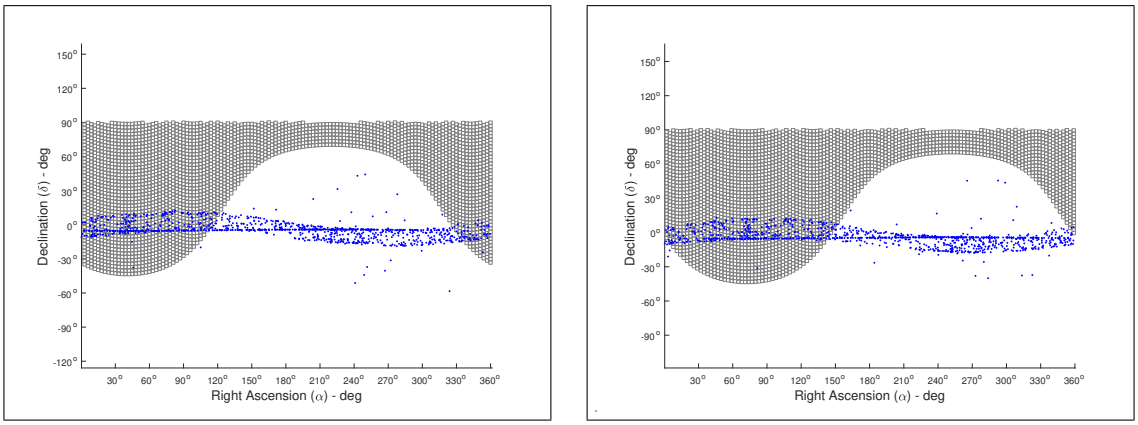
Instead of solving for δ_{min} directly, a simple iterative solution can be found by solving for the elevation associated with δ_{min}:

$$h = \sin^{-1}\left(\sin\phi \sin\delta_{min} + \cos\phi \cos\delta_{min} \cos(\theta - \alpha)\right) \tag{3.15}$$

where, θ is the LMST. Equation 3.15 is solved iteratively, adjusting δ_{min} until $|h - h_{min}| < 0.001$ radians. Equation 3.15 is solved for increments in α to generate the bottom of the FOR shown in Figure 3.5.

The top of the FOR represents the pointing directions that are near the normal to the reference plane. As declination approaches $\delta = 90°$, right ascension becomes undefined (singularity), and those pointing directions overlap significantly in three-dimensional space. The GEO objects used in this work generally have inclinations less than 30°, which avoids the issues that will arise from the singularity in this measurement space.

As discussed in section 2.2.1, and mentioned above, fixing the principal direction in space causes this frame to rotate with respect to the observer as the Earth rotates; therefore, the angles that define the FOR are subject to constant change. This is shown in Figures 3.6(a) and 3.6(b); equation 3.15 must be solved across the span of α, at every time, as the FOR shifts.



(a) The FOR in EVE 2 hours, 6 mins after Figure 3.5.

(b) The FOR in EVE 4 hours, 14 mins after Figure 3.5.

Figure 3.6. Shift of EVE FOR over time as the observer position rotates with Earth.

The FOR definition in the LME coordinate system shares some attributes with each of the previously discussed systems. Figure 3.7 shows the FOR which, like the

FOR in LMLH, is static with respect to the observer. The angles in this system are the hour angle (τ) and declination (δ) given in equations 2.22-2.23. The δ_{min} in LME is also calculated from equation 3.15, where τ replaces $\theta - \alpha$; however, since the FOR is static with respect to the observer, the calculations of δ_{min} are only required once.

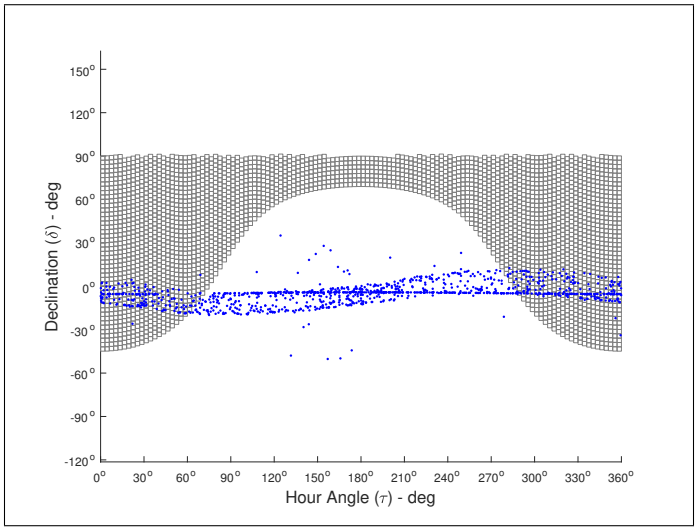


Figure 3.7. The static FOR as represented in the LME measurement space. As in Figure 3.5, the geostationary objects form a horizontal line.

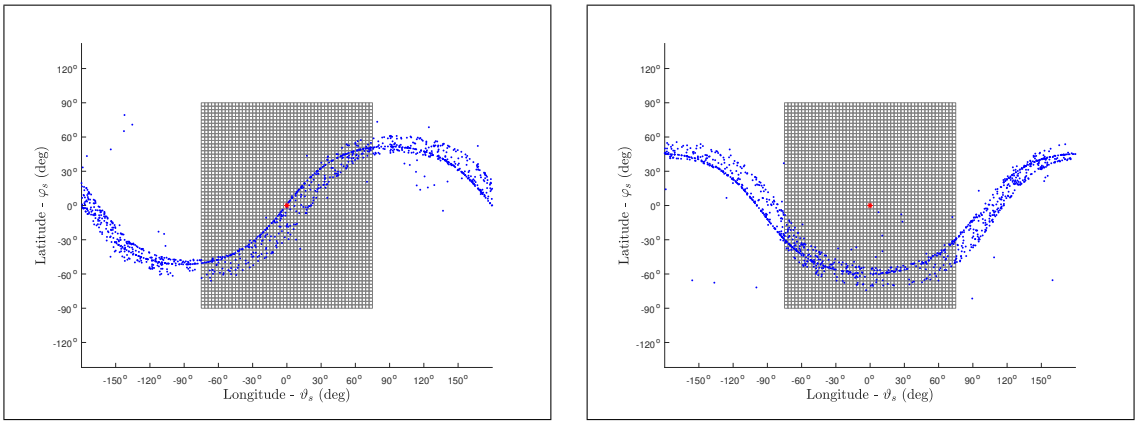
As with the other systems, the white space below the FOR in Figure 3.7 represents the pointing directions blocked by the Earth. The LME and EVE system use the same reference plane, so the singularity caused by $\delta = 90°$ has the same effect in both systems.

3.2.2 Space-Based Sensors

There are significantly fewer space-based sensors in use and, consequently, not much documentation of the coordinate systems used to define the measurements for such sensors. As discussed in section 2.2.2, this work compares the Satellite Orbit Radial (SOR) system, based on the RSW coordinate frame described in Ref. [3] (page

165), and the Satellite Meridian Equatorial (SME) system, which is analogous to the LME coordinate system for ground-based sensors. The SOR and SME coordinate systems are both fixed to the observer location, and the resulting FOR is static with respect to the observer in each space.

The FOR in the SOR coordinate system is oriented based on the satellites orbital radius and orbit plane, with the angles ϑ_s (longitude) and φ_s (latitude) given by equations 2.24-2.25. The sensor used in this work is in an inclined orbit ($\approx 52°$), causing the changing distribution of the GEO objects, with respect to the observer, seen in Figure 3.8. The orbital motion of the sensor causes a shift in the apparent position and orientation of the objects in the measurement space; this is shown by comparing Figure 3.8(a), where the sensor is near its line of nodes, and Figure 3.8(b), where the sensor is near its highest orbital position (maximum z_{sbs}). The shift is



(a) The FOR and expected measurements of the target objects in the SOR system when the sensor is near its line of nodes.

(b) The FOR and expected measurements of the target objects in the SOR system when the sensor is near its highest orbital position.

Figure 3.8. As the sensor orbits, the GEO objects shift in position and orientation, with respect to the FOR.

best observed by looking at point $\vartheta_s = 0°$, $\varphi_s = 0°$ (the red $*$). In Figure 3.8(a) the distribution of the objects passes through this point because the sensor is near the Earth's equator which puts the (0,0) point of the FOR in line with the GEO belt;

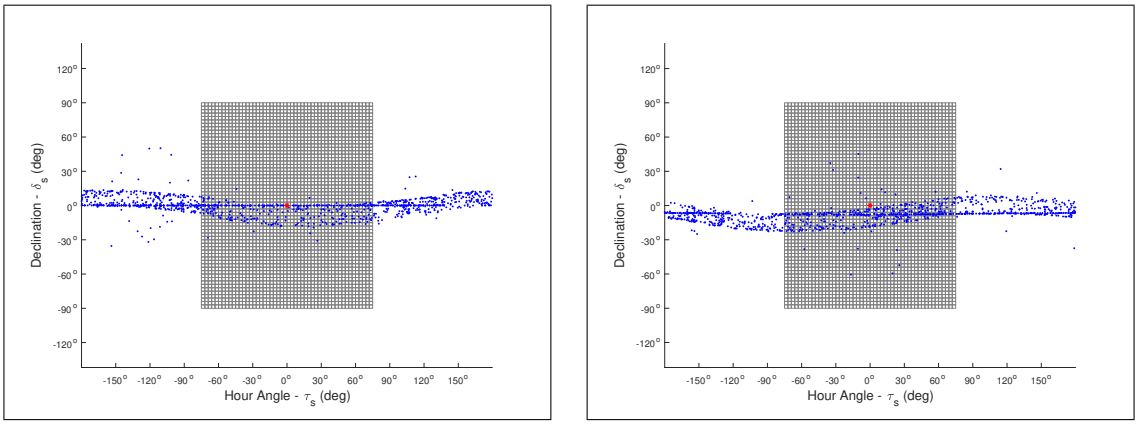
however, the inclined orbit and the orientation of the sensor causes the objects to appear to be distributed along an incline through the same point, because the sensor has a fixed orientation with respect to the orbit plane. In Figure 3.8(b), the sensor is near the highest point in its orbit, which causes the GEO belt to appear well below the (0,0) point of the FOR. In this case, the objects crossing $\vartheta_s = 0°$ do not appear inclined with respect to the FOR because of the orientation of the sensor at this point in its orbit.

The reference plane of the SME coordinate system is parallel to the Earth's equatorial plane at all times, and the measurement angles are given by equations 2.28-2.29. The orbital motion of the sensor still causes the objects to shift in position with respect to the FOR, but the shift is dramatic as that seen in the SOR measurement space; the same orbit positions used in Figures 3.8(a) and 3.8(b) are used in Figures 3.9(a) and 3.9(b), respectively. In the SME system, the shift of the objects with respect to the FOR is seen by looking at the declination axis; in Figure 3.9(a) the objects are distributed about $\delta_s = 0°$, while in Figure 3.9(b) they are distributed about $\delta_s \approx -6.5°$. In both cases the orientation of the GEO objects is very similar.

Because the space-based sensor is in LEO (orbital period of 97.6 minutes), it passes the objects at GEO multiple times during one day. This results in multiple opportunities for the sensor to observe each of the objects, but does not guarantee that all of the objects can be observed. The ground-based sensor is only able to see the objects that are in its FOR during the observation window; some of the objects remain in the FOR for long periods of time, while others are only visible for short periods of time. These factors will be considered during the simulations presented in Chapter 5.

3.2.3 Uncertainty Transformation

As previously discussed, every object has an associated mean and covariance that are propagated using an EKF. For this work, the system dynamics are limited to



(a) The FOR and expected measurements of the target objects in the SME system when the sensor is near its line of nodes.

(b) The FOR and expected measurements of the target objects in the SME system when the sensor is near its highest orbital position.

Figure 3.9. As the sensor orbits, the GEO objects appear to shift in position with respect to the FOR.

the two-body motion discussed in section 2.1, which are applied in the ECI coordinate frame. The optimization of the sensor tasking problem is performed in the chosen measurement space of the observer. Thus, the PDF must be transformed into the measurement space in order to perform the optimization. There is a non-linear relationship between the angles, in each of the chosen measurement spaces, and the position coordinates, in the frame of propagation; additionally, the full transformations of the uncertainty between the frame of propagation and the measurement spaces is non-linear.

The covariance is rotated into the measurement spaces via the Jacobian matrix, $\mathbf{H}_k$, which is specific to each measurement space. The Jacobian matrix is a linear

operator, which provides a linear approximation of the non-linear transformation. As an example, the LME measurement space, $\mathbf{H}_k$, is:

$$\mathbf{H}_k = \begin{bmatrix} \frac{\partial \tau}{\partial x} & \frac{\partial \tau}{\partial y} & \frac{\partial \tau}{\partial z} & \frac{\partial \tau}{\partial \dot{x}} & \frac{\partial \tau}{\partial \dot{y}} & \frac{\partial \tau}{\partial \dot{z}} \\ \frac{\partial \delta}{\partial x} & \frac{\partial \delta}{\partial y} & \frac{\partial \delta}{\partial z} & \frac{\partial \delta}{\partial \dot{x}} & \frac{\partial \delta}{\partial \dot{y}} & \frac{\partial \delta}{\partial \dot{z}} \end{bmatrix} \tag{3.16}$$

where the elements are the partial derivatives with respect to the position coordinates (e.g. $\partial\tau/\partial x$) evaluated at the mean state of the object, $\mathbf{m}_k(i)$, and k represents a discrete point in time. Because the transformation of a PDF from ECI to the measurement spaces is known to be non-linear, the transformation using the linear operator $\mathbf{H}_k$ will result in some error. However, if the transformations are well approximated as linear, then the transformed PDF may still be used in the sensor tasking optimization problem.

The measurement spaces are two dimensional, while the ECI coordinate frame is six dimensional. Thus, the Jacobian matrices are rectangular (2×6), in order to transform the 6×6 Cartesian covariance into a 2×2 covariance in the measurement space. The transformation of the covariance to the measurement space is given by:

$$\mathbf{P}_z = \mathbf{H}_k \cdot \mathbf{P} \cdot \mathbf{H}_k^T \tag{3.17}$$

where $\mathbf{P}$ is the object covariance in the ECI frame, $\mathbf{P}_z$ is the object covariance in the measurement space, and $\mathbf{H}_k$ is the Jacobian matrix (e.g. equation 3.16, in LME). The partials that make up the elements of the Jacobian matrices for each measurement space are given in Appendix A. For each of the measurement spaces used, the angles do not depend on the velocity, and so, the partial derivatives with respect to the velocity coordinates (e.g. $\partial\delta/\partial\dot{z}$) are all zero.

To check if the transformation between the state space and each measurement space can be approximated as linear, a Monte Carlo (MC) analysis of a single target object is performed; a conservative approach to representing an object's full state uncertainty is by using $N_p = 10^n$ particles in a MC analysis, where n is the dimension

of the state. For this analysis, the object PDF is initialized with a mean state and a Gaussian ($\mathcal{N}(\mathbf{m}, \mathbf{P})$) covariance and then propagated for GO HERE. A random number generator is used to generate one million probable states (or particles), $\mathbf{X}_p$, for the object in the state space, based on the state and covariance after the propagation.

The $\mathbf{m}$ and each of the $\mathbf{X}_p$ are transformed to the measurement space by the appropriate pair of equations in section 2.2.1 or 2.2.2. The location of the measurements, $\mathbf{z}_p$, corresponding to the $\mathbf{X}_p$ will provide a true representation of the uncertainty transformation from the state space to the measurement space. The covariance is transformed to the angle space by equation 3.17 and, along with the measurement derived from the mean state ($\tilde{\mathbf{z}}$), represents the linear approximation of the PDF in the measurement space.

If the PDF transformation is well approximated as linear, the $\mathbf{z}_p$ will be well aligned with the σ (standard deviation) intervals of the transformed PDF ($\mathcal{N}(\tilde{\mathbf{z}}, \mathbf{P}_z)$) in the measurement space. To check this mathematically, the squared Mahalanobis Distance (MD) is introduced as:

$$MD = (\mathbf{z}_p - \tilde{\mathbf{z}}) \cdot \mathbf{P}_z \cdot (\mathbf{z}_p - \tilde{\mathbf{z}})^T \tag{3.18}$$

where equation 3.18 calculates the MD in the measurement space [55, 56]. The MD measures of how far a point is from a distribution in variances (σ^2) of the distribution [56].

If the transformation of the PDF is linear, then the mean of the MD for all of the MC particles will be equal to the dimension of the state:

$$\widetilde{\mathrm{MD}} = \frac{1}{N_p} \sum_{j=1}^{N_p} \mathrm{MD}_j = 2 \tag{3.19}$$

where the measurement spaces are all two dimensional. If $\widetilde{\mathrm{MD}}$ is nearly equal to the dimension, then the linear transformation of the PDF provides a valid approximation of the true PDF in the measurement space.

To test how well the transformed PDFs approximate the true PDFs, a single object is used to perform the analysis for both the ground-based and space-based sensors. For the ground-based sensor, the object chosen is INSAT 2D[1], from the TLE catalog of 26 September 2016. The initial mean is generated using an SGP4 propagation to 0000h UT on 27 September 2016, and the object is fitted with an initial covariance where the position uncertainty is $\sigma_{x,y,z} = 50$ kilometers and the velocity uncertainty is $\sigma_{\dot{x},\dot{y},\dot{z}} = 10$ meters per second. The value for the initial position uncertainties is chosen to represent the limited accuracy of the TLE that generated the initial mean state, as well as the errors that have developed since the epoch of the TLE. The value for the velocity uncertainties are intended to be large enough to highlight the non-linearities that will occur during the transformations, without being unrealistically large. The initial mean and covariance of the object is propagated under two-body dynamics for 43.5 hours, and then one million particles are generated in the ECI coordinate frame; these particles represent the true PDF of the object. Finally, the particles and the resulting PDFs are transformed into each of the measurement spaces.

Figures 3.10-3.12 provide the graphical representation of the transformations into each of the measurement spaces. A sixth-order polynomial fit of the particle locations in each measurement space is also included, which shows that none of the transformations are truly linear. However, visual inspection of the particles in Figure 3.10, appears to show good alignment with the σ intervals through the middle of the transformed PDF in the LME measurement space. The polynomial fit shows a linear approximation through the center most portion of the PDF, though not perfectly aligned with the axis; the non-linearities in the transformation are most evident at the tails, near the 3-σ intervals.

In Figure 3.11, the EVE measurement space shows the same behavior as the LME measurement space, though the orientations of the PDF and particles are mirrored. The orientation difference is due to EVE being a right-handed system and LME a

[1]INSAT 2d: NORAD ID - 24820, International Designator - 1997-027B $i = 13.37°$, $a = 40,943$ km, $e = 0.0324$.

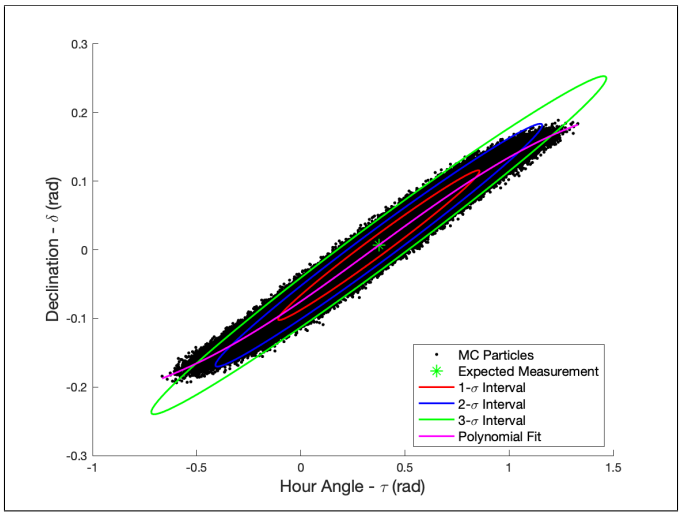


Figure 3.10. Transformed particle measurements in the LME system plotted against 1,2,3-σ intervals, with a polynomial fit of the probable measurements.

left-handed system. The only other difference between the two is a rotation about the third axis, which does not affect the transformation of the particles and the PDF, other than to create a difference in the angular values of α and τ.

In Figure 3.4, the LMLH measurement space shows very non-linear alignment between the particles and PDF after the transformation. The LMLH measurement space differs from the LME by a rotation about the second axis ($\mathbf{R}_2(\pi/2 - \phi)$ in equation 2.18); from this, it is inferred that the rotation of the reference plane is the primary cause of the increased non-linear behavior of the transformation.

Table 3.1 provides the $\widetilde{\text{MD}}$ of the particle measurements in each of the measurement spaces being considered. The row titled "ideal" represents the mean MD for a set of particles, assuming no loss is experienced in the transformation. The mean MD value for the LMLH system is drastically larger than the ideal value, indicating the highly non-linear transformation for this system. The values for the LME and EVE systems are close to the ideal value, both being 7.53% smaller than the ideal. Along with the alignment of the particles and the transformed PDFs in Figures 3.10 and

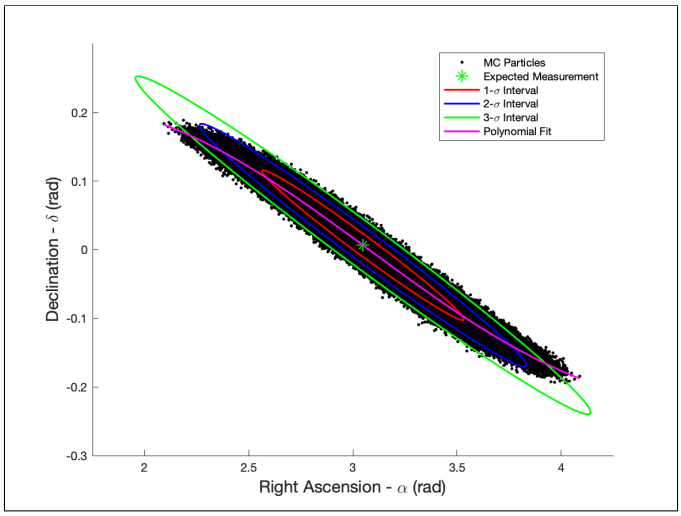


Figure 3.11. Transformed particle measurements in the EVE system plotted against 1,2,3-σ intervals, with a polynomial fit of the probable measurements.

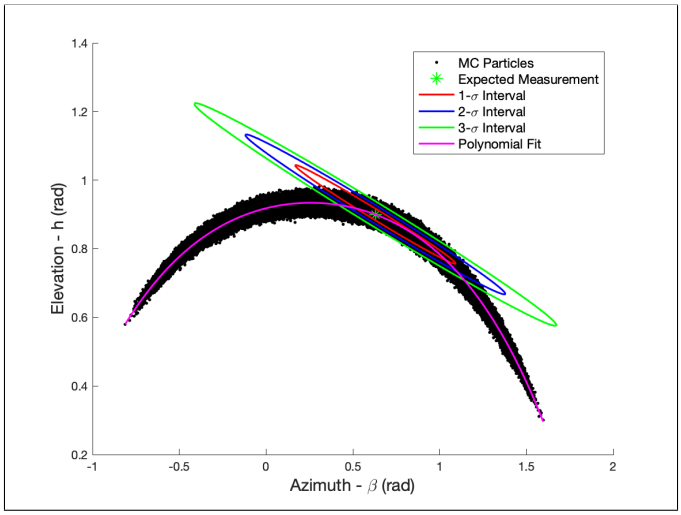


Figure 3.12. Transformed probable measurements in the LMLH system plotted against 1,2,3-σ curves, with a polynomial fit of the probable measurements.

3.11, these values being close the ideal value leads one to infer that the transformation to the EVE and LME systems can be approximated as linear.

Table 3.1. The Mean MD for the probable states in the ground-based sensor measurement spaces. All values are measured in variances, σ^2, for the transformed distribution.

Measure	Mean MD
Ideal	2.0000
LME	1.8408
EVE	1.8408
LMLH	106.8925

For the space-based sensor, the object chosen was SATCOM 2[2], from the TLE catalog of 26 September 2016. Again, SGP4 is used to generate the initial mean at 0000 hours UT on 27 September 2016, the same uncertainties are used for the covariance. Again, the PDF is propagated for 43.5 hours under two-body motion, and the same Monte Carlo analysis was applied with the space-based sensor measurements spaces.

Figure 3.13 presents the transformation into the SME measurement space. While many of the particles are closely aligned with transformed PDF, there are sizable deviations near the ends of the particle distribution. The polynomial fit of the particle locations also shows a noticeable curve; the non-linearity is more clearly seen in the SME measurement space than in the LME measurement space.

Like the LMLH measurement space, the SOR shows very non-linear alignment of the particles and PDF after the transformation in Figure 3.14. The SOR coordinate system also requires a rotation of the reference plane, this time about the first axis, from which it can again be inferred that the rotation of the reference plane away from parallel to Earth's equator is the primary source of the larger non-linearity in the transformations to the LMLH and SOR coordinate system.

Table 3.2 provides the $\widetilde{\text{MD}}$ of the particle measurements in each of the measurement spaces being considered; the ideal row is the same as in Table 3.1. The $\widetilde{\text{MD}}$

[2]SATCOM 2: NORAD ID - 8774, International Designator - 1976-029A, $i = 13.85°$, $a = 42,634$ km, $e = 0.0059$.

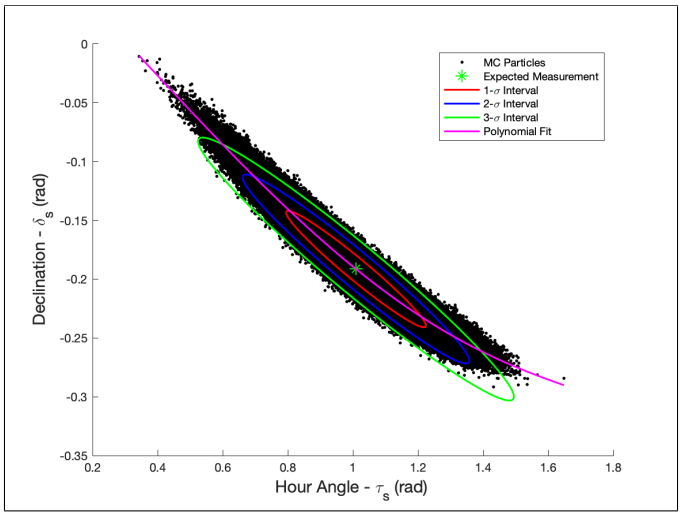


Figure 3.13. Transformed particle measurements in the SME system plotted against 1,2,3-σ intervals, with a polynomial fit of the probable measurements.

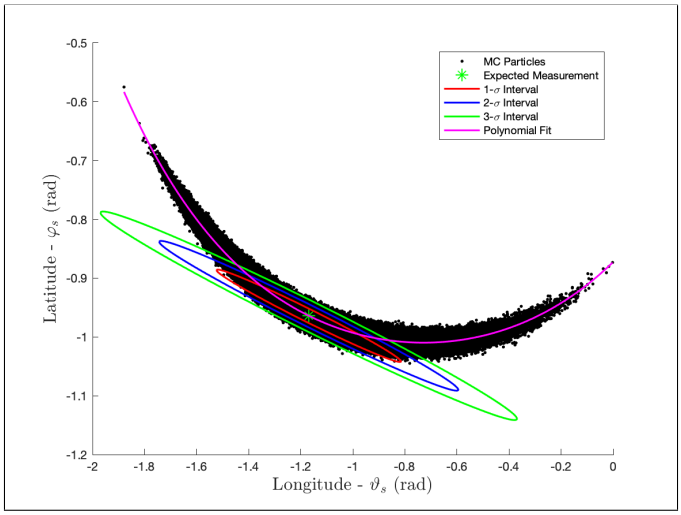


Figure 3.14. Transformed probable measurements in the SOR system plotted against 1,2,3-σ curves, with a polynomial fit of the probable measurements.

for the SME coordinate system is 6.13% larger than the ideal value. From Figure 3.13, the non-linearity is apparent, but with the small difference in the $\widetilde{\text{MD}}$ value,

the author is choosing to make the assumption that the transformation to the SME coordinate system can also be approximated as linear. The SOR coordinate system has a large difference in the $\widetilde{\text{MD}}$ value. Therefore, the transformation can not be approximated as linear.

Table 3.2. The Mean MD for the probable states in the space-based sensor measurement systems. All values are measured in variances, σ^2, for the transformed distribution.

Measure	Mean MD
Ideal	2.0000
SME	2.1227
SOR	12.7979

Based on the analyses above, the LME and SME coordinate systems are chosen for the measurement spaces of the ground-based and space-based sensors, respectively. These system show the closest approximation of linearity in the transformation of the PDFs. Additionally, both measurement systems are static with respect to the observer, only requiring the definition of the FOR to be generated once; the EVE system requires re-generating the representation of the FOR for the ground-based sensor at every observation time.

3.3 Summary of Orbit Uncertainty

The estimation of orbit states is inherently filled with uncertainty. This dissertation attempts to optimize the sensor tasking problem based on the probability of seeing as many objects as possible from a catalog, during a single observation period. In order to accomplish this, a fast and accurate method for transforming the uncertainties is sought in order to use the uncertainty to determine the best viewing directions for the sensor. Under the stated assumptions, the LME and SME coordi-

nate frames satisfy research objective 2 for the ground-based and space-based sensors, respectively. These coordinate frames will be used in the remainder of this work.

4. SENSOR TASKING OPTIMIZATION

Generating optimal survey strategies for employing sensors is a continued area of research, with many proposed solutions that depend on the desired outcome [7, 14, 31, 35, 36, 41, 43, 43–48]. The complexity of the problem involves where to point a given sensor, when to observe a particular RSO, how to balance follow-up and survey observations, and many other aspects [15, 30, 33, 43]. This chapter will introduce the formulation of the problem, the simplifications that are employed, and the optimization techniques which have been applied to solve the problem.

4.1 Sensor Tasking Formulation

In this work, the problem is stated as a maximization problem, based on the formulation from Frueh, et al. (2018), which is given by:

$$\begin{aligned}\max A = \sum_{g=1}^{l}\sum_{h=1}^{r_g}\sum_{f=1}^{m_g}\sum_{t_{g,f}=1}^{j_g}\Big[\Big(\sum_{i=1}^{n}\mu_{past}\big(\alpha_i(t_{g,f}),\delta_i(t_{g,f})\big)\\ \cdot\, p_d\big(\alpha_i(t_{g,f}),\delta_i(t_{g,f}),\boldsymbol{o},t_{g,f}\big)\cdot d\big(\alpha_{g,f},\delta_{g,f},\alpha_i,\delta_i,t_{g,f}\big)\Big)\\ + k\big(\alpha_{g,f},\delta_{g,f},t_{g,f}\big)\Big]\end{aligned} \tag{4.1}$$

$$\alpha_{g,f} - \tfrac{1}{2}\mathrm{FOV} - \alpha_i(t_{g,f}) \leq 0 \tag{4.2}$$

$$-\alpha_{g,f} - \tfrac{1}{2}\mathrm{FOV} + \alpha_i(t_{g,f}) \leq 0 \tag{4.3}$$

$$\delta_{g,f} - \tfrac{1}{2}\mathrm{FOV} - \delta_i(t_{g,f}) \leq 0 \tag{4.4}$$

$$-\delta_{g,f} - \tfrac{1}{2}\mathrm{FOV} + \delta_i(t_{g,f}) \leq 0 \tag{4.5}$$

$$\mathbf{R} - \sigma(\alpha_i, \dot{\alpha}_i, \delta_i, \dot{\delta}_i, \rho_i, \dot{\rho}_i, \nu) \leq 0 \tag{4.6}$$

where $A : \mathbb{R}^{k+4} \rightarrow \mathbb{R}$ is sought to be maximized. l is the number of sensors available during the optimization window, and r_g is the number of operations executed by sensor g during the optimization window. r_g provides for the flexibility of considering a multi-use sensor, where the operating time may be divided between multiple users, and for accounting for long optimization intervals where a sensor may not be able to operate continuously (e.g. during the day for optical sensors) [7]. m_g represents the total number of viewing directions, or observations, that are assigned for sensor g in during each individual operational time frame, r_g; during each observation, m_g, the sensor takes j_g images. r_g, m_g, and j_g provide the time discretization for a given sensor, and there is no reason that the time discretization for any two sensors will be the same.

The final summation is over the n objects which are to be observed during the optimization interval. The $\mu_{past} \in [0, 1]$ term is an object specific function that provides a way to weight the desire to see each object. For $\mu_{past} = 1$ the object is sought to be observed, and for $\mu_{past} = 0$ it will be ignored; other values of μ_{past} can be used to increase or decrease the influence of a given object on the optimization based on some factor tied to the orbit quality, e.g. observability [15]. The $p_d(\alpha_i(t_{g,f}), \delta_i(t_{g,f}), \boldsymbol{o}, t_{g,f})$ is the probability of detection of each object. As discussed in section 2.4, it is dependent on the geometry between the object and observer $(\alpha_i(t_{g,f}), \delta_i(t_{g,f}))$, and on other object specific parameters which may be represented by $\boldsymbol{o}$ [7]. The $d(\alpha_{g,f}, \delta_{g,f}, \alpha_i, \delta_i, t_{g,f})$ represents the association between a viewing direction $(\alpha_{g,f}, \delta_{g,f})$ and the object location (α_i, δ_i) at time $t_{g,f}$; $d = 1$ if equations 4.2-4.5 are satisfied, and zero if any of them are violated. The final constraint provides another threshold for the association, d; σ is an object specific function that, at a minimum, includes the full state of the object, but could also represent the object PDF covariance or other similar measures. $\mathbf{R}$ is a threshold, such that when σ exceeds $\mathbf{R}$, d can be set to one and otherwise it will be zero despite satisfaction of equations 4.2-4.5.

Frueh et al. (2018) include the function $k(\alpha_{g,f}, \delta_{g,f}, t_{g,f})$ to account for beginning from no known objects. k is a weighting function which could be a probability surface

that indicates where hypothesized objects may exist [19]. Including the k function also allows the method to optimize with survey observations at times when no known objects are desired to be seen (e.g. $d = 0 \; \forall \; i$).

For this work, the observation intervals are assumed to be single nights with no breaks during the interval ($r_g = 1$). Additionally, to simplify the computations, the individual images are not considered, but instead the observation is assumed to occur at the midpoint of the image exposures. Thus, equation 4.1 simplifies to:

$$\max A = \sum_{g=1}^{l} \sum_{f=1}^{m_g} \cdot \Big(\sum_{i=1}^{n} \mu(\widetilde{\mathbf{Z}}_i, \mathbf{P}z_i) \cdot p_d(\widetilde{\mathbf{Z}}_i) \cdot d(h_{g,f}, \widetilde{\mathbf{Z}}_i, \mathbf{P}z_i) \Big) \tag{4.7}$$

where A represent the expected value of the RSOs to be observed. Equation 4.6 is not used in this work, and equations 4.2-4.5 are replaced by the CDF value of an object for a given viewing direction, where $d \in [0, 1)$. In other words, $d(h_{g,f}, \widetilde{\mathbf{Z}}_i, \mathbf{P}z_i)$ is the associated probability that object i falls within the field of view (FOV) for a pointing direction, $h_{g,f}$, as calculated by equation 4.10. The formulation of Frueh et al. (2018) is based on the EVE coordinate frame (α, δ) using the mean states of the target objects. Since this work uses the PDF transformed into the measurement spaces, the angles in equation 4.1 have been replaced by the object mean and covariance as represented in the measurement space $(\widetilde{\mathbf{Z}}_i, \mathbf{P}z_i)$, and the angles representing the pointing direction, $h_{g,f}$; this work will also employ the LME coordinate system for ground-based sensors, and the SME for space-based sensors. The object specific value $\mu(\widetilde{\mathbf{Z}}_i, \mathbf{P}z_i)$, hereafter μ_i, is simplified to indicate if object i has been observed during the current window ($\mu_i = 0$), or not ($\mu_i = 1$).

This problem defined in equation 4.7 suffers from the *curse of dimensionality*, where the optimization of the solution becomes more difficult as the population of the catalog (n) grows, as the duration of the observation window (m_g) increases, and as more sensors (l) are incorporated into a single optimization problem. The number of viewing directions, m_g, is dependent on the length of the observation window and the time between observations. There are many possible limitations to the duration of an observation window (e.g. time scheduled on a shared sensor or

length of the night) [7]. Once the duration of the observation window is known, it is broken up into the number of viewing directions which can be assigned. While it is not required, in this work the time between observations is fixed, so that the duration of the observation window is divided by the time between observations to determine m_g. The time between observations is given by:

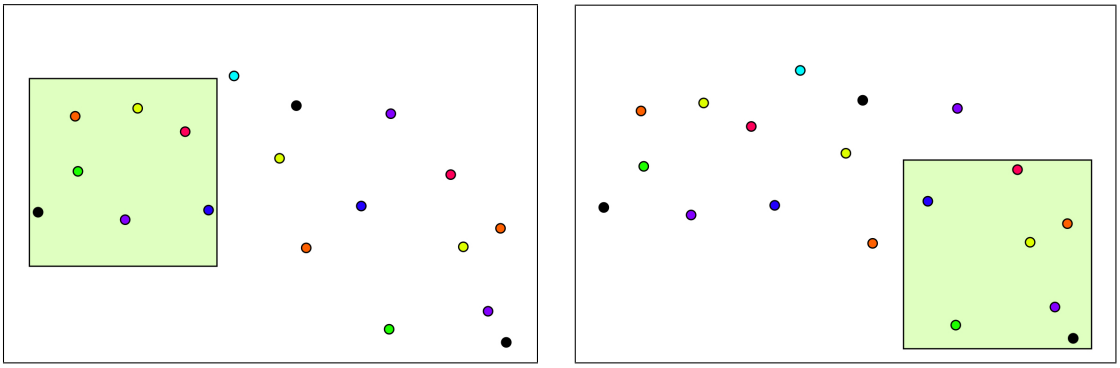
$$t_{obs} = j \cdot t_{exposure} + (j-1) \cdot t_{readout} + t_{repos} \tag{4.8}$$

where j represents the number of images taken at each observation, $t_{exposure}$ is the length of the exposure for each image, $t_{readout}$ is the time required to readout the image after exposure, and t_{repos} is the time to re-position the sensor from where it is currently pointed to where it will point at the next viewing direction [7]; t_{repos} includes the time for the vibrations to settle out after the sensor is moved. The readout times are multiplied by $j-1$ because it is assumed that the final image is read out while the sensor is being re-positioned ($t_{repos} > t_{readout}$) [7].

4.1.1 Discretization of Field of Regard

Taking the Field of Regard as a continuous space, there are potentially an infinite number of pointing directions that may be assigned to the sensor. If any pointing direction is allowed, Figure 4.1 illustrates how two possible pointing directions could find the same number of objects. The choice of pointing direction now becomes another possible optimization choice. By discretizing the FOR into a grid of allowed pointing directions, the optimization complexity is reduced from an infinite space to a discrete set of grid fields, making the problem tractable [7].

The grid for the ground-based sensor is defined for the Purdue Optical Ground Station (Lat: 32.903294° N, Long: 105.528590° W), in the LME measurement space, while the grid for the space-based sensor is defined in the SME measurement space for a theoretical sensor using the International Space Station orbit (a = 6784 km, $e = 3.96 \times 10^{-4}$, $i = 51.64°$, $\Omega = 246.64°$, $\omega = 198.23°$). Both sensors assume an



(a) Choose to observe left most objects.

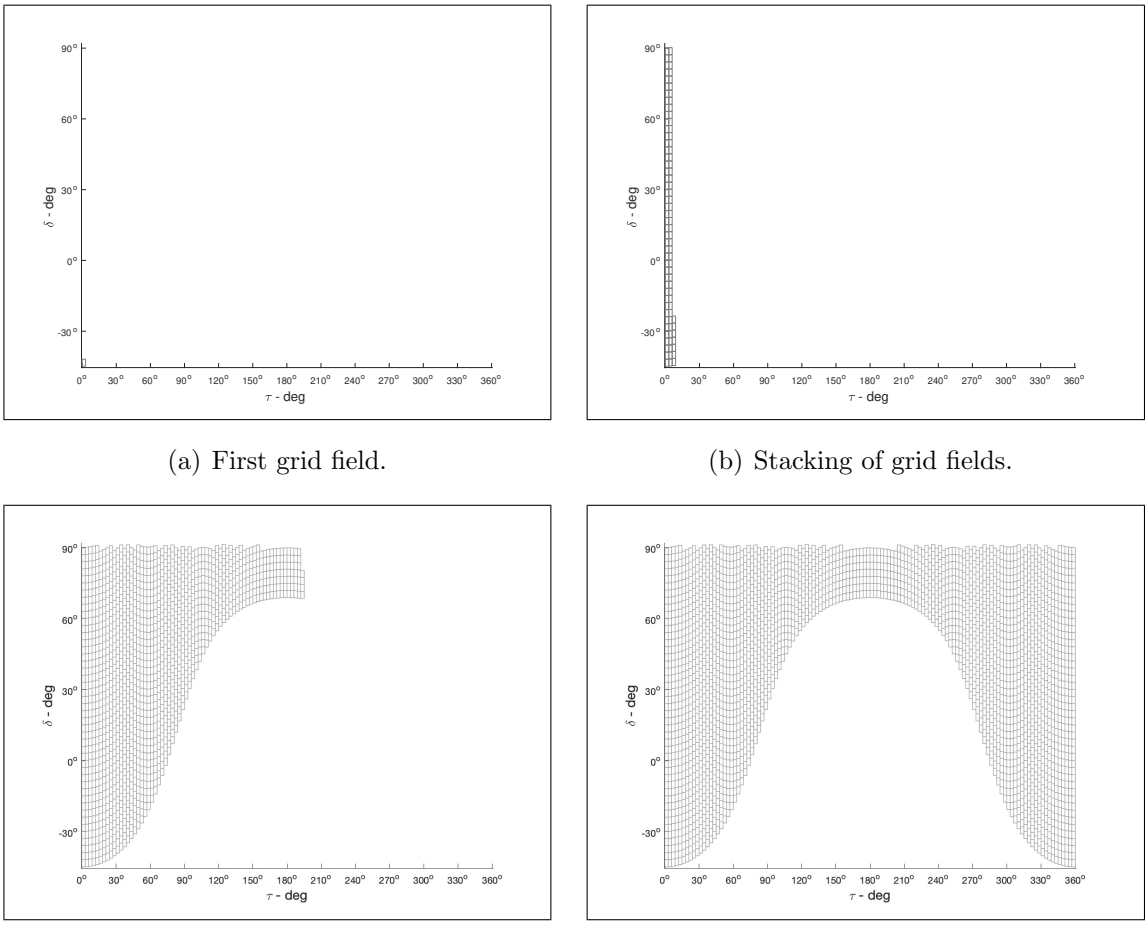
(b) Choose to observe right most objects.

Figure 4.1. A cartoon example of choosing the pointing direction based on observing multiple objects at once.

effective FOV of 3° square, which is used to discretize the FOR. The ground-based sensor also has the minimum elevation $h_{min} = 12°$, below which observations may not be taken. Each grid is formed by dividing the sensor FOR into stripes of pointing directions based on the angle τ, for the ground-based sensor, and τ_s, for the space-based sensor.

The number of stripes in the LME grid is determined by dividing the span of τ by the FOV size ($360/3 = 120$). The first stripe is centered on $\tau = 1.5°$, so the left edge of the effective FOV is at $\tau = 0°$; each successive stripe is one FOV width away from the previous stripe $(4.5°, \ldots, 358.5°)$. The declination of the first pointing direction in each stripe is defined by finding the δ_{min} from equation 3.15, for the associated τ value, and adding half the FOV height; this places the bottom edge of the first grid field at the minimum declination. Holding the τ constant, the pointing directions are then stacked in δ, by addition of the FOV height, until they reach $\approx 90°$. Figure 4.2 shows the generation of the LME grid for the ground-based sensor.

The SME grid is not restricted by a minimum declination, so the definition of the stripes is more straight forward. The first grid field in the first stripe is located at $(\tau_s, \delta_s) = (-73.5°, -88.5°)$, and successive grid fields are stacked in δ_s until $\delta_s = 88.5°$.



(a) First grid field.

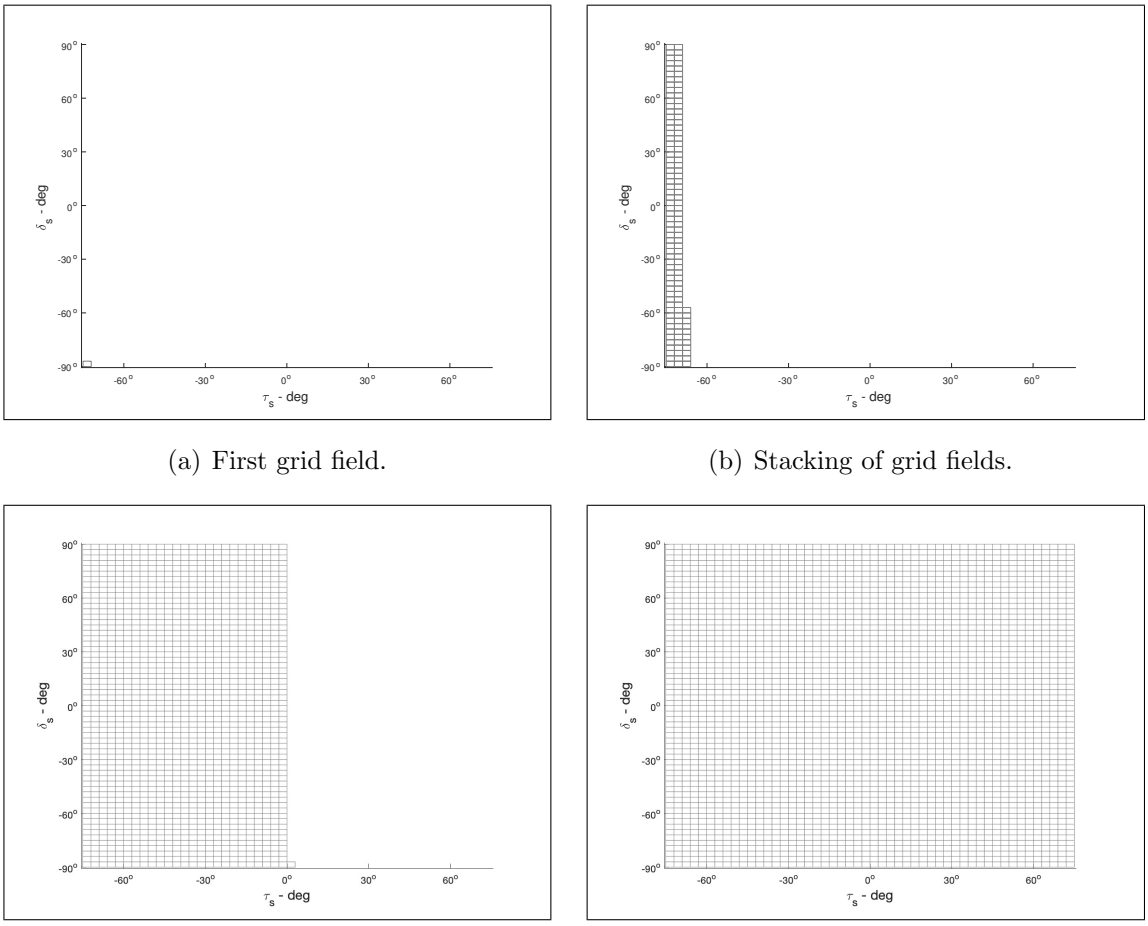
(b) Stacking of grid fields.

(c) Midway through grid creation.

(d) Complete grid.

Figure 4.2. The generation of the grid in LME is shown; each grid field represents the effective field of view for one of the allowed pointing directions.

The next stripe is then started at $(\tau_s, \delta_s) = (-70.5°, -88.5°)$ and the process repeats generating the complete grid. Figure 4.3 shows the generation of the SME grid for the space-based sensor.



(a) First grid field.

(b) Stacking of grid fields.

(c) Midway through grid creation.

(d) Complete grid.

Figure 4.3. The generation of the grid in SME is shown; each grid field represents the effective field of view for one of the allowed pointing directions.

4.1.2 Cumulative Distribution Function in the Measurement Space

In section 3.2.3, the transformation of an object PDF from the ECI frame, where propagation occurs, to the measurement space is described, and the LME and SME are shown to be the most accurate frames for the transformations. With the grids

defined in the previous section, the probability that each object falls in each grid field is calculated by integrating the PDF over the grid field:

$$d(h, \widetilde{\mathbf{Z}}_i, \mathbf{P}z_i) = \int_h \frac{1}{\sqrt{(2\pi)^2 |\mathbf{P}z_i|}} \exp\Big(- \frac{(\mathbf{Z}_{i,h} - \widetilde{\mathbf{Z}}_i)^T \mathbf{P}z_i^{-1} (\mathbf{Z}_{i,h} - \widetilde{\mathbf{Z}}_i)}{2} \Big) \mathrm{d}h \qquad (4.9)$$

where, the summation represents each of the objects in the catalog, $\widetilde{\mathbf{Z}}_i$ is the object mean in the measurement space, $\mathbf{P}z_i$ is the transformed covariance, h indicates the grid field over which the integration is performed, and $\mathbf{Z}_{i,h}$ are the possible measurements of the object in the grid field; the grid field edges provide the intervals for the integration. Combining the results of equation 4.9 for every object provides a combined Cumulative Distribution Function (CDF) value for each grid field.

$$V(h) = \sum_{i=1}^{n} d(h, \widetilde{\mathbf{Z}}_i, \mathbf{P}z_i) \qquad (4.10)$$

Equation 4.10 provides the combined probabilities of seeing objects in every grid field; however, it does not guarantee objects will be observed in a chosen grid field.

Figure 4.4 shows the combined CDF values of the LME grid for a catalog of objects, at one moment in time, as calculated by equations 4.9 and 4.10; the blue dots show the location of the expected measurements ($\widetilde{\mathbf{Z}}_i$) for the objects. As expected, the combined CDF values generally align with the catalog of the objects. Combining the individual probabilities in this way provides a method for determining the viewing direction with the highest combined probability of observing objects.

The focus of this work is optimizing the sensor tasking by using the formulation presented in equation 4.7, and by taking into account all of the objects in the catalog and the uncertainty in the locations of those objects. To reduce the dimensionality of the problem, the time between observations is fixed and the FOR is discretized, as discussed, for each sensor. The rest of this chapter seeks to analyze efficient methods for generating the solutions; some of these efficient solution methods assume convexity in the problem. The following section discusses convexity and why it does not apply to the sensor tasking problem. Then two classical optimization algorithms

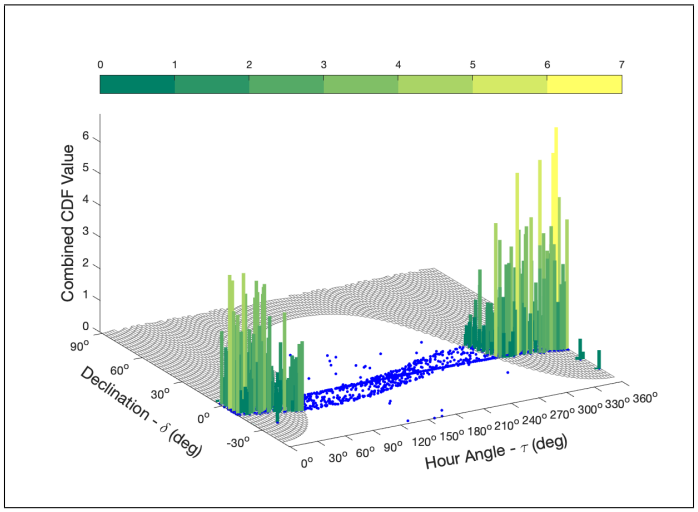


Figure 4.4. The combined cumulative distribution function values combine the likelihood that an object is in the grid field and the value gained by observing that object.

(which assume convexity), and two reinforcement learning based algorithms will be described and formulated to solve the sensor tasking problem.

4.2 Convexity

Beginning in the 1930s, optimization techniques became an area of great research interest and many advances have been made over the years [57–59]. Convex optimization involves a special class of problems whose solutions are theoretically more simple to determine [57, 59]. This section discusses the conditions required for convexity, specifically for discrete systems.

The general convexity condition for a continuous function, is given by:

$$\lambda f(x) + (1 - \lambda) f(y) \leq f(\lambda x + (1 - \lambda) y) \tag{4.11}$$

where, $f(\cdot)$ represents the functions, x and y represent points where the function is evaluated, and λ is a value between zero and one. Equation 4.11 is formulated for a maximization problem, because the sensor tasking problem in this work is treated as

such; typically, equation 4.11 is given for a minimization problem, where $\leq$ is replaced by $\geq$ (as in Ref. [57]).

The continuous optimization problem need not be restricted to a single dimension for x and y. It can be stated as $x, y \in S$, $S \subseteq \mathbb{R}^n$, where n is the dimension of the points. Then f takes the points in and produces a function value as an output, $f : S \to \mathbb{R}$. There are many powerful techniques that are used to solve continuous convex problems, but the same techniques often fail when applied to a discrete problem because the characteristics of the problem are different [59]. Midpoint convexity is a characteristic of continuous convex problems that does not hold for discrete convex problems. Midpoint convexity is defined by setting $\lambda = \frac{1}{2}$ in equation 4.11, which after rearranging leads to [57, 59, 60]:

$$f(x) + f(y) \leq 2\, f\Big(\frac{x+y}{2}\Big) \tag{4.12}$$

Equation 4.12 is equivalent to the general convexity condition for a continuous convex function per Theorem 2.5 on page 224 of Murota (2009) [60]. Instead of using "some" point between x and y, the point exactly half way between them is used.

In contrast to continuous convex functions, discrete convex functions are defined as $x, y \in S$, $S \subseteq \mathbb{Z}^n$, for $g : S \to \mathbb{R} \cup \{+\infty\}$ or $g : S \to \mathbb{Z} \cup \{+\infty\}$ depending on if the function, g, is real-valued or integer-valued[1] [59]. Equation 4.12 will fail for the discrete function g, if x and y are adjacent points in the set S, because the midpoint between them will not be in S. Additionally, there is no guarantee that the midpoint between any two points in the set S, will also be in S; this is shown in Figure 4.5, with the vector points x and y replaced by p and q [59, 60]. The three cases in Figure 4.5 represent different configurations of the problem, but in each case the midpoint can only be approximated by points representing the componentwise rounding of the vector midpoint to $\lceil \frac{p+q}{2} \rceil$ and $\lfloor \frac{p+q}{2} \rfloor$ ($\lceil\ \rceil$ and $\lfloor\ \rfloor$ represent componentwise rounding up and down to the nearest integer values, i.e. the ceiling and floor functions).

[1] Generally speaking, the elements of S should not need to be integers ($\in \mathbb{Z}$). Rather, they should be required to have finite differences between an element of S and every other element of S.

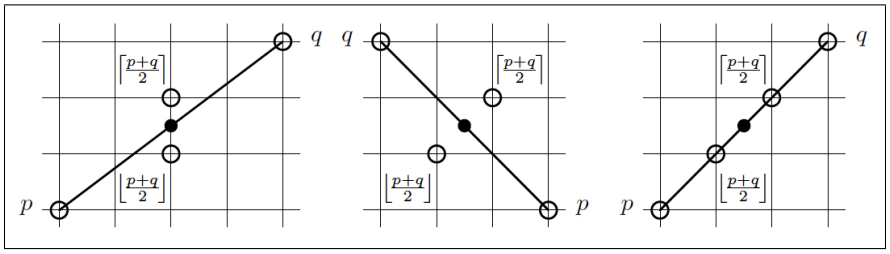


Figure 4.5. The *midpoint* between two points may not be within the set, $S \subseteq \mathbb{Z}^n$, of allowable points (From Ref. [59], Figure 1.9, pg. 23).

In order to develop the expression that is analogous to equation 4.12, Murota defines $L^\natural$-convex functions which are a class of discrete functions that satisfy *discrete midpoint convexity* as given by:

$$g(x) + g(y) \leq g\left(\left\lceil \frac{x+y}{2} \right\rceil\right) + g\left(\left\lfloor \frac{x+y}{2} \right\rfloor\right) \tag{4.13}$$

where, again, $\lceil \frac{x+y}{2} \rceil$ and $\lfloor \frac{x+y}{2} \rfloor$ are component wise rounding of the vector $\frac{x+y}{2}$. Equation 4.13 can also be stated as the sum of the function values of the two points closest to the midpoint will be less than or equal the sum of the function values for the original points. The full development of the discrete midpoint convexity is given in Ref. [59].

Another characteristic of continuous convex problems that does not hold for discrete convex problems is *equidistance convexity* [60]. Equidistance convexity is defined for a continuous function f, by adding equation 4.11 with $\lambda = \alpha$ and equation 4.11 with $\lambda = 1 - \alpha$ together (where $0 < \alpha < 1$), which leads to [59, 60]:

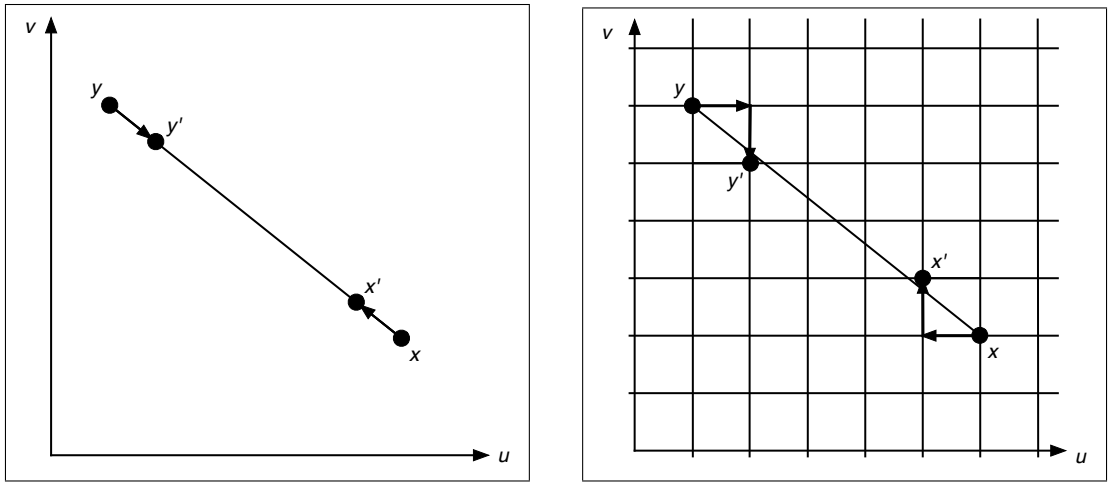
$$f(x) + f(y) \leq f(x - \alpha(x - y)) + f(y + \alpha(x - y)) \tag{4.14}$$

The result is that the sum of the function values at any two points will never increase if the two points are moved toward each other by equal distances along the line segment connecting them. As with the midpoint convexity, discrete functions can only be adjusted to points within the allowable set $S \subseteq \mathbb{Z}^n$.

In generating the discrete equivalent to equidistance convexity, Murota defines $M^{\natural}$-convex functions which are another class of discrete functions that satisfy the *exchange property*:

$$g(x) + g(y) \leq g(x - \chi_u + \chi_v) + g(y + \chi_u - \chi_v) \tag{4.15}$$
$$\text{given: } u \in \text{supp}^+(x - y)$$
$$v \in \text{supp}^-(x - y) \cup \{0\}$$

where, χ_u and χ_v are the characteristic vectors of the positive ($\text{supp}^+(x - y)$) and negative ($\text{supp}^-(x - y)$) support functions, respectively [59, 60]. The characteristic vectors shift the elements of the vector points x and y in the proper directions to ensure that the new points approach each other "along" the line segment connecting the original points. Figure 4.6 shows how the equidistance convexity and the exchange property compare [59, 60]. Again, the full development of the exchange property is given in Ref. [59].



(a) Equidistance Convexity.

(b) The Exchange Property.

Figure 4.6. A comparison of the equidistance convexity of continuous systems and the exchange property of discrete systems (Murota, K., *Recent Developments in Discrete Convex Analysis*, pg. 225, Figs. 11.3-11.4, 2009).

The $L^\natural$ and $M^\natural$-convex expressions given in equations 4.13 and 4.15 are generalizations that encompass a larger set of possible discrete convex functions than their L and M-convex equivalents [59, 60]. These expressions will be used as a way to determine if the sensor tasking problem is possibly convex. A full development of discrete convex analysis is given in Ref. [59] and a number of example problems are discussed in Ref. [60].

4.2.1 Convexity and the Sensor Tasking Problem

The problem of sensor tasking for SSA can be modeled as a discrete process in both the temporal and the spatial domain: the sensor is only capable of observing a small portion of the sky (spatial) at a given observation time (temporal). In this case, the points x and y in equations 4.13 and 4.15, are sets of viewing directions assigned to be observed. The grids introduced in section 3.2.3, provide the spatial discretization for the problem, dividing the FOR into set pointing directions for the sensor. Temporally, the sensor is only capable of taking one observation at a time, and then may be repositioned before taking another observation at a later time. These discretizations result in a high dimensionality for the sensor tasking problem [7, 12, 36, 44, 45].

In addition to the dimensionality of the problem, temporal coupling is present. The coupling is present because previously observed objects are not sought to be observed again within the same observation time window, but many of them are able to be observed at later times; therefore, the current tasking choice is coupled with the choices in the previous and future time steps. This coupling makes determination of a global optimal solution difficult to obtain.

For the sensor tasking problem, no general convexity condition is tested, but non-convexity is shown via a counter example, using equations 4.13 and 4.15. A full night scenario is run using 1231 GEO objects ($r_p \geq 35,378$km and $r_a \leq 45,378$km) selected from the publicly available Two-Line Element (TLE) catalog from 26 September

2016[2]. The Purdue Optical Ground Station (POGS) is modeled as the observer (Lat: 33° N, Long: 106° W) with a $3° \times 3°$ FOV, and the grid generated in the LME coordinate system[3]. Two-body propagation is used to propagate the objects to each time step during the observation interval from 1826 hours, 44.7 seconds on 28 September 2016 to 0514 hrs, 13.2 seconds on 29 September 2016 (times are in POGS local time). The observations are simulated at the midpoint of each step and no uncertainty is considered.

Three points (sensor tasking solutions) are generated using a greedy algorithm: x is the local optimal for each viewing direction, while y_1 and y_2 represent the second and third best choices for each viewing direction, respectively. The point pairs (x, y_1) and (x, y_2) are analyzed with equations 4.13 and 4.15 to check for convexity. A total of eleven different solutions (e.g. x, y_2, $\lceil \frac{x+y_1}{2} \rceil$) are generated in order to test both equations and the number of unique objects observed at least once, is determined for each of the resulting solutions; unique objects means that an object observed more than once is only counted once.

Table 4.1 shows the results of these convexity conditions for both pairs of points. The left sides of equations 4.13 and 4.15 are equal and provide the sum of the unique objects found by each point pair in the first row. The sum of the solutions for the adjusted the points used on the right hand side of equation 4.13 are provided in the second row, and those for the right hand side of equation 4.15 are provided in the third row. If the problem is convex, the values in the first row should be smaller than the values below them. The sensor tasking problem fails to achieve either discrete midpoint convexity or equidistance convexity.

The sensor tasking problem is often stated to be non-convex [36, 44, 45, 61]. From this analysis of the $L^\natural$ and $M^\natural$ convexity, that statement is valid. Thus, complex optimization techniques that do not assume convexity are expected to generate better solutions.

[2]Catalog used is from www.Space-Track.org.

[3]POGS position vector: $\bar{R}^{ECEF} \approx [-1434, -5161, 3466]^T$ km

Table 4.1. The values for the left and right sides of equations 4.13 and 4.15 are given for the pairs (x,y_1) and (x,y_2).

Measure	(x,y_1)	(x,y_2)
Left Side	717	664
$L^\natural$ Right Side	604	642
$M^\natural$ Right Side	487	501

4.3 Classical Optimization Methods

While the sensor tasking problem is not fully and always convex, computational advantages continue to make classical optimization techniques based on convexity a desirable tool. The following sections introduce the local optimal greedy and Weapon-Target Assignment algorithms, which are based on classical optimization methods and only guaranteed to produce optimal solutions for a truly convex problem that is temporally decoupled. While the temporal coupling exists, and the problem is known to be non-convex, these algorithms may still provide an advantage in the efficiency with which they produce solutions.

4.3.1 Greedy

The greedy algorithm used in this work is a very simple optimizer that only considers the momentary state of the observation space. It does not consider the rise and set times of the target objects, which could be used to adjust the value of observing objects at different times; considering the rise and set times of the objects would potentially improve the solutions of each of the optimizers [7]. The benefit of such a simple optimizer is that it is very computationally efficient.

The greedy algorithm determines the local optimal at each observation time by choosing the grid field with the highest weighting, h^*, and assigning that grid field

for the viewing direction a_f. The weighting of each grid field h is calculated by the inner summation in equation 4.7:

$$\sum_{i=1}^{n} \mu_i \cdot p_d \cdot d(h, \widetilde{\mathbf{Z}}_i, \mathbf{P}z_i)$$

where, if the positions of the RSOs are assumed to be perfectly known, then the term $d(h, \widetilde{\mathbf{Z}}_i, \mathbf{P}z_i)$ becomes $d(h, \mathbf{Z}_i)$, and represents the mean position of target object i falling within the grid field h or not; the algorithm then chooses the grid field with the most RSOs that have not previously been observed. When the uncertainty of the RSO positions is used to determine the weighting, the algorithm attempts to find the grid field that will contain the maximum combined probability of RSOs that need to be observed, via computation of the combined CDF as outlined in section 3.2.2, equation 4.10. The greedy algorithm then chooses h^* as [49]:

$$h^* = \arg\max_{h} \sum_{i=1}^{n} \mu_i \cdot p_d \cdot d(h, \widetilde{\mathbf{Z}}_i, \mathbf{P}z_i) \tag{4.16}$$

where μ_i is equal to one if the RSO has not been observed, and is set to zero after the RSO is observed. The probability of detection, p_d, scales the influence of each object, providing input on how likely it is the object will be seen, in addition to the probability that the object is in the grid field.

The greedy algorithm assumes that the local optimal choice at each step will lead to the global optimal solution [57, 59, 62, 63]. As mentioned above, this explicitly neglects temporal coupling and the non-convexity of the problem, leading to a sub-optimal solution. However, the computational efficiency is desirable, and thus the greedy solution is used as a point of comparison with the other algorithms.

4.3.2 Weapon-Target Assignment (WTA)

The WTA technique was presented by Hosein, et al. (1988) for the optimization of engaging incoming target weapons (called targets), with limited defensive weapons (called weapons) [64, 65]. The goal is to minimize the damage inflicted by the targets

on the asset being defended. There are two forms of the WTA problem: a) the static problem where all weapons are fired at the same time, and b) the dynamic problem, also called the shoot-look-shoot problem, where some weapons are fired, the results are assessed, then more weapons are fired [64, 65].

The static WTA problem is stated by Hosein et al. (1988, eqn. 2.1, pg. 2) as [64]:

$$\min_{x_{ij}\in[0,1]} J = \sum_{i=1}^{N} V_i \prod_{j=1}^{M} (1 - p_{ij})^{x_{ij}} \tag{4.17}$$

$$\text{subject to: } \sum_{i=1}^{N} x_{ij} = 1, \quad j = 1, 2, \ldots, M$$

where N is the number of targets, M is the number of weapons, V_i is the value of target i, p_{ij} is the probability of destruction for target j based on weapon i being fired at it, and x_{ij} represents if weapon i is assigned to target j ($x_{ij} = 1$), or not ($x_{ij} = 0$). The cost function, J, is referred to as the leakage, and represents the value of the surviving targets after the weapons are employed [64]. Hosein et al. (1988) state that equation 4.17 is proven to be NP-complete, though optimal solutions are possible in polynomial time under specific simplifying assumptions [64].

The dynamic WTA problem assigns the weapons in stages, and allows for adjusting the weapons used in the remaining stages [64]. Hosein et al. (1988) present the following steps for assigning the weapons at each stage (pg. 2):

a) "Determine which targets have survived the last engagement,"

b) "Assign and fire a subset of the remaining weapons with the objective of minimizing the total expected value of the surviving targets at the end of the final stage."

While Hosein et al. (1988) provide some specific scenarios, additional forms of the static and dynamic WTA problems are presented and discussed in Murphey (2000), each having specific scenarios where they may be implemented [65]. The WTA method developed in this work was developed based on two of the dynamics WTA methods from Murphey (2000): a) the shoot-look-shoot (DWTA-SLS) and b) the stochastic

demand (DWTA-SD) problems [65]. The DWTA-SD problem assumes that only a subset of the targets ($n(t) \leq N$) have been discovered at time t, and additional targets may be discovered by waiting some later time. However, there is also a cost $c(t)$ for waiting to target the objects. Murphey (2000, pg. 47) presents the DWTA-SD problem as [65]:

$$\min \sum_{t=1}^{T} c(t) \sum_{j=1}^{n(t)} V_j \prod_{i=1}^{M} q_{ij}^{x_{ij}(t)} \tag{4.18}$$

$$\text{subject to: } \sum_{t=1}^{T} \sum_{j=1}^{n(t)} x_{ij}(t) = 1, \ i = 1, 2, \ldots, M$$

where T is the latest time to assign weapons, V_j is the value of target j, q_{ij} is the probability of survival for target j if engaged by weapon i, and $x_{ij}(t)$ indicates whether weapon i is assigned to target j, or not. The goal is to minimize the value of the targets over all time steps, $t \in [0, T]$ [65]. The problem with the DWTA-SD problem is that, in general, it cannot be solved because the optimal decision cannot be made until after the final time T [65]. To overcome this, the form of the DWTA-SD problem is combined with the DWTA-SLS method, where the cost of stage t is calculated considering the already known cost of stage $t-1$ [65]. The formulation in the DWTA-SD problem can be adapted equation 4.7, and the solution can be generated based on the shoot-look-shoot method.

For the sensor tasking problem, the objects to be observed are considered the *targets* and the grid fields available are the *weapons*. Unlike the usual WTA problem, only one weapon (grid field) may be employed at each stage ($M = 1$), there is no limitation on the number of stages to which a weapon is assigned, and multiple targets are generally sought to be engaged by each weapon. Because the grid fields cannot be employed all at once, the static WTA problem is not applicable.

Instead of minimizing the value of the surviving targets, the sensor tasking problem is seeking to maximize the value of the objects observed. Thus, the WTA algo-

rithm assigns a viewing direction a_f by considering the probability of observing the most objects over the next k stages of the observation window:

$$h^* = \max\Big(\sum_{j=f}^{f+k-1} w_h \cdot \sum_{i=1}^{n} \mu_i \cdot p_d(h, \widetilde{\mathbf{Z}}_i(j)) \cdot d(h, \widetilde{\mathbf{Z}}_i(j), \mathbf{P} z_i(j)) \Big) \tag{4.19}$$

$$\text{where: } w_h = \begin{cases} 0 & \text{if used in a previous stage} \\ 1 & \text{otherwise} \end{cases} \tag{4.20}$$

where, f is the current observation being considered. Because $M = 1$ at every stage, the product term in equation 4.18 is not present in equation 4.19. Additionally, the q_{ij} term is replaced by the probability that the object is observed in the given grid field ($p_d(h, \widetilde{\mathbf{Z}}_i(j)) \cdot d(h, \widetilde{\mathbf{Z}}_i(j), \mathbf{P} z_i(j))$). The method assumes that the same grid field should not be chosen more than once during the k stages considered, which is enforced by the multiplier w_h as defined in equation 4.20; this also limits the number of combinations that must be compared to find the maximum in equation 4.19. w_h does not provide a true cost of waiting, but instead provides a means for assessing the benefit of order in which the grid fields may be assigned.

By considering multiple steps, the optimizer is attempting to find a solution that is regionally optimal over multiple steps. It requires propagating the object states and covariances to k observation times in order to provide all of the necessary information for equation 4.19. The optimal grid field, h^*, is only determined for the time step, f, which is the first stage in equation 4.19. That grid field is assigned as the viewing direction, a_f, and the μ_i values are updated for the objects seen. The grid fields that are assessed for the other stages in equation 4.19 are only used to identify h^*; when the optimizer moves to time step $f + 1$, the information from the previous step is ignored and all new combinations are considered.

This WTA algorithm is similar to the greedy algorithm in that it assigns the viewing directions one at a time, however it uses a regional (k stage) optimal instead of a local optimal. Increasing the number of stages used increases the computational load

of the WTA algorithm, as the combinatorial complexity increases with each added stage. The number of possible solutions that must be checked is the combination C_k^H, where H represents the number of grid fields available to the sensor and k is the number of stages; to decrease the number of combinations, the number of grid fields is limited to those where RSOs are likely to be observed, e.g.:

$$\sum_{i=1}^{n} \mu_i \cdot p_d \cdot d(h, \widetilde{\mathbf{Z}}_i, \mathbf{P}z_i) > 2.2204 \times 10^{-16}$$

where 2.2204×10^{-16} is the machine precision of the computer used for this research. Even with this limitation, there are hundreds of grid fields available at every observation time that are considered by the WTA algorithm. Theoretically, if $k = m_g$ the algorithm would find the true optimal solution; however, this would be the same as performing a brute force analysis of all possible sets of viewing directions, which is not possible from a computational standpoint.

The WTA algorithm attempts to generate a better solution than the greedy algorithm, by considering the effect of the current choice on the next few observation times. Despite the use of more information in the viewing direction choices, the algorithm still assumes the problem is convex, and is not expected to generate the optimal solution. Additionally, the consideration of additional observation times increases the computational load over that of the greedy algorithm.

4.4 Reinforcement Learning

Reinforcement Learning (RL) is a class of solution techniques that are used to generate optimal solutions to a given problem; RL is often used in the Machine Learning and Artificial Intelligence communities [62, 66, 67]. RL methods employ agents to find an optimal solution in a "dynamic environment" by learning, through feedback, which actions to take at a given state [62, 66, 68]. According to Mariano and Morales (2001), RL "approximates dynamic programming" (pg. 324), though Lilith and Doğançay state that dynamic programming assumes the environment is

perfectly known, while RL does not require knowledge of the environment [66,68]. As RL agents make choices in the environment, they receive positive feedback for good decisions and negative feedback for bad decisions.

The environment in RL problems is also sometimes called the state space for the problem. At every time t, the agent is at some state s, and must choose one action a from a set of possible actions [62, 66, 67]. Each state action pair has an associated transition probability and reward, which determine how likely the agent is to choose that action from that state and the value to the solution of the choice, respectively. As the agent interacts with the environment, testing each state action pair infinitely many times, the environment model is learned leading to the optimal solution [62,66].

There are two types of RL problems: Model-based and Model-free [62]. Model-based RL problems use historical, or training data to learn the environment first, and then solve future policies based on this learned information [62]. Model-free RL techniques require the agents to generate the solution in parallel with learning the environment [62]. Because of the nature of the sensor tasking problem, where effects such as non-linear motion, maneuvers, changes to RSO priorities, and weather can change the solution space for each observation interval, learning from past data is not guaranteed to provide good solutions. Thus, two model-free RL techniques are investigated in this work. The first is the Distributed Q-Learning algorithm, which is an extension of the Q-Learning algorithm [67, 68]. The second is the Ant Colony Optimization, which is based the behavior of biological ants [69, 70]; Ant Colony Optimization is a method of Swarm Intelligence (related to Machine Learning), but it shares characteristics with traditional model-free Reinforcement Learning [67].

4.4.1 Ant Colony Optimization (ACO)

The Ant Colony Optimization was first introduced by Dorigo et al. in 1991, and has continued to develop as a family of methods for solving a variety of optimization problems such as the Traveling Salesman Problem, the Quadratic Assignment Prob-

lem, and the Job-shop Scheduling Problem [70–72]. The technique is built on the observation that a colony of ants will, in the limit, find the optimal (shortest) path from their nest to a food source [67, 70–73].

The technique is based on sets of agents (ants) exploring the environment over which the cost function is optimized [71, 72]. Their exploration is guided by two principles: the amount of pheromone left by previous agents, τ, and a problem specific heuristic value η calculated for each action choice [62, 67, 72]. A simple example is shown in Figure 4.7, where agents are attempting to find the shortest path from A to E and back with no prior knowledge of the search space. Initially, the agents split evenly between the path from B to D, because there is no information provided about which path is better [71]. As the agents pass, they lay pheromone, which also evaporates over time. The shorter path will experience less evaporation before the next agents passes over that path, so the pheromone will grow faster than on the longer path [71].

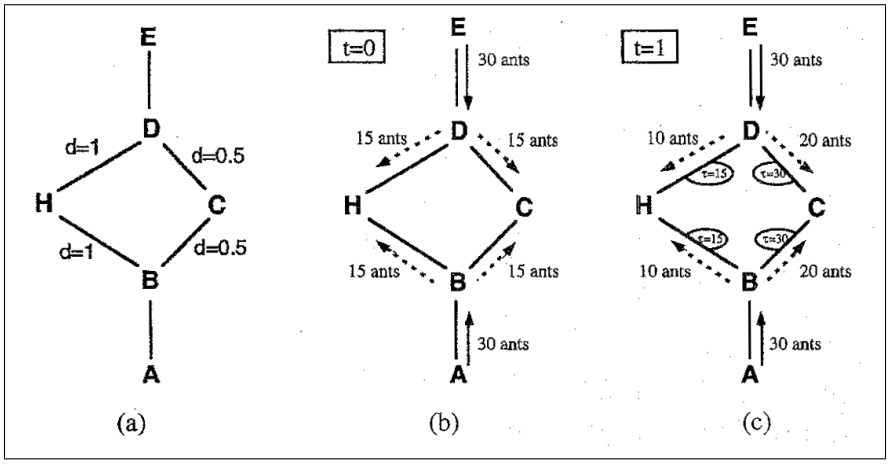


Figure 4.7. An example of the method employed with ACO: a) shows the distances on the bridge that the agents must travel, b) shows the initial traversing with the agents split evenly, c) shows the agents learning based on the pheromone left behind by previous agents. (From Ref. [71], Fig. 2, pg. 30).

Figure 4.7 is mainly concerned with the pheromone values, τ, that influence the ACO algorithm. To introduce the heuristic and develop the ACO problem, Dorigo et al. (1996) discuss solving the Travelling Salesman Problem (TSP); here the agents are attempting to determine the shortest closed loop path through n towns, where each town can be visited at most once [71]. Dorigo et al. (1996) describe η_{ij} as the inverse of the distance from town i to town j, which on its own leads the agents toward greedy decisions [71]. The pheromone, τ_{ij}, is left on each edge (path between towns) by each agent that traverses that edge, which encourages the agents to choose paths that other agents have chosen instead of only considering the shortest path [71]. Combining η_{ij} and τ_{ij}, Dorigo et al. (1996) introduce the transition probability at time t as:

$$p_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{k \in \text{allowed}_k} [\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}} & \text{if } j \in \text{allowed}_k \\ 0 & \text{otherwise} \end{cases} \tag{4.21}$$

where α and β are used to determine how much the transition probability is influenced by τ_{ij} and η_{ij}, respectively, and the allowed$_k$ are the towns to which agent k has not previously traveled [71].

In the TSP, the distances between the towns are static, while the pheromone values fluctuate. Dorigo et al. (1996) update the pheromones upon completion of a "cycle" according to [71]:

$$\tau_{ij}(t+n) = \rho \cdot \tau_{ij}(t) + \sum_{k=1} m \Delta \tau_{ij}^{k} \tag{4.22}$$

where n is the number of steps in a cycle (number of towns), m is the number of agents, and ρ represents how much the pheromone evaporates ($0 < \rho < 1$). If an edge is not selected regularly, ρ causes it to become undesirable by decreasing the intensity

of the pheromone on that edge. The amount of pheromone placed by each agent is given by:

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & \text{if the edge } (i,j) \text{ is used by agent k} \\ 0 & \text{otherwise} \end{cases} \tag{4.23}$$

where Q is a constant and L_k is the length of the complete tour for agent k during that cycle [71]. An agent that chooses an edge (i, j), but ends up with a large L_k still places pheromone on the edge, but the result is less intense than another agent with a shorter L_k. The process is iterated until some stopping condition is met.

The definition of the sensor tasking problem in this work can be constructed to represent a path through the weighted viewing directions A. As discussed above, the viewing directions obey a temporal discretization (m_g) and a spatial discretization (h). Unlike the TSP, where agents only visited a town once per cycle, the agents in the sensor tasking problem may choose to move to any grid field, h, at observation time, f. In doing so, each agent generates a *path* through the observation window, where at each f they choose an h to define the viewing direction, a_f. Figure 4.8 shows an example of possible paths that an agent might build through a few steps of the optimization problem; the grids used are from the LME measurement space.

The value of observing previously unobserved RSOs is used as the heuristic, $\eta_{f,h}$, in this work; if the agent choices are solely based on the heuristic, they would tend toward greedy solutions. Each agent carries their own μ_i for the objects ($\mu_i(k)$), and updates the value based on when an object is observed. Therefore, the heuristic value, $\eta_{f,h}$, is different for each agent and is not calculated until the agent is ready to move to step f. The value for $\eta_{f,h}(k)$ is given by:

$$\eta_{f,h}(k) = \sum_{i=1}^{n} \mu_i(k) \cdot p_d \cdot d \tag{4.24}$$

where the right hand side is the inner most summation of equation 4.7, as calculated for a given agent. $\eta_{f,h}(k)$ is determined for every grid field, h, at the observation time,

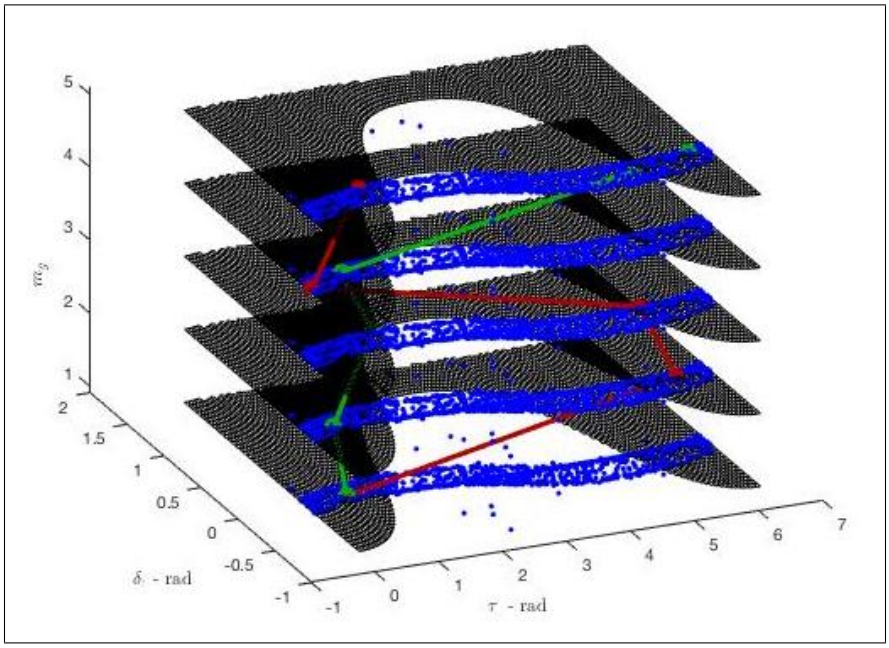


Figure 4.8. The green and red lines represent possible paths through a subset of the m_g viewing directions.

f. After calculating all of the heuristic values for step f, the values are combined with the pheromone values.

In contrast to the heuristic values, the pheromone values are not unique to the agents, instead representing a common desirability of each grid field at each observation time. The result is a pheromone matrix, $T_{m_g \times H}$, where the rows represent the observation times and the columns represent the grid fields available to the sensor. Each agent accesses the values from the same pheromone matrix as they generate their solutions.

When a set of agents completes an iteration of the problem, the k solutions generated are used to update the elements of the pheromone matrix, $\tau_{f,h}$, based on the value obtained by each agent that chooses grid field $h \in H$ at observation time

$f \in m_g$. The total pheromone deposited on each viewing direction is determined by:

$$\Delta\tau_{f,h} = \sum_{k=1}^{s(f,h)} \frac{V_k}{Q} \tag{4.25}$$

where, $s(f,h)$ is the number of agents that chose that viewing direction, V_k is the expected value gained by the agent k for the complete solution, and Q is a constant scaling factor; for this work Q is the total number of RSOs in the catalog, though it could also be the number of objects that are visible during the night [71]. The evaporation and addition of pheromone is performed after solutions are generated. The evolution of the pheromone is given by:

$$\tau_{f,h}(t+1) = \rho \cdot \tau_{f,h}(t) + \Delta\tau_{f,h} \tag{4.26}$$

where t indicates pheromone values from the previous iteration. The pheromone tells the agents how valuable each grid field was at a given time step, based on previous solutions in which it was chosen at that time step. While the $\eta_{f,h}(k)$ is unique to each agent, the $\tau_{f,h}$ values are common across all agents.

The agents make choices of which viewing direction to choose by considering the combination of pheromone and heuristic values in the weighting factors, $w_{f,h}(k)$ [72]:

$$w_{f,h}(k) = \frac{[\tau_{f,h}]^\alpha \cdot [\eta_{f,h}(k)]^\beta}{\sum_{j=1}^{H} [\tau_{f,j}]^\alpha \cdot [\eta_{f,j}(k)]^\beta} \tag{4.27}$$

where H is the number of grid fields contained in the grid, k represents the current agent, and α and β control the importance of $\tau_{f,h}$ and $\eta_{f,h}(k)$, respectively [72]. The agents, use the $w_{f,h}(k)$ to probabilistically choose to which grid field they will move; this allows the agents to solve the sensor tasking optimization problem as formulated in equation 4.7, by cooperatively assessing possible paths simultaneously.

The grids used in this work contain large numbers of grid fields, through which the agents must traverse. In order to determine an optimal path, the author chose to use a large number of agents to search the large number of the possible grid fields at

each observation time. A drawback to having too many agents is the possibility of a large number choosing the same viewing direction during an early iteration, causing the $\tau_{f,h}$ value to increase quickly; the result would be a very large $w_{f,h}(k)$ value that is almost solely driven by $\tau_{f,h}$. In order to avoid this, and to encourage diversity throughout the iterations, the number of agents that can choose a given grid field at a given observation time is restricted by K_{lim}; K_{lim} is a free tuning parameter, which the author has set to ten for this work. When the number of agents that have chosen a grid field reaches K_{lim}, the remaining agents are not allowed to select that grid field. The result is the adjusted weighting function:

$$w_{f,h}(k) = \begin{cases} \frac{[\tau_{f,h}]^{\alpha} \cdot [\eta_{f,h}(k)]^{\beta}}{\sum_{j=1}^{\mathbf{L}_f} [\tau_{f,j}]^{\alpha} \cdot [\eta_{f,j}(k)]^{\beta}} & \text{if } h \in \mathbf{L}_f \\ 0 & \text{otherwise} \end{cases} \tag{4.28}$$

where $\mathbf{L}_f$ is a list of grid fields the agent may choose at the current observation time. Initially, every grid field is in $\mathbf{L}_f$ and they are only removed if the maximum number of agents (K_{lim}) choose that grid field at that time.

Once a set of agents has passed through the entire set of observation times, the total value for A, the optimization cost function equation 4.7, is determined for each agent. For the first set of agents, the path generated by the agent that finds the highest value of A is saved. For the following iterations, the highest value of A is compared with the previous best value; if the new value is higher, that value, and the path associated with it, replace the previously saved value and path. Additionally, it is possible to have more than one path that achieves the maximum value for A; when this happens, all the paths are saved until a higher value for the cost function A is found.

ACO provides a complex method for optimizing the solution of the sensor tasking problem. The heuristic encourages choices that directly focus on the desired outcome, similar to the greedy optimizer, while the pheromone encourages choices that look at the whole solution in order to try and make better decisions. There are various tuning parameters (including α, β, Q, K_{lim}, and the number of agents) that are

adjustable, and impact the solutions in different ways. Determining the optimal values for the tuning parameters requires significant time before the actual solutions can be generated; Appendix D describes the tuning steps taken for this work. If properly tuned, the ACO solutions are expected to produce higher A values than the greedy and WTA algorithms.

4.4.2 Distributed Q-Learning (DQL)

Distributed Q-Learning is a reinforcement learning algorithm that was introduced by Mariano and Morales, which extends the single agent Q-Learning method [62, 67, 68]. In Q-Learning, the learning agent uses the value function $Q(s, a)$ to choose the next action to execute in the policy, assuming an optimal policy will be followed after the current action is chosen [67]. For each state $s \in S$, there are available actions $a \in A_s$, each of which has an estimated value $Q(s, a)$ [68]. Generally, the agent will choose the action with the largest $Q(s, a)$ (exploitation), while occasionally choosing another action instead (exploration) [67, 68].

The method for determining when to choose the action with the maximum $Q(s, a)$, and when to choose some other action is not prescribed by the Q-Learning algorithm. A common method found in the literature, and used in this work, is the ϵ-greedy strategy given by:

$$\pi(s) = \begin{cases} \arg\max_a Q(s, a) & \text{if } q \leq \epsilon \\ a_{random} & \text{otherwise} \end{cases} \tag{4.29}$$

where $\pi(s)$ is the choice of action (a), and q is a random value from the uniform distribution over $[0, 1]$ [68]. Equation 4.29 allows the agent to continually assess the solution, by occasionally $(0 < \epsilon \ll 1)$ choosing actions that are not the current maximum $Q(s, a)$ to see if they lead to better rewards. This possibility for deviation and variation gives the method the flexibility to handle non-convex problems. However,

the less conservative epsilon is set, the more flexibility the method allows, at the cost of slower convergence and increased run times [67, 68].

The algorithm assumes that every choice after the one currently being made will be optimal, and uses that assumption as part of the function for updating the value function. Thus, the $Q(s, a)$ values are updated by:

$$Q(s, a) \Leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)] \tag{4.30}$$

where s is the current state, a is the chosen action, r is the reward, s' is the new state, a' are the actions available at s', γ is a discount factor ($0 < \gamma \leq 1$) for future actions, and α is the learning rate (often $\alpha = 1$) [67]. The $Q(s', a')$ are the values for the next state-action pairs; i.e., the choice to be made after the current action is selected.

The discount, γ, and learning rate, α, are parameters that are freely tuned in order to generate the optimal solution. The discount parameter ensures that future actions are not weighted to heavily with respect to the current $Q(s, a)$ values. The learning rate can be set as static, or adjusted during the optimization, in order to keep the $Q(s, a)$ values from growing to quickly [62]. The reward, r, may be a value calculated based on the policy generated (problem specific), or some predetermined value based on the result of an action chosen (user defined); in this work r is problem specific, representing the value of the cost function for the generated solution.

In order to converge to the optimal solution, Kaelbling et al. (1996) state that Q-Learning requires that "each action is executed in each state an infinite number of times on an infinite run and α is decayed appropriately" (pg. 254) [62]. With a single agent, it can take a long time to try all the possibilities enough to consider them executed an infinite number of times; this can lead to slow convergence, especially when the state and/or action spaces are very large [62].

DQL extends Q-Learning, from a single agent to a set of agents in order to provide more exploration of the dynamic environment during each iteration [68]. It is similar to ACO in that there are multiple agents attempting to solve the problem during each iteration [68]. DQL differs from ACO in that it does not use a heuristic to influence

the policy choices, instead relying only on the information from previous iterations (the $Q(s,a)$ values), and random exploration to determine an optimal policy; the $Q(s,a)$ values are similar to the pheromone matrix used by ACO.

The DQL agents do not interact with the $Q(s,a)$ values directly. Instead, at the beginning of an iteration, each agent is given a copy of the current $Q(s,a)$ values, $Q_{c^i}(s,a)$, where c^i represents the copy carried by agent i; each agent solves the problem independently using their local copy [68]. The agents solve the problem and update their own $Q_{c^i}(s,a)$ values according to:

$$Q_{c^i}(s,a) \Leftarrow Q_{c^i}(s,a) + \alpha[\gamma \max_{a'} Q_{c^i}(s',a') - Q_{c^i}(s,a)] \tag{4.31}$$

which is similar to equation 4.30, but does not include the reward [68]. Equation 4.29 is still used by each agent, with $Q(s,a)$ replaced by $Q_{c^i}(s,a)$, to determine if it will choose the action with the best estimate, or explore other actions.

Once all the agents complete an iteration, the $Q(s,a)$ values are updated using equation 4.30, based on the state-action pairs chosen in the best policy for that iteration [68]. Consequently, the rewards, r, are only calculated at the end of each iteration, based on the best policy. The use of many agents allows the algorithm to search more state action pairs in each iteration [68].

For the sensor tasking problem, the state s is the set of viewing directions chosen up to the previous observation time ($\{a_1, a_2, \ldots, a_{f-1}\}$). The action a is the choice of which grid field, h, to choose for the current viewing direction a_f. While every grid field could possibly be assigned, the more actions available at each step, the longer it takes for the solution space to be searched. Additionally, it can be seen in Figure 4.4, that some of the grid fields have a higher likelihood of objects being observed than other. In order to reduce the number of actions allowed at each step, any grid field where the probability of observing RSOs is zero is removed from the action list for that observation step; for this work, any value that does not exceed the machine precision of the computer ($\approx 2.2204 \times 10^{-16}$) is considered equal to zero.

For the DQL algorithm, the set of weighted viewing directions is also represented by the set of state-action pairs generated by the chosen solution; in other words, each value of a_f corresponds to a $Q(s,a)$ value. After the solution is generated, the reward that corresponds to the solution is generated in order to update the $Q(s,a)$ values that were in the chosen solution. For this work, the reward r_f is given by:

$$r_f = \frac{a_f}{N} \tag{4.32}$$

where a_f is the weighted viewing direction value at observation time f, and N is the number of RSOs in the catalog. The $Q(s,a)$ values that were chosen in the current iteration, are updated using the rewards from equation 4.32 in equation 4.30.

Similar to the ACO method, each agent is solving the sensor tasking problem during each iteration. They use their copy of the value functions, $Q_{c^i}(s,a)$, to choose which action to take (which grid field, h, to choose) at the next observation time. With each choice, the $Q_{c^i}(s,a)$ values are updated according to equation 4.31. Additionally, each agent records the number of newly observed RSOs observed by the action; at the end of the iteration, the agent that determines the best solution uses these values to generate the rewards that will be applied in equation 4.30.

The literature does not provide a specific method for setting the initial $Q(s,a)$ values. If they are set to a constant value, the agents begin from no understanding of the optimal policy, and must generate the policy by balancing exploration of new state-action pairs with exploitation of previously explored *good* state-action pairs [67, 68]. The disadvantage is that the time to determine the optimal solution can be very long.

To speed up the convergence of the method, this work enhances some of the $Q(s,a)$ values based on the sub-optimal greedy solution. All of the state-actions pairs are initialized to one, with the associated state action pairs from the greedy solution increased to two. The agents then proceed to exploit those values or explore the other actions in the usual way.

As the number of iterations becomes large, it is possible for a few of the $Q(s,a)$ values at each observation time to become large, while others remain small; this can occur when a given $Q(s,a)$ value is repeatedly chosen in the best solutions. To encourage the exploration of other state-action pairs, this work uses a weighted sampling in the ϵ-greedy strategy to choose the random action. This changes equation 4.29 to:

$$\pi(s) = \begin{cases} \arg\max_a Q(s,a) & \text{if } q \leq \epsilon \\ (w_h \cdot a)_{random} & \text{otherwise} \end{cases} \tag{4.33}$$

where the weight w_h is given by:

$$w_h = \frac{1}{Q_{c^i}(s,a)} \tag{4.34}$$

Every action, including the estimated best action, is allowed to be chosen by the ϵ-greedy search strategy. The addition of the weighting value helps to ensure that as one state-action pair is executed a large number of times, it becomes less likely to be chosen when an agent explores the search space.

DQL provides a flexible method for searching the possible solutions for the sensor tasking problem. There is no problem specific heuristic, with the solution being dependent on fully searching the solution space and the rewards received. With proper tuning, DQL is expected to produce better solutions than the greedy and WTA algorithms; the steps used for tuning the DQL algorithm in this work are described in Appendix D. The difference in the solutions of the DQL and ACO algorithms is dependent on the tuning for each, as well as the potential benefit of the heuristic for the ACO solution.

4.5 Summary

The sensor tasking problem used in this work is a simplification of the formulation in Frueh et al. (2018); it is a maximization problem, where the value to be maximized

is the sum of the weighted viewing directions. In order to make the problem tractable, the size of the time steps between the observations were fixed, and the FOR was divided into fixed pointing directions for each sensor. This simplified formulation will be used in the simulations in the next chapter to compare the optimizers introduced in this section.

The convexity of the sensor tasking problem was also investigated in this chapter. A counter example showed that the problem is not expected to be convex. The four optimizers that are used in the next chapter were also introduced above. The greedy and WTA optimizers assume a convex problem is being solved, while the ACO and DQL optimizers do not. The ACO and DQL optimizers are expected to generate more effective sensor tasking solutions.

5. OPTIMIZER COMPARISONS AND RESULTS

The optimizers introduced in Chapter 4 are used to generate solutions for the formulation in equation 4.7. Six simulations are run with each optimizer generating solutions. The simulations are split into three scenarios which are run for the ground-based and the space-based sensor: a) absolute knowledge with $p_d = 1$, b) uncertain state estimates with $p_d = 1$, and c) uncertain state estimates with a calculated $0 < p_d < 1$. The first two scenarios assume perfect observability, such that, if the object falls within the chosen viewing direction, it will be observed regardless of the illumination geometry. In the first scenario, only the mean states are considered by the optimizers, treating the problem as though the states can be perfectly known; thus, the value of each viewing direction is the number of objects within the FOV. This is an idealization, but it provides a best possible case against which the other scenarios can be compared. The second scenario incorporates the uncertainty of the object states, such that the value of a viewing direction is the combined CDF values of the target objects as given by equation 4.9. The final scenario introduces the calculation of the probability of detection (equation 2.42) as a scaling on the object CDF values; therefore, the value of the viewing directions includes both the likelihood that an object is in the FOV and the likelihood it will be observed if it is in the FOV.

In all of the scenarios discussed in this section, immediate feedback is assumed. In other words, after a viewing direction choice is made, the optimization algorithm is informed of which objects are observed. For the ACO optimizer, each agent is made aware of the objects that its choices observe, but not the objects other agents observe. The DQL algorithm does not use the feedback in the midst of each iteration, only considering the number of objects observed by each agent at the end in order to determine the best solution and thus where to apply the rewards in equation 4.30.

5.1 Comparison Metrics

The optimizers are compared based on the effectiveness of the solutions and the efficiency of employment. The effectiveness is measured in terms of the number of unique objects the solutions are able to observe; the solutions are the sets of viewing directions chosen by the optimizers. Additionally, the sum of the values for the chosen viewing directions are compared; these are the values of the viewing directions, as defined in equation 4.10, from which the optimization algorithms choose the grid field to assign. These values indicate the expected value observed (combined CDF values), as a comparison to the actual number of objects observed; in the absolute knowledge scenarios the two values will be the same.

The efficiency of the optimizers are measured in two ways. First, the time required by the optimizer to generate the solution is employed as a direct comparison. This compares the time required for the optimizers to generate solutions. To measure this, the *tic* and *toc* stopwatch functions within MATLAB® are employed. Only the optimization process is measured, while the time required for the calculations required by every optimizer (e.g., propagating the objects and calculating the CDF values) are not measured. All of the scenarios are run on the Space Information Dynamics group's Linux server [1], using MATLAB® [2].

The second measure of efficiency is the time to set up the optimizers. The greedy algorithm requires no set up, while the set up for the WTA algorithm is a simple choice of the number of stages, k that are considered when choosing each viewing direction; the choice of k does have a direct effect on the time required for the optimizer to calculate solutions. The ACO and DQL algorithms both require tuning that is specific to each of the scenarios. Setting up these algorithms requires setting values, running the simulation, noting the results, making adjustments, and running again; a description of the process used in this work is found in Appendix D. Addi-

[1] Red Hat Enterprise Linux Workstation release 6.10, Copyright 2011-2017 Red Hat, Inc.; x86_64, 32 Intel® Xeon® E5-2630 v3 (2.40GHz)

[2] R2017a Update 3, 64-bit (Copyright 1984-2017 The MathWorks, Inc.)

tionally, the nature of the sensor tasking problem, as implemented in this work, does not guarantee that the tuning for a given scenario will be relevant in another scenario (e.g. a different observation night, or observer location). Based on this second measure of efficiency, the classical optimizers employed in this work are significantly more efficient than the learning optimizers.

5.2 Catalog and Sensors

The catalog of target objects used in the simulations contains 1275 RSOs from the USSTRATCOM TLE catalog for September 24, 2018[3]. The objects chosen are bounded in semi-major axis ($37,000 \leq a \leq 45,000$ km), while all other orbital elements are unrestricted. The resulting catalog contains RSOs in the GEO region, as well as some in geosynchronous transfer orbits.

The initial states are generated by propagating the TLEs with an SGP4 propagator to four hours before the start of the observation window and extracting the position and velocity vectors; the observation windows for the ground-based and space-based sensors are different, so the exact times are listed in the appropriate sections. The position and velocity vectors from SGP4 provide the first moment of the PDF for each object. The TLEs do not provide information on the uncertainty of the orbit state, so the second moment of the PDF for each object must be initialized [27]. For this work each object is given a diagonal covariance with position uncertainty $\sigma_x = \sigma_y = \sigma_z = 50$ km and velocity uncertainty $\sigma_{\dot{x}} = \sigma_{\dot{y}} = \sigma_{\dot{z}} = 0.01$ km/s; as the objects are propagated, the PDFs will evolve and influence multiple grid fields. The true object states are found by taking a sample from each of the intial PDFs; these true states are propagated along with the mean states, and are used to determine which objects are seen at each observation. Propagation is done in an Extended Kalman Filter framework, as discussed in Section 3.1.2, in order to simulate a real-world sensor tasking scenario for catalog maintenance.

[3]Catalog is from www.Space-Track.org for day 267, 2018.

The greedy and WTA algorithms operate in the same manner for all of the simulations. The WTA algorithm uses $k = 3$ stages for analysis of the best viewing directions. For the ACO and DQL optimizers, tuning is required to account for differences in the sensors and to deal with the different cases; the exact parameters used will be presented in each section. Additionally, in all cases, as soon as the viewing direction is assigned, the optimizer determines which objects are observed and changes their μ_i values to zero, so they will not be considered in the choice of future viewing directions; in the case of the ACO optimizer, this is done for every agent as each carries their own solution, and the individual agents are not guaranteed to see the same objects.

5.2.1 Ground-based Sensor

The ground-based sensor used for the following simulation is the Purdue Optical Ground Station (POGS), located in New Mexico (Lat: 33° N, Long: 106° W)[4]. POGS has a $3 \times 3°$ FOV, which is used to define the grid of pointing direction in the LME coordinate frame, as discussed in section 4.1.1. To determine which objects fall within the chosen viewing directions, the observer position is rotated to the inertial frame by the Local Mean Sidereal Time, θ_{lmst}, in order to generate the pointing vector to the true, or expected, position of the object. The following ground-based sensor simulations are based on observations beginning at 0132 hours 32 secs (UT) on 26 September 2018, and lasting for 10 hours and 40 mins (300 observations between astronomical sunset and sunrise)[5].

[4]POGS position vector: $\bar{R}_1^{ECEF} \approx [-1434, -5161, 3466]^T$ km

[5]Local time for the window began at 1932 hours

Absolute Knowledge

Absolute knowledge assumes the mean state is the true state of the object. Thus, the value of the viewing direction is based on the number of previously unobserved objects that fall within the grid field, as determined by:

$$\sum_{f=1}^{m_g} \cdot \Big(\sum_{i=1}^{n} \mu_i \cdot d(\tau_{g,f}, \delta_{g,f}, \tau_i, \delta_i, t_{g,f}) \Big)$$

$$\begin{aligned} \tau_{g,f} - \tfrac{1}{2}\text{FOV} - \tau_i &\leq 0 \\ -\tau_{g,f} - \tfrac{1}{2}\text{FOV} + \tau_i &\leq 0 \\ \delta_{g,f} - \tfrac{1}{2}\text{FOV} - \delta_i &\leq 0 \\ -\delta_{g,f} - \tfrac{1}{2}\text{FOV} + \delta_i &\leq 0 \end{aligned}$$

where the grid field ($h_{g,f}$) is represented by the angles that define its center ($\tau_{g,f}, \delta_{g,f}$), the object is located by the measurement angles for its mean state (τ_i, δ_i), and μ_i indicates if the object has been observed or not. These conditions are the same as equations 4.2-4.5, but in the LME coordinate frame used in this work.

The ACO algorithm performs eight iterations, which is user defined value [71]; the intent is to allow enough iterations for the pheromone to adequately develop without causing the computation times to become very long. Stopping conditions, such as stagnation, were found to be difficult to effectively implement in the sensor tasking problem [71]. It is found to be common for one iteration to produce solutions that are inferior to a previous iteration, while another subsequent iteration improves on the previous best solution.

This author found nothing in the literature about how to choose the number of agents to use in the ACO algorithm. In order to allow for a large number of possible combinations by the agents, the number of agents allowed is based on the total number of non-empty grid fields at the first observation time (196), multiplied by the number

of agents allowed to choose a given viewing direction ($K_{lim} = 10$, as discussed in section 4.4.1), for a total of 1,960 agents.

The pheromone evaporation constant is set to $\rho = 1/4$, while the pheromone importance is set to $\alpha = 1.0$ and the heuristic importance is set to $\beta = 1.2$. All of these values were found through experimentation with this particular case of the problem (see Appendix D).

For the DQL algorithm, the number of iterations and agents were defined by the author. The number of agents was chosen to be similar to the number of agents in the ACO algorithm, and set to 2,000. The number of iterations was set to 500; this significantly higher number of iterations was used because the agents have no knowledge of the solution space, no heuristic to help guide decisions, and the method requires "each action is executed in each state an infinite number of times on an infinite run and α is decayed appropriately" (pg. 254) [62]. The large number of iterations is tenable for the DQL algorithm, because there are many fewer calculations required during the optimization than there are for the ACO or WTA algorithms.

The other tuning parameters for the DQL are the learning rate, discount parameter, and exploration rate. The learning rate is set as $\alpha = 0.9$, the discount parameter is set to $\gamma = 0.7$, and the exploration rate is set to $\epsilon = 0.1$. As with the ACO parameters, these were set through experimentation with this particular case of the problem (see Appendix D).

Table 5.1 contains the number of successfully observed objects for the solutions generated by each of the optimizers, as well as the time required to generate the solutions. The *Possible* row represents the number of objects that appear in the field of regard (above the minimum elevation) at some point during the observation period, thus making them theoretically observable; this allows for assessing how well each optimizer performs against a common benchmark and the percentages in the second column are based on this value. The compute time for the optimizers only refers to the time to generate the solutions; the propagation of the objects and determination

of the grid field values at every time are common to all optimization algorithms, so these are not included when comparing the computation times.

Table 5.1. Number of successfully observed objects and computation times for each optimizer assuming absolute knowledge of the object positions at all times.

	# of RSOs	% of RSOs	Compute Time
Possible	502	-	-
Greedy	499	99.4	4.8 secs
WTA	498	99.2	50 mins, 53.7 secs
ACO	501	99.8	3 hrs, 25 mins
DQL	501	99.8	2 hrs, 59 mins

Table 5.1 shows that none of the optimizers observe all 502 objects. However, it must be noted that observing all objects is not guaranteed to be possible, within the give time frame. While some objects are visible to the sensor throughout the observation interval, others may only be visible at the beginning or end of the night, limiting the opportunity to observe them. ACO and DQL perform the best, observing all but one object. The greedy solution observes only two objects less, while the WTA solution observes three less objects. The number and percentage of objects observed do not provide a noticeable delineation between the optimizers in this idealized scenario.

The computation times do provide significant differences between the optimizers. As expected, the greedy algorithm is computationally the most efficient, because it only needs to select the best value at each step. The WTA algorithm requires more computation time in order to assess multiple stages, generate all possible combinations of the viewing directions, and select the grid field. The DQL and ACO algorithms are the most computationally expensive as they both require large sets of agents to analyze the problem multiple times in order to generate solutions. In this scenario,

all of the computation times are shorter than the simulation time frame, and short enough to use for pre-planning a single night of observations.

As they are implemented in this work, the greedy and WTA algorithms each produce only one solution. The DQL and ACO algorithms are capable of producing multiple solutions that all achieve the same final value for the weighted viewing directions. In this scenario, the ACO algorithm generates four solutions, while the DQL algorithm only generates one solution; the fact that multiple solutions can be generated to achieve the same value, while not a proof, supports the non-convexity of the sensor tasking problem.

The number of RSOs observed expresses how well the complete solution does, while the growth rate of objects observed indicates how the solutions are doing throughout the observation interval. This is shown in Figure 5.1 where the optimizers are differentiated by color and the individual solutions by the line style. The greedy and WTA solutions are very similar, maintaining a steep ascent as long as possible and only reaching a plateau when they approach their maximum value. The four ACS solutions have a slower growth rate through the middle of the observation window, before passing the greedy and WTA solutions. The DQL_1 solution initially tracks with the ACS solutions, then approaches the greedy and WTA solutions, before finally increasing to its final value near the end of the simulation. The ACS and DQL solutions show that there are multiple unique solutions that can all achieve the same value.

In addition to the total number of RSOs, the percentage of visible objects, and the growth rates of the objects observed, the optimizers are compared based on the viewing direction chosen by each solution and the corresponding number of objects observed in those viewing directions. Figures 5.2-5.4 show all of the solutions generated by the optimization algorithms. The timing of the observations are represented by the color bar on the right side of each image; the dark blue are early in the observation interval and bright green are near the end of the interval. The size of the red $*$ approximates the value (number of new RSOs observed) of the chosen viewing

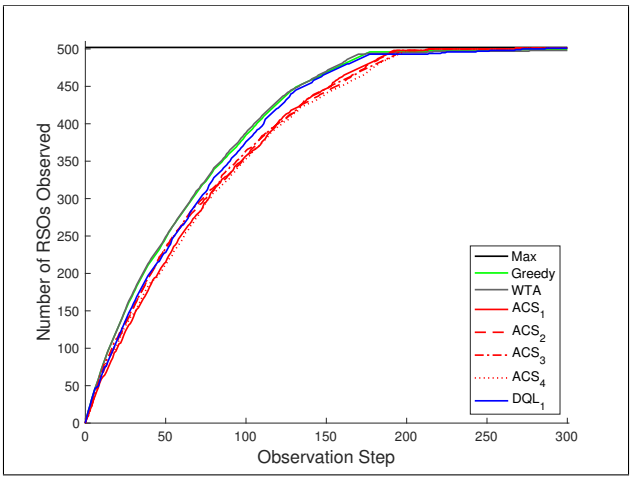
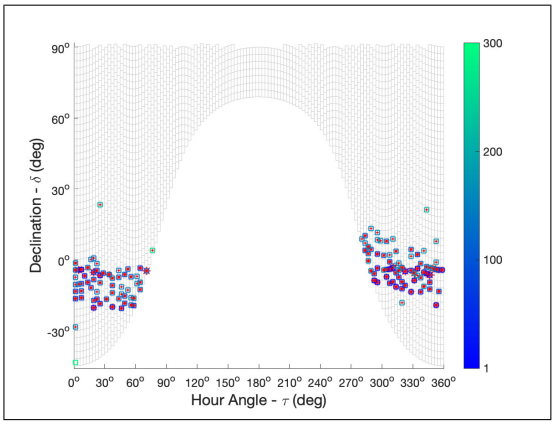


Figure 5.1. Growth rates for RSOs observed: the black line represents the maximum number of RSOs visible; algorithms are shown by color (greedy-green, WTA-grey, ACO-red, DQL-blue) and solutions by line style.

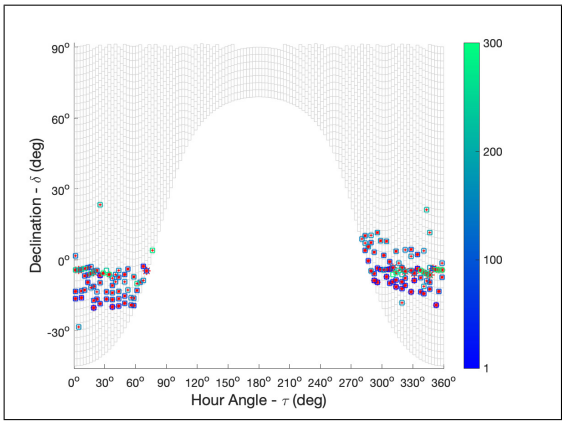
direction. In Figure 5.2, both the greedy and the WTA solutions present a tendency to observe larger numbers of objects in early viewing directions; the $*$ in these fields fill, or even extend beyond, the associated grid fields. By contrast, the later viewing directions appear to have only small dots in them.

Figure 5.3 represents the viewing directions selected for two of the ACO solutions; the other plots can be found in Appendix B. Again, the color of the grid field represents the timing within the observation interval and the size of the $*$ approximates the value of the viewing direction. While the early viewing directions are generally filled by the $*$, many of the later viewing directions are empty, indicating that no objects are observed. It is possible for the weights given by equation 4.28 to be zero in this scenario. When that happens, the agents are allowed to freely explore the viewing directions where objects are expected to be, regardless of whether those object have been previously observed.

Figure 5.4 represents the viewing directions chosen for the DQL solution. As with the ACO solutions, the agents may choose any grid field whether they contain objects needing to be observed or not; many of the choices did observe no new objects, as

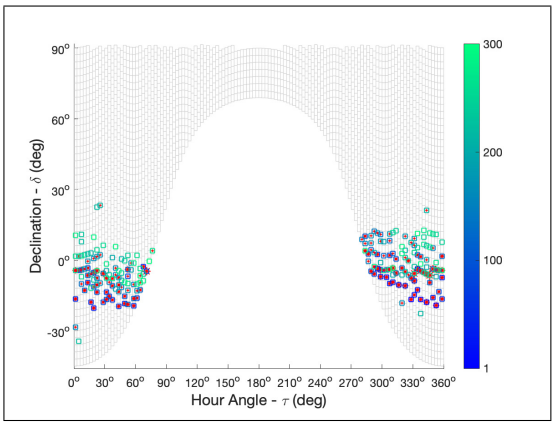


(a) Greedy.

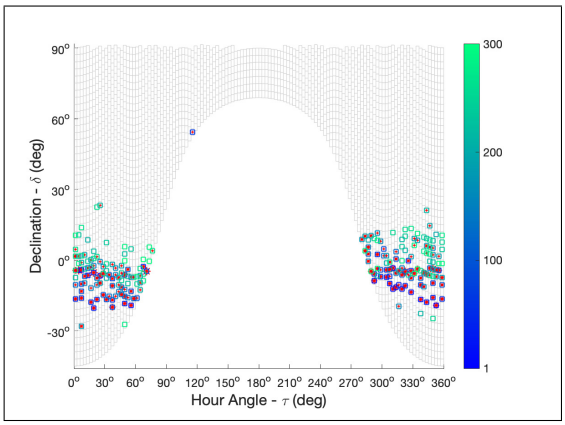


(b) WTA.

Figure 5.2. Grid fields, and associated values (# of RSOs), for the solutions generated by the greedy and WTA algorithms when the mean states are assumed to be the true states.



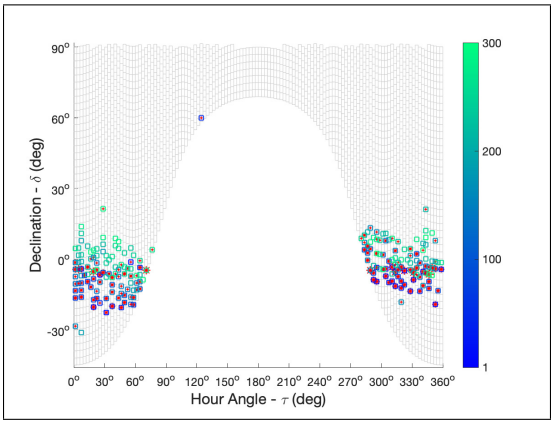
(a) ACO_1.



(b) ACO_4.

Figure 5.3. Grid fields, and associated values (# of RSOs), for two solutions generated by the ACO algorithm when the mean states are assumed to be the true states.

indicated by the viewing directions with no $*$ in them. However, the flexibility also allowed the solution to observe object at a very high declination; ACO_4 was the only other solution generated that selected a similar viewing direction.



(a) DQL_1.

Figure 5.4. Grid fields, and associated values (# of RSOs), for the solutions generated by the DQL algorithm when the mean states are assumed to be the true states.

Because the ACO and DQL algorithms are designed to allow the agents to explore the solution space, the final solutions tend to exhibit more unique grid fields chosen for the viewing directions than do the solutions for the greedy and the WTA algorithms. This can be seen in Figures 5.2-5.4, where there appear to be very few *empty* viewing directions for the greedy and WTA solutions, while there are a good number for the ACO and DQL solutions. Table 5.2 shows how many unique grid fields were assigned in each solution for the 300 viewing directions. Because the objects move with respect to the observer, it is expected that there will be grid fields chosen more than once during the observation interval. However, the higher numbers assigned in the ACO and DQL solutions, indicate that it may be possible to use those viewing directions for other purposes, such as survey.

This absolute knowledge case, with $p_d = 1$, provides an upper bound on how well the sensor tasking optimization can do. In this case, almost all of the observable objects are observed. However, this is not a realistic scenario. The following sections will add in the state uncertainties and the probability of detection, and show how those aspects affect the solutions generated.

Table 5.2. Number of unique grid fields used for the viewing directions in each of the solutions generated for the scenario where absolute knowledge is assumed.

	# of Unique $h_{g,f}$
Greedy	135
WTA	138
ACO_1	203
ACO_2	204
ACO_3	220
ACO_4	214
DQL_1	206

Uncertain State Estimates

In this simulation, the probability of detection will be kept as one, and the value of the viewing direction will be determined based on the probability that an object falls within the assigned grid field. In order to determine this probability, the first two moments of each object's Gaussian PDF are propagated in an EKF framework, under two-body dynamics. A CDF value is generated for every grid field by integrating each object's PDF across the corresponding FOV; the resulting values are represented by $d(h_{g,f}, \widetilde{\mathbf{Z}}_i, \mathbf{P}z_i)$ in equation 4.7. While the object PDFs are infinite, the CDF values become very small as the distance from the mean increases. Thus objects with a CDF value that does not exceed the computer's machine precision ($\approx 2.2204 \times 10^{-16}$) in a given grid field, are considered as having zero probability of being within that grid field.

The *true state* for each object is propagated to each observation time. While the selection of the viewing direction is solely based on the uncertainty in the object state, determining if an object is observed is based on the true state; the position from the true state and the observer position are used with equations 2.22-2.23 to determine

the angles to the object and whether it falls within the chosen grid field. Immediate feedback via the true states is assumed and, as with the previous scenario, μ_i is set to zero when an object is observed and otherwise remains one.

In this simulation the number of ACO agents is set based on half the number of grid fields at the first observation time that have combined CDF values greater than zero; using half the number of grid fields limits the impact to the computational load of the change to using the CDF values without significantly impacting the solutions generated. The resulting number grid fields is multiplied by $K_{lim} = 10$ to get a total of 1,120 agents. The pheromone importance ($\alpha = 1.0$) and evaporation constant ($\rho = 1/4$) are kept the same as the previous scenario, but the heuristic importance was increased to $\beta = 1.4$. The DQL parameters used are exactly the same as the previous scenario (500 iterations, 2000 agents, $\alpha = 0.9$, $\gamma = 0.7$, $\epsilon = 0.1$).

The total number of RSOs observed by the solutions for each optimizer are given in Table 5.3. Again, the top row represents the number of objects that fall within the FOR and thus, are theoretically observable. Overall, it has to be noted that a reduced number of successfully observed objects is expected since uncertainties have been introduced in the object positions and, as discussed in Chapter 3, linearization is present in both the object propagation and the transformation into the observation space.

Table 5.3. Number of successfully observed objects and computation times for each optimizer when uncertainty is included in the state estimates used by the optimizers.

	# of RSOs	% of RSOs	Compute Time
Possible	502	-	-
Greedy	459	91.4	6.9 secs
WTA	477	95.0	1 hr, 45 mins
ACO	488	97.2	2 hrs, 32 mins
DQL	474	94.4	4 hours, 52 mins

Again, the ACO algorithm performs best, in terms of the RSOs observed, followed by the WTA and DQL solutions. In this scenario, the DQL algorithm generates eight unique solutions, while all other algorithms only generate one solution. The greedy solution, which initializes the DQL function values, finds fifteen less objects than the DQL solutions; the greedy algorithm does not consider the temporal coupling of the sensor tasking problem, while each of the other optimizers allow for some temporal consideration.

Table 5.3 also shows the computation times for the algorithms. The greedy algorithm took about two seconds longer than in the previous scenario, but was still able to generate a solution in a few seconds. The WTA and DQL algorithms both showed notable increases in the computation times. Using the CDF values leads to an increase in the number of grid fields that must be considered at each observation time. For the WTA algorithm, this leads to a significant increase in the number of combinations that must be assessed at each step. In the case of the DQL algorithm, this leads to a larger value function which must be searched; the increase requires more calculations of equations 4.31 and 4.34, leading to longer times to generate the solutions.

The ACO algorithm must also deal with more computations due to the higher number of grid fields to choose from at each observation time. However, by decreasing the number of agents, the effect is minimized. Decreasing the number of agents in the DQL algorithm would have a similar impact on the computation time, but the impact on the effectiveness is not clear. Additionally, there is no guarantee that increasing the number of ACO agents, which will increase the computation time, will have a positive effect on the effectiveness of the solutions. This highlights the difficulty, and potential impact, of the tuning process for these learning algorithms.

Figure 5.5 shows the growth rates of the RSOs observed by each of the solutions. The greedy, WTA, and ACS solutions initially show steep growth in the number of objects observed. The WTA grows more quickly than the greedy and ACS, but both the greedy and the WTA begin showing less consistent growth around 100^{th} obser-

vation. The ACS maintains a more smooth growth, slowly approaching a levelling off around the 200$^{\text{th}}$ observation, with slow growth continuing until the end of the window. The DQL algorithm generated eight unique solutions, four of which are shown here, that display a slower growth rate than the other optimizers; however, the growth does not suffer the same levelling off that the other solutions experience and eventually approaches a similar value to the WTA solution.

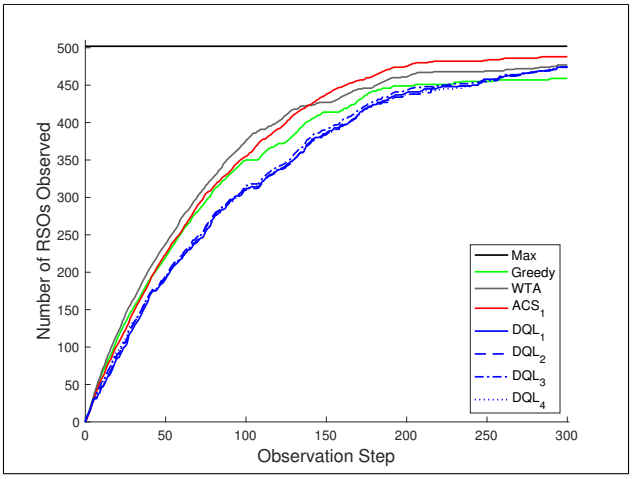


Figure 5.5. Growth rates for RSOs observed: the black line represents the maximum number of RSOs visible; algorithms are indicated by color (greedy-green, WTA-grey, ACO-red, DQL-blue) and solutions generated by the same algorithm are delineated by line style.

While maximizing the number of RSOs observed is the goal of the optimization, the decision variable is the weight of the grid fields chosen; the weights of the chosen grid fields, (from equation 4.9) can be summed over the entire night to provide a combined value, A_v, for each solution. In the previous scenario (Absolute Knowledge) the two values are the same, however, when the CDF values are used to generate the grid field weights, they are different. Table 5.4 presents the sum of the grid field weights for solutions from the optimizers; because DQL generated eight solutions, only the lowest (DQL_1) and highest (DQL_3) values are shown.

Table 5.4. The summation of the grid field values for each solution when uncertainty is included in the state estimates used by the optimizers; unlike the number of RSOs observed, there is no clear definition of a maximum possible value.

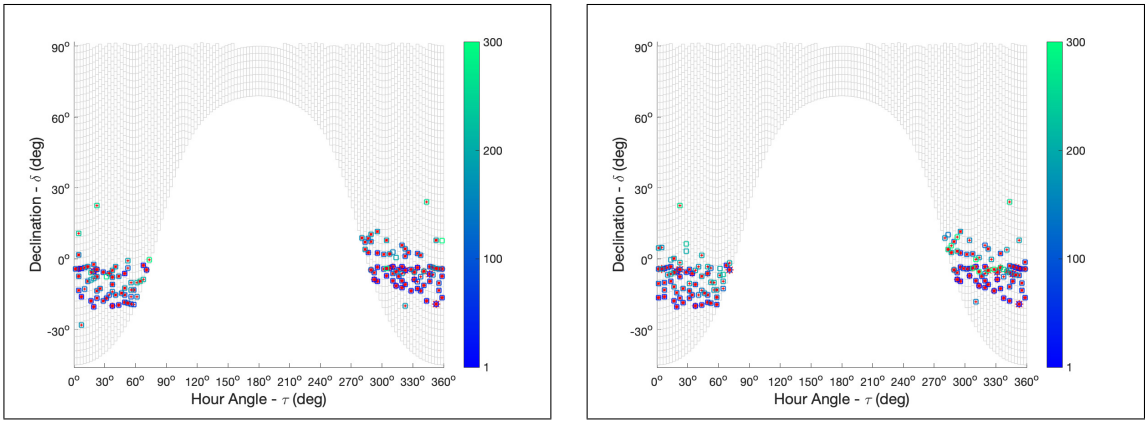
Optimizer	greedy	WTA	ACO	DQL_3	DQL_1
A_v	573.21	536.15	501.31	436.09	423.02

In Table 5.4, the values decrease as the optimizers consider more information. The greedy algorithm maximizes the value of the chosen grid field at every time; if that choice observes no RSOs, it is possible a similar weighting will be assigned for the next observation time as well. This leads to a very high value. The WTA algorithm, by attempting to maximize value over multiple steps, is able to assign viewing direction that may have lower weights than those assigned by greedy; over the entire observation window, these choices lead the WTA algorithm to generate a better solution than the greedy.

The ACO and DQL algorithms have lower total A_v values, while still generating better solutions then the classical optimizers. There are two reasons for this: a) the effect of learning good choices, and b) the ability of the agents to explore all possible choices. As the agents in these algorithms learn which choices are good, they provide the algorithms with a way to choose viewing directions that lead to high numbers of RSOs observed, even if the weights for those grid fields, from equation 4.9, are not the highest at that observation time. Additionally, the agents are able to search the solutions space, which leads to the opportunity to assign some viewing directions that have very small values (not likely to observe RSOs), while still generating good overall solutions; this can be seen by comparing the viewing directions assigned by each solution.

The viewing directions for the greedy and WTA solutions are presented in Figure 5.6, and those for the ACO_1, DQL_1, DQL_2 and DQL_3 solutions are presented in Figure

5.7. Comparing the images in Figure 5.6 to those in Figure 5.2, the distributions of the grid fields does not change significantly for the greedy and WTA solutions. However, in Figure 5.2(a), there is a grid field in the bottom left of the grid that is assigned. In the absolute knowledge case, there were observations where no new objects could be observed, and so all grid field weights were zero; in these instances, the bottom left most grid field was assigned (this occurs often toward the end of the observation window). When considering the uncertainty, there is no time when the grid field weights are all zero; this can be seen by an increase in the assignment of grid fields later in the observation window (upper third of the color bar) that are more aligned with those from earlier observation times. There are also slight differences in the viewing direction distributions for the WTA solutions, but they are not as noticeable.

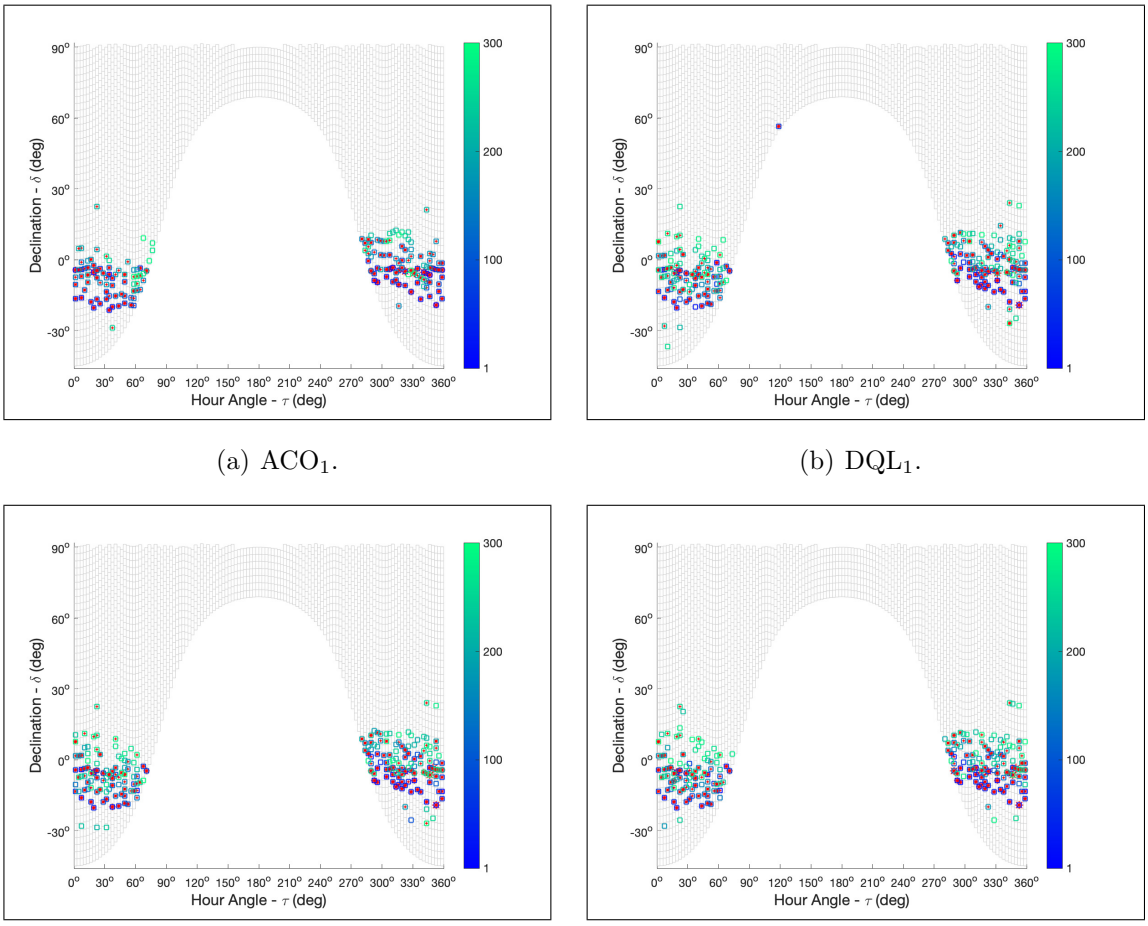


(a) Greedy.

(b) WTA.

Figure 5.6. Grid fields, and associated # of RSOs, for the solutions generated by the greedy and WTA algorithms when uncertainty is considered in the optimization.

The viewing directions assigned for the ACO and DQL solutions, in this scenario and the previous scenario, result in a decent spread of the chosen grid fields in τ and δ. This is again due to the learning of the agents and their opportunity to explore.



(a) ACO_1.

(b) DQL_1.

(c) DQL_2.

(d) DQL_3.

Figure 5.7. Grid fields, and associated # of RSOs, for the solutions generated by the ACO and DQL algorithms when uncertainty is considered in the optimization.

Comparing Figures 5.2-5.4 with Figures 5.6-5.7 does not indicate that there are significant changes in the number of unique grid fields selected. Table 5.5 shows that this is generally true by comparing the number of unique grid fields assigned by optimizer in the two scenarios; when the ACO and DQL algorithms generate multiple solutions, only the minimum and maximum number of unique grid fields are included. The ACO algorithm is the only one in which the number of unique grid fields changes considerably; while the number of unique grid fields decreases significantly,

the solution still outperforms the other solutions. This indicates that there is, at most, a weak relationship between the number of unique grid fields assigned and the effectiveness of the solution.

Table 5.5. The number of unique grid fields used for the viewing directions is compared for each algorithm in the scenarios with absolute knowledge and uncertain state estimate.

	# of Unique $h_{g,f}$	
	Uncertain States	Absolute Knowledge
Greedy	133	135
WTA	141	138
ACO	170	203, 220
DQL	200, 214	206

Introducing the uncertain state estimates reduces the number of objects observed, as expected. The ACO algorithm still generates the best solution, while the WTA and DQL algorithms both outperform the temporally ignorant greedy algorithm; plots related to the additional DQL solutions can be found in Appendix B.

Uncertain State Estimates, Non-Unity Probability of Detection

The final scenario for the ground-based introduces the probability of detection as a scaling factor for the CDF value of each object. These scaled CDF values are then used to generate the values for the viewing directions at every time. The probability of detection is also used to determine if the objects are seen. To decide if an object is observed, a random number is drawn from the uniform distribution, $u \in U(0,1)$, and an object is seen if $p_d \geq u$; this is an additional random process at work in the problem that was not seen in the previous scenarios.

The same number of agents are used by the ACO algorithm in this scenario as were used in the previous scenario; the number of agents is based on the non-zero

CDF values without the p_d scaling. Again, the pheromone importance ($\alpha = 1.0$) and evaporation constant ($\rho = 1/4$) are kept the same, while the heuristic importance is increased to $\beta = 1.6$. The DQL parameters remain the same as the previous scenarios (500 iterations, 2000 agents, $\alpha = 0.9$, $\gamma = 0.7$, $\epsilon = 0.1$).

Table 5.6 presents the total number of RSOs observed by the solutions for each optimizer. Introducing the p_d did not have significant impacts on the number of RSOs observed by each optimizers. The greedy solution actually increased the number of objects observed by including the p_d, though it is still the least effective solution. With the determination of what is actually observed being a random process, successive runs with different random number generator seeds are expected to generate different solutions and likely different numbers of RSOs observed by the solutions of the algorithms.

Table 5.6. Number of successfully observed objects and computation times for each optimizer when uncertainty and probability of detection are considered by the optimizers.

	# of RSOs	% of RSOs	Compute Time
Possible	502	-	-
Greedy	467	93.0	5.1 secs
WTA	470	93.6	1 hr, 37 mins
ACO	487	97.0	5 hrs, 2 mins
DQL	473	94.2	4 hrs, 44 mins

The computation times for the greedy, WTA, and DQL solutions are essentially unchanged from the previous scenario, while the computation time for the ACO solution almost doubled. As stated previously, these times only represent the actual solution generation, so the inclusion of the p_d calculation is not the reason for the increase. The author believes that the load on the server may have been a factor,

but this is not known for sure; additional running of the code may result in different computation times.

The growth rates of objects observed are shown in Figure 5.8; in this scenario, the DQL algorithm only generated one solution. The growth rates are very similar to those seen in the previous scenario, with the biggest difference being the greedy solution.

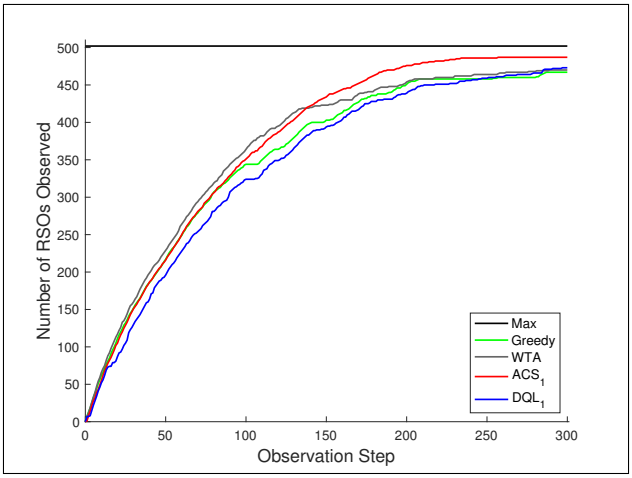


Figure 5.8. Growth rates for RSOs observed: the black line represents the maximum number of RSOs visible; algorithms are indicated by color (greedy-green, WTA-grey, ACO-red, DQL-blue) and solutions generated by the same algorithm are delineated by line style.

Table 5.7 shows the combined values of the solutions for each optimizer, A_v. The values for the greedy and WTA solutions have decreased from the previous scenario, while the values for the ACO and DQL solutions both increased. Including the p_d, scales the weights calculated for the grid fields, which accounts for the decrease in the greedy and WTA values; because $0 \leq p_d \leq 1$ the scaling is always a decrease in value. But these optimizers are still attempting to find maximum value, so their A_v values remain higher than the ACO and DQL values.

The increase in the A_v values for the ACO and DQL solutions is also tied to p_d. In the previous scenario, if an object was in the chosen grid field, it was observed. Now,

Table 5.7. The summation of the grid field values for each solution when uncertainty and p_d are used by the optimizers; again, there is no clear definition of a maximum possible value.

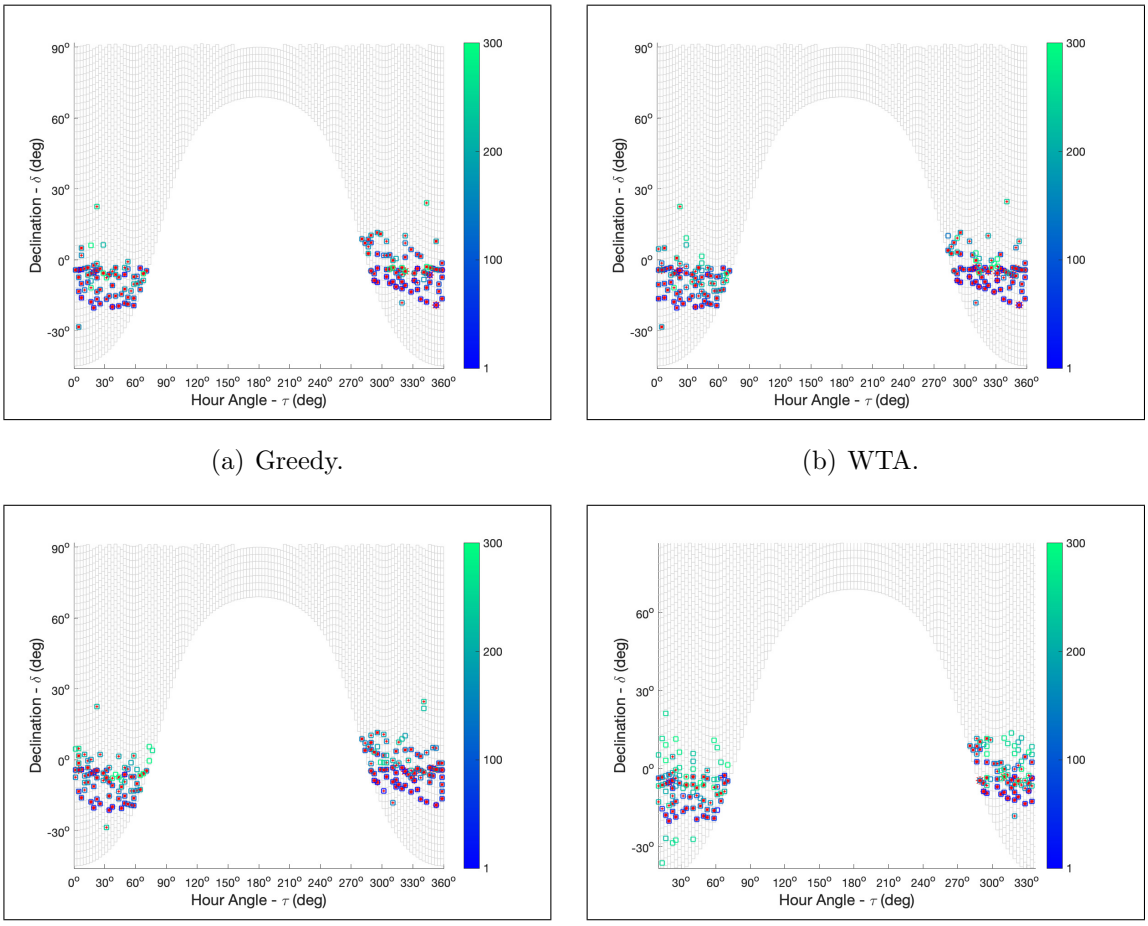
Optimizer	greedy	WTA	ACO	DQL
A_v	552.88	532.33	507.62	443.31

the object must be in the grid field, and it must pass a check on the p_d to determine if it is observed. In other words, the objects with low p_d values are less likely to lead the agents toward learning the grid fields in which those objects would be observed, because the feedback to the optimizer is likely to say that object was not observed. Thus, the agents will tend toward learning where objects that have higher p_d values are likely to be observed, which correlates to higher A_v values.

The objects observed by the chosen viewing directions are presented for each of the algorithms in Figure 5.9. The DQL shows the most diverse set of viewing directions, as well as the most change from the previous scenario. The ACO has a slightly higher diversity of viewing directions than the greedy and WTA, but each of these three are similar to the viewing directions observed previously.

Table 5.8, compares the number of unique grid fields assigned by each of the solutions in this scenario and the previous. The greedy and WTA solutions increased the number of unique grid fields selected, the ACO solution, again, decreased the number of unique grid fields selected, and the DQL selected almost the same number of unique grid fields as the previous scenario. Again, the p_d encourages the greedy, WTA, ACO toward choosing viewing directions with a better chance of observing objects, which causes the changes in these values. The flexibility of the DQL, which is tied to the ϵ-greedy search strategy, is less sensitive to the p_d in terms of the effect it has on the number of unique grid fields chosen.

In each scenario presented, the DQL solutions have chosen the highest number of unique grid fields, but they have not observed the highest number of RSOs. The



(a) Greedy.

(b) WTA.

(c) ACO_1.

(d) DQL_1.

Figure 5.9. Grid fields, and associated # of RSOs, for the solutions of each optimizer when uncertainty and p_d are considered in the optimization.

ACO solutions have also tended to select higher numbers of grid fields then the greedy and WTA solutions, and have observed the most RSOs. Increasing the amount of information for the optimizers to use in generating the solutions, generally led the greedy and WTA solutions toward increasing the number of unique grid fields; however, the effect on the number of RSOs observed by the greedy and WTA solutions was not consistent. Thus, one cannot infer a direct relationship between choosing more unique grid fields and observing more RSOs.

Table 5.8. The number of unique grid fields used for the viewing directions is compared for each algorithm in both scenarios with uncertain state estimates.

	# of Unique $h_{g,f}$	
	$p_d = 1$	Caculated p_d
Greedy	133	140
WTA	141	151
ACO	170	160
DQL	200, 214	215

Adding the probability of detection to the uncertain state estimates actually increased the efficacy of the greedy algorithm, while the other optimizers experienced slight decreases in effectiveness. Over all three scenarios, the ACO algorithm performed the best in terms of the number of objects observed; the WTA and DQL solutions did not perform as well as the ACO solutions, but they generally outperformed the greedy solutions. Generating the greedy solutions consistently used the least computation time, with the WTA computations being the second most efficient. The ACO and DQL computation times were the highest, though the effect of the number of agents on the ACO could be clearly seen between the first and second scenarios. The number of agents and iterations used in both the ACO and DQL algorithms was set by the author, and is not assumed to be optimal; additional tuning of both algorithms could lead to more effective and/or more efficient implementations.

The ground-based sensor scenarios tested each of the algorithms in one possible sensor tasking set-up. Changing the sensor to a space-based platform will generate significant changes for the tasking. The next sections will investigate the same general scenarios with the space-based sensor.

5.2.2 Space-based Sensor

The space-based sensor used in the following simulations is modelled as a sensor placed in the orbit of the International Space Station (ISS). The orbit for the sensor is taken from the same catalog as the target objects, with its orbit being defined by $[a = 6,784\text{km}, e = 3.96 \times 10^{-4}, i = 51.64°, \omega = 198.23°, \Omega = 246.64°, n = 97.6\text{mins}]$. The sensor is modeled as having the same $3 \times 3°$ FOV, and the same time between observations, as the ground-based sensor. The grid of pointing direction is generated in the SME coordinate frame, as discussed in section 4.1.1. Determination of which objects fall within the chosen viewing direction, is done by applying equations 2.27-2.29 with the observer to object pointing direction, and checking to see if the resulting angles fall in the selected viewing direction.

The following space-based sensor simulations are based on observations beginning at 1827 hours and 51 seconds UT, on 25 September 2018. The simulation lasts for 12 hours (338 observations). While the space-based sensor is not restricted to astronomical sunset and sunrise, it must ensure that observations are not taken toward the Sun. To do this, the Sun position is checked with regard to the grid, and any grid field whose center is within $\pm 4.5°$, in τ_s or δ_s, is not allowed to be selected.

The RSOs in the selected catalog have mean motions on the order of one revolution per day, while the sensor completes nearly 15 revolutions about Earth each day. As the space-based sensor orbits, its grid passes across all of the target objects, and thus is theoretically capable of seeing all 1275 objects. However, the time an object spends within the FOR, during each of the sensor's orbits, will be limited.

Absolute Knowledge

This scenario again assumes that the mean states of the RSOs are the true states, and the sensor will be able to see the objects used to select the chosen viewing direction. Because the sensor completes more than seven orbits during the observation

interval, the solutions will be compared with the full number of target objects to determine the efficacy of each optimizer.

The ACO algorithm still performs eight iterations, and the number of non-empty grid fields at the first observation (213) time is again multiplied by $K_{lim} = 10$, resulting in a total of 2130 agents. The pheromone evaporation constant and pheromone importance values are the same as those used in the ground-based scenarios, $\rho = 1/4$ and $\alpha = 1.0$ respectively. Through testing, the heuristic importance was increased to $\beta = 1.5$.

For the DQL algorithm, the number of agents is increased to 3000, over the 2000 used in the ground-based scenarios. The number of iterations is kept at 500. The learning rate and discount parameter are kept at $\alpha = 0.9$ and $\gamma = 0.7$, respectively, while the exploration rate is set to $\epsilon = 0.05$. Again, experimentation was performed in order to tune the optimizer, and achieve better results; the tuning is described in Appendix D.

Table 5.9 presents the number and percentage of successfully observed objects, and the computation time required for the solutions of each optimizer. Under the assumptions of this scenario, the greedy performs best, followed closely by WTA, ACO, and DQL, in that order. Because of the orbital motion of the sensor, the objects are moving across the grid much faster than in the ground-based sensor scenario; additionally, the objects become visible again on the following pass. This results in the greedy and WTA solutions being able to assign viewing directions that observe many RSOs, without as much impact of not being able to see other objects later. None of the optimizers observe all of the objects which could possibly be observed, however, because the space-based sensor is not restricted by the rise and set of the Sun, it is likely that the full number of objects could be observed with a longer scenario.

The computation time for the greedy solution is almost the same as it was for the ground-based sensor, while the WTA computation increase slightly; the number of non-empty grid fields is larger for the space-based sensor which directly effects the

Table 5.9. Number of successfully observed objects and computation times for each optimizer assuming absolute knowledge of the object positions at all times.

	# of RSOs	% of RSOs	Compute Time
Possible	1275	-	-
Greedy	1182	92.7	6.9 secs
WTA	1180	92.5	1 hr, 15 mins
ACO	1172	91.9	4 hrs, 30 mins
DQL	1170	91.8	5 hrs, 42 secs

computation time for the WTA optimizer. The longer scenario for the space-based sensor, the increased number of non-empty grid fields, and the increases in the agents lead to longer computation times for both ACO and DQL than those required for the ground-based sensor (Table 5.9).

Figure 5.10 shows the growth rates for each of the solutions which, even more than in the ground-based scenario, are very similar. The growth rates are also very smooth, showing almost no points where no new objects are observed. It is possible that the absolute knowledge scenario for a space-based sensor is convex (or convex extensible [57,59]), but this is not proven and the growth rates are expected to become less smooth when uncertainty is introduced in the following scenario.

Figure 5.11 shows all the solutions generated by the optimization algorithms. The timing and values of the observations are represented in the same manner as in the ground-based scenarios. As the sensor orbits, the objects cross the grid from left to right; this is the reason the greedy, WTA and DQL optimizers have more dense grid field choices on the left side of the grid. The objects do remain in the FOR for multiple observation times, so the observations are spread across the grid; however, the optimizers are attempting to observe new objects as they enter the FOR. The greedy and DQL solutions tend to assign later observations to grid fields on the left

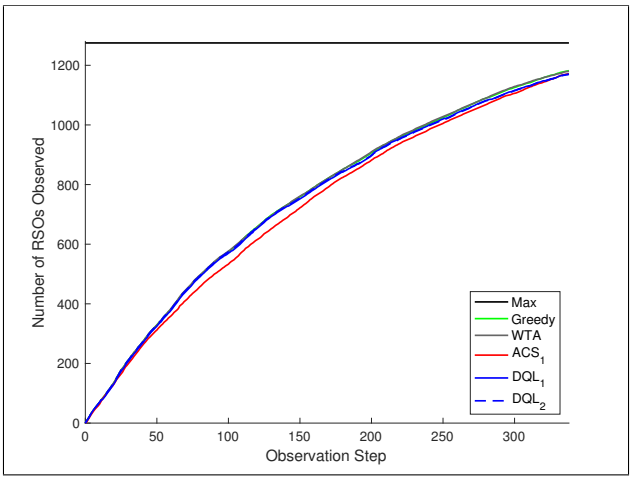
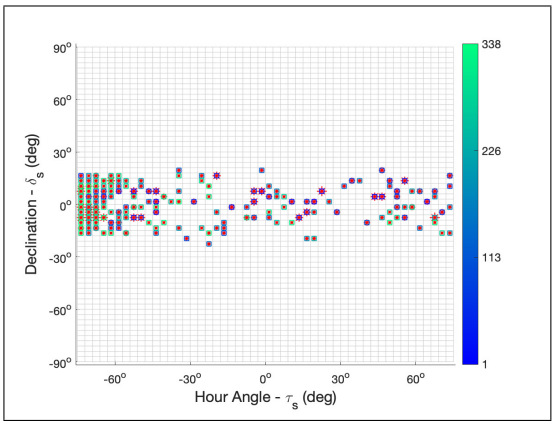


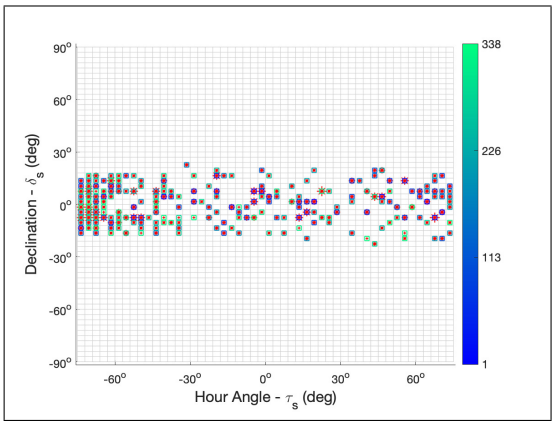
Figure 5.10. Growth rates for RSOs observed: the black line represents the maximum number of RSOs visible; algorithms are shown by color (greedy-green, WTA-grey, ACO-red, DQL-blue) and solutions by line style.

side of the grid, the left edges appearing almost all bright green. The WTA and ACO algorithms show better distributions of of the grid fields across the entire grid; for the WTA, this includes a wider distribution of the grid fields chosen for the later observation times than found in the greedy and DQL solutions.

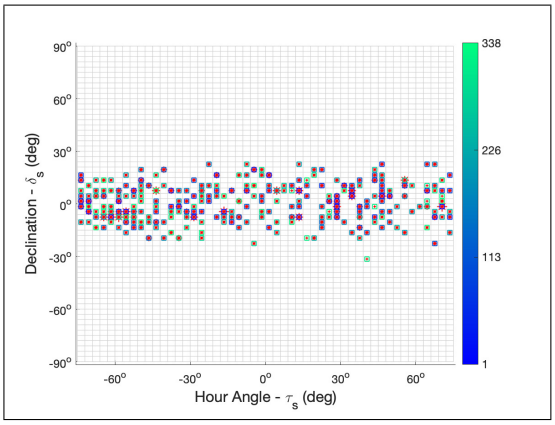
From Figure 5.11, it is expected, and found, that the WTA and ACO solutions will select more unique viewing directions than greedy and DQL solutions (greedy - 169, WTA - 218, ACO - 257, DQL_1 - 174, DQL_2 - 178). However, since the growth rates are all fairly smooth and similar, it cannot be inferred that the number of unique grid fields is directly tied to the number of objects found; the same was found for the ground-base sensor, so comparisons of the number of unique grid fields are not presented here, but can be found in Appendix B.



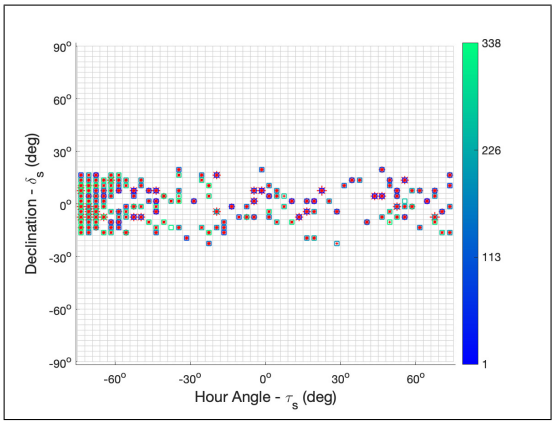
(a) Greedy.



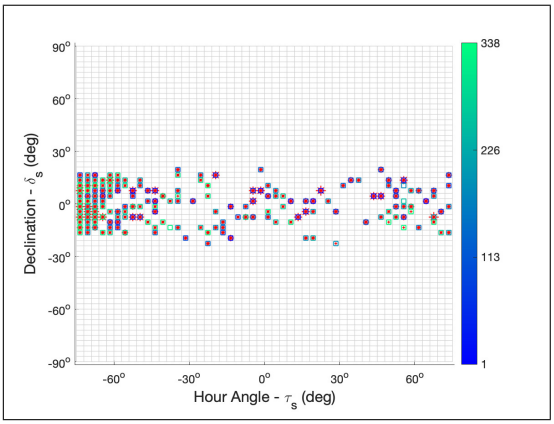
(b) WTA.



(c) ACO_1.



(d) DQL_1.



(e) DQL_2.

Figure 5.11. Grid fields, and associated values (# of RSOs), for the solutions generated by the each of the optimizers when the mean states are assumed to be the true states.

Table 5.10 shows a breakdown of how many of the selected viewing directions observed a given number of objects. Generally, the solutions assign similar numbers of viewing directions in terms of the number of objects observed, with two exceptions in the case of the ACO solution; these exceptions are highlighted in **bold**. The ACO solution chooses 21 more viewing directions that observe four objects, while selecting twelve less viewing directions that observe seven objects; the result is a zero sum difference ($(21 \cdot 4) - (12 \cdot 7) = 0$). The fact that the solutions generally assign similar numbers of viewing directions that observe the same number of objects is expected, because the number of objects observed and their growth rates are all very similar.

Table 5.10. The number of viewing directions assigned, in which a given number of RSOs were successfully observed.

	# of viewing directions which observed:									
# of objects	0	1	2	3	4	5	6	7	8	9
Greedy	0	24	98	88	40	30	30	19	6	3
WTA	1	26	93	90	30	30	30	19	6	3
ACO_1	0	16	100	85	**61**	29	30	**7**	7	3
DQL_1	4	27	95	84	40	29	29	21	6	3
DQL_2	4	26	94	86	40	30	30	19	6	3

The optimizers are shown to behave differently with the space-based sensor than with the ground-based sensor. In this scenario, the greedy optimizer generated the best solution, followed by WTA, ACO and DQL. The next step is to add in the uncertainty of the state estimates in order to see how the optimizers perform in that scenario, and whether the results follow what is seen in this case, or align more with what was observed in the ground-based scenarios.

Uncertain State Estimates

The probability of detection remains unity for this scenario, and the value of the viewing direction is determined based on the probability that an object falls within the assigned grid field; the probabilities are found by taking the CDF values of the transformed PDFs over all grid fields in the measurement space ($d(h_{g,f}, \widetilde{\mathbf{Z}}_i, \mathbf{P}z_i)$ in equation 4.7). The same cut-off for the CDF value that was introduced in the ground-based scenario is used here (computer machine precision $\approx 2.2204 \times 10^{-16}$).

Again, the *true state* for each object is propagated to each observation time, and the selection of the viewing direction is solely based on the uncertainty in the object state. Determining if an object is observed is based on the true state; the pointing direction based on the true state and the observer position are used with equations 2.28-2.29 to determine the pointing angles for an object's true state. Immediate feedback via the truth object is assumed and, μ_i is treated in the same manner as the ground-based scenario.

For this scenario, the number of agents for the DQL optimizer remains at 3,000, and the number of agents for the ACO is also set to 3,000; again, these values were chosen by the author, for this scenario, through experimentation. For the ACO optimizer, the heuristic importance was increased to $\beta = 1.8$, while the pheromone importance ($\alpha = 1.0$) and evaporation constant ($\rho = 1/4$) are kept constant. The other DQL parameters are also unchanged (500 iterations, $\alpha = 0.9$, $\gamma = 0.7$, $\epsilon = 0.05$).

The total number of RSOs observed by the solutions for each optimizer are given in Table 5.11. While the number of objects observed is expected to decrease, the percentage of visible objects observed dropped significantly more than it did in the ground-based scenarios (e.g., greedy: space-based 90.7% to 74.0%, ground-based 99.4% to 91.4%). The rate at which the objects pass through the grid contributes to this decrease; for example, if there are four objects that contribute sizable CDF values to the choice of a viewing direction, but one of those objects is not actually observed,

that object may not have enough influence to get a viewing direction selected at the following observation steps, in order for that object to be observed.

Table 5.11. Number of successfully observed objects, percentage of visible objects, and computation times for each optimizer when uncertainty is included in the state estimates used by the optimizers.

	# of RSOs	% of RSOs	Compute Time
Possible	1275	-	-
Greedy	944	74.0	6.9 secs
WTA	930	72.9	5 hrs, 54 mins
ACO	971	76.2	9 hrs, 24 mins
DQL	979	76.8	8 hrs, 23 mins

In this scenario, the DQL algorithm finds the most objects, observing eight more objects than the ACO solution. This indicates that the tuning is well suited for this scenario. It may be possible to improve the tuning for the ACO algorithm to generate a similarly effective solution. For repeatability in both the ACO and DQL algorithms, the random number generators used in all of these scenarios are seeded. If a random seed were used instead, the solutions for the ACO and DQL are expected to vary with each run, and thus the best solution may require running many instances of each algorithm; the same is true for the ground-based scenario.

As with the ground-based scenarios, the computation times for the WTA, ACO, and DQL algorithms all increased when the uncertainty was introduced. For the ACO, this is due to the additional agents and the increased number of calculations required for each of those agents (e.g. pheromone deposit and viewing direction weightings). For the WTA, using the CDF values increases the number of grid fields that must be considered for each viewing direction, which leads to a significantly higher number of combinations that must be investigated for each k stage analysis. The DQL agents also have more options for choosing actions when they explore the solution space, but

there are no additional calculations required, so the increase in computation time is not as large as those for the WTA and ACO.

Figure 5.12 shows the growth rates of the RSOs observed by each of the solutions. Again, the growth rates are very similar throughout the observation window, though the learning algorithms begin to separate from the classical algorithms after the 200^{th} observation. There are two unique solutions from the ACO algorithm in this scenario, though they only differ in three viewing directions assigned, resulting in the two growth rates being difficult to distinguish in Figure 5.12; a close up of the growth rates is shown in Appendix B.

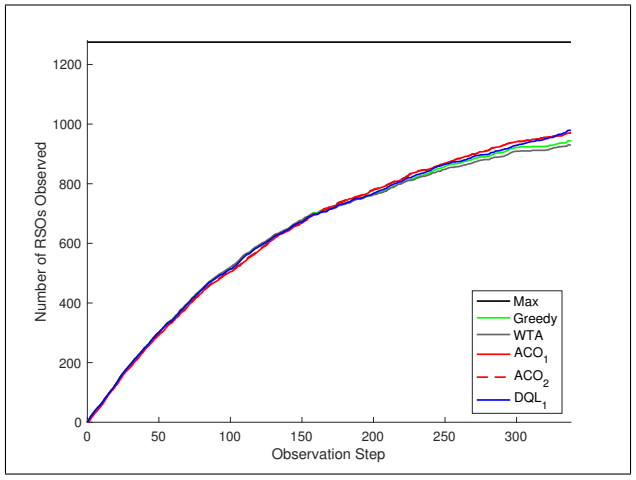


Figure 5.12. Growth rates for RSOs observed: the black line represents the maximum number of RSOs visible; algorithms are indicated by color (greedy-green, WTA-grey, ACO-red, DQL-blue) and solutions generated by the same algorithm are delineated by line style.

As discussed in the ground-based sensor scenarios, the values of the viewing directions assigned, A_v, are different than the number of RSOs observed. Table 5.12 shows the values for the space-based sensor solutions generated by each of the optimizers. The difference in the scenarios can be seen in the fact that the WTA produces a higher value than the greedy solution; in the ground-based scenarios, greedy always

produced the highest A_v value. The ACO and DQL solutions still result in lower A_v values than the greedy and WTA solutions.

Table 5.12. The summation of the grid field values for each solution when uncertainty is used by the optimizers; there is no clear definition of a maximum possible value.

Optimizer	greedy	WTA	ACO_1	ACO_2	DQL
A_v	996.83	998.35	949.69	952.42	919.15

When considering the uncertainty, the number of viewing directions that observed no objects or one object increased significantly over the absolute knowledge case; this can be seen by comparing Tables 5.10 and 5.13. The similarity of the greedy, WTA, and DQL growth rates can be seen by looking at the number of viewing directions assigned based on the number of RSOs observed; the number of viewing directions that observe six or more objects are identical, those that observed between three and five are similar, and the largest differences are in the number that observe two or less. The ACO solutions show more variability in the number of viewing directions, based on the number of objects observed, which accounts for the fact that those growth rates stand out slightly from the other solutions in Figure 5.12. The bottom half of Table 5.13 shows the distribution of the last 21 viewing directions based on the number of objects observed; the rise in the DQL growth at the end of the observation window is highlighted by the number of viewing directions that observe two, three, or four objects.

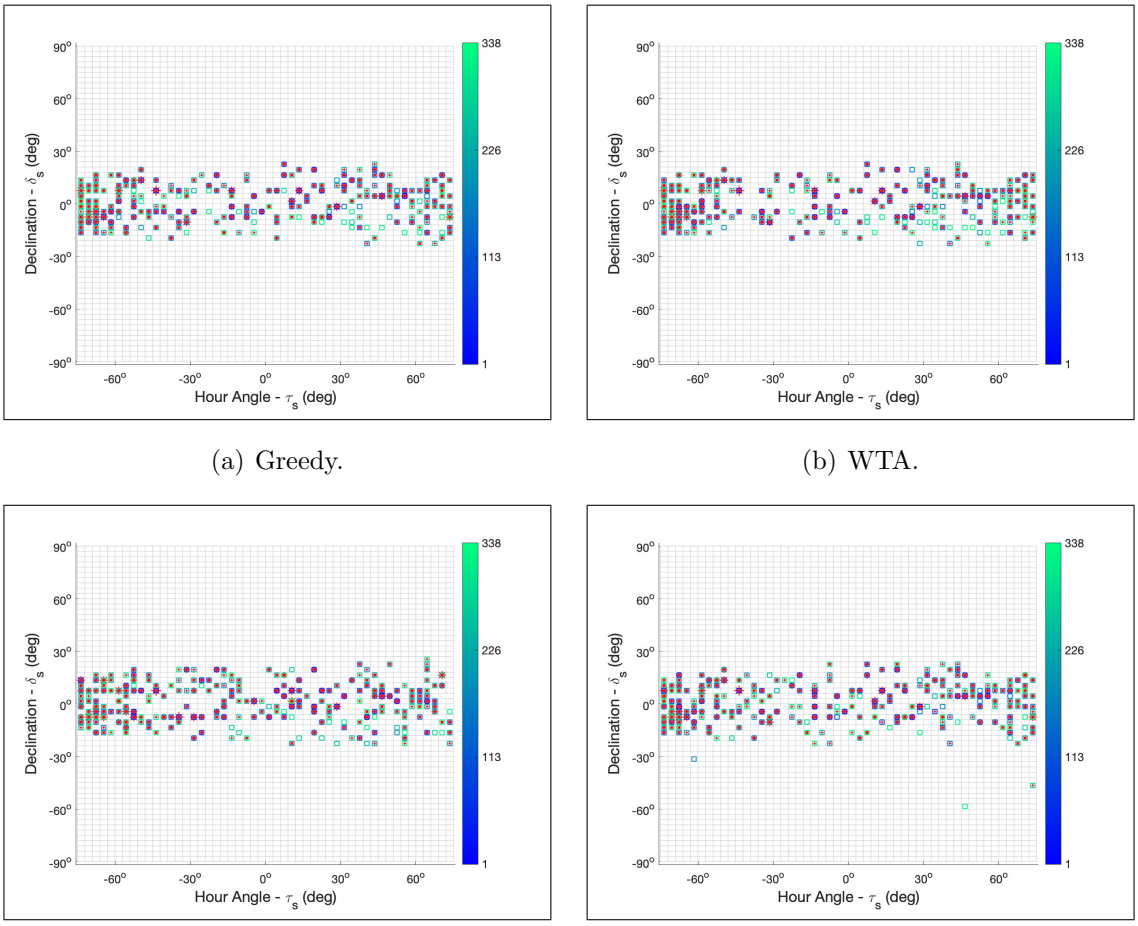
Figure 5.13 shows the distribution of viewing directions chosen by each solution. In Figure 5.13(a), like in Figure 5.11(a), the tendency for the greedy solution to assign later viewing directions more densely on the left edge of the grid is repeated. The greedy, WTA, and ACO_1 solutions assign all viewing directions between $\delta_s = \pm 30°$, which is generally where the mean states of the objects are located, and thus the highest probabilities for objects to be observed; the greedy and WTA algorithms are

Table 5.13. The number of viewing directions assigned, in which a given number of RSOs were successfully observed.

	# of viewing directions which observed:									
# of objects	0	1	2	3	4	5	6	7	8	9
Greedy	56	53	62	46	46	30	28	10	6	1
WTA	64	50	59	45	43	32	28	10	6	1
ACO_1	35	65	71	53	39	28	29	11	5	2
ACO_2	36	64	70	54	39	28	29	11	5	2
DQL_1	41	59	65	48	49	31	28	10	6	1
	The last 21 viewing directions									
Greedy	11	5	2	2	1	-	-	-	-	-
WTA	10	8	0	2	1	-	-	-	-	-
ACO_1	9	6	6	0	0	-	-	-	-	-
ACO_2	9	6	6	0	0	-	-	-	-	-
DQL_1	6	6	3	3	3	-	-	-	-	-

focused on maximization based on these probabilities, while the ACO algorithm is directly linked to them through the heuristic values. The ACO_2 solution is very similar to the ACO_1 solution, so it is not presented here, but can be found in Appendix B. The DQL solution assigns three viewing directions outside of $\delta_s = \pm 30°$; while only one of these viewing directions observes an object, this shows that the DQL algorithm is performing a wider search of the full solution space.

The introduction of the uncertainty in the object states, again reduced the total number of objects observed, though by a larger percentage than was found in the ground-based sensor case. In this scenario, the DQL and ACO solutions are able to observe many more objects than the classical algorithms; the ability of the agents to learn the solution space allows them to overcome the difficulty of only considering the probability of where the objects are located. Additionally, the greedy algorithm



(a) Greedy.

(b) WTA.

(c) ACO_1.

(d) DQL_1.

Figure 5.13. Grid fields, and associated # of RSOs, for the solutions generated by each of the optimizers when uncertainty is considered in the optimization.

is able to generate a better solution than the WTA algorithm; the limited temporal consideration (multiple stages) of the WTA algorithm does not appear to be of significant benefit to the solution. The final simulation will add the probability of detection calculation to the problem of the space-based sensor tasking.

Uncertain State Estimates, Non-Unity Probability of Detection

Again, probability of detection is introduced as a scaling factor for the CDF value of each object for the space-based sensor. As with the ground-based sensor, these scaled CDF values are then used to generate the values for the viewing directions at every time and the probability of detection is used to determine if the objects are seen, based on $p_d \geq u$ ($u \in U(0,1)$).

The same parameters are used by the ACO algorithm as were used in the previous scenario ($\beta = 1.8$, $\alpha = 1.0$, and $\rho = 1/4$). The DQL parameters also remain unchanged from the previous scenarios (500 iterations, 3000 agents, $\alpha = 0.9$, $\gamma = 0.7$, $\epsilon = 0.05$).

Table 5.14 presents the total number of RSOs observed by the solutions for each optimizer. Unlike the ground-based sensor, introducing the p_d resulted in an increase in the number of RSOs observed by each optimizer. Because the observer is orbiting, the object-Sun-observer geometry is changing rapidly during the observation window; this leads to times when the object p_d are very small, and others where they approach one. This added information facilitates better decisions by all of the optimizers.

Table 5.14. Number of successfully observed objects and computation times for each optimizer when uncertainty and probability of detection are considered by the optimizers.

	# of RSOs	% of RSOs	Compute Time
Possible	1275	-	-
Greedy	1021	80.1	15.3 secs
WTA	1004	78.7	4 hrs, 20 mins
ACO	1032	80.9	15 hrs, 10 mins
DQL	1019	79.9	7 hrs, 49 mins

The ACO sees the most significant increase in the computation time with the p_d included, almost seven hours of additional computation over the previous scenario.

The greedy algorithm increase by a matter of seconds, but still finished significantly faster than the other optimizers. Both the WTA and the DQL algorithms saw slight decreases in the computation times. The author believes the changes in the computation times for the WTA, ACO, and DQL algorithms may have been partly due to the overall load on the server used for running the simulations. However, the greedy and WTA algorithms still show better computation times than the ACO and DQL algorithms.

As with the previous space-based sensor scenarios, Figure 5.14 shows that the growth rates for the different solutions are very similar. Again, the growth rates only begin to diverge near the end of the observation window.

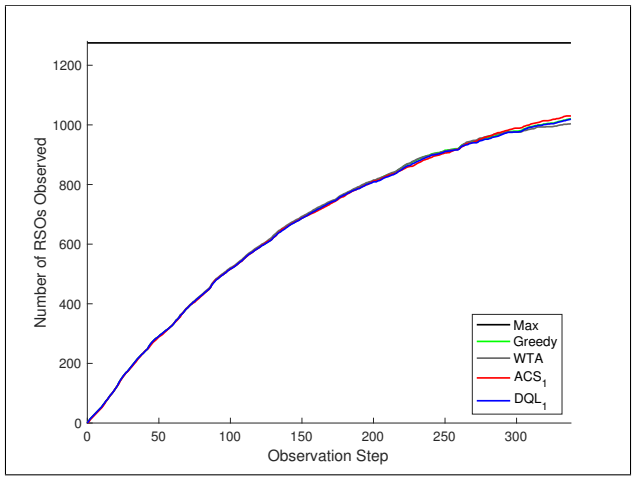


Figure 5.14. Growth rates for RSOs observed: the black line represents the maximum number of RSOs visible; algorithms are indicated by color (greedy-green, WTA-grey, ACO-red, DQL-blue).

Table 5.15 shows the combined values of the solutions for each optimizer, A_v, which have all increased from the previous scenario; because the number of RSOs observed increased, the A_v values should also increase. The DQL solutions experiences the largest increase, going from the lowest value in the previous scenario to the second highest value in this scenario. While the number of objects observed by the ACO algorithm is generally higher than the other algorithms, the A_v value is always lower;

the same is true for the DQL algorithm in all previous scenarios, but is not true in this scenario. The fact that the number of RSOs observed and the A_v values generally differ for the learning algorithms is due to the ability of the agents to learn more than just which grid fields have the highest probability at each observation time.

Table 5.15. The summation of the grid field values for each solution when uncertainty and p_d are used by the optimizers; again, there is no clear definition of a maximum possible value.

Optimizer	greedy	WTA	ACO	DQL
A_v	1044.38	1041.55	1021.60	1042.38

Including the p_d, forces the optimizers to choose viewing directions where the objects have better observation geometry; this means the early viewing directions may observe less objects than in the previous scenario. However, the objects are visible multiple times during the observation window, and the observation geometries will change. Figure 5.15 shows the effect of the p_d on the growth rate, and thus the final solution, for the greedy optimizer. When the p_d is considered, the growth rate is initially slower, but it remains steady longer than when it is not included. The growth rates of the solutions for the other optimizers, when comparing the two scenarios, show similar behavior; the plots for WTA, ACO, and DQL are found in Appendix B.

The number of objects observed by the chosen viewing directions are presented for each of the algorithms in Figure 5.16. Comparing the Figures 5.16(a)-5.16(c) to Figures 5.13(a)-5.13(c), the greedy, WTA, and ACO solutions show a similar distribution to the previous scenario. There is one notable difference, the number of grid fields chosen on the far left edge of the grid has decreased for each solution; the inclusion of p_d, which depends on the Sun-object-observer geometry, results in periods where the objects near the edges are less likely to be observed, so the algorithms tend to choose them less. Comparing Figure 5.16(d) with Figure 5.13(d), again shows that the DQL

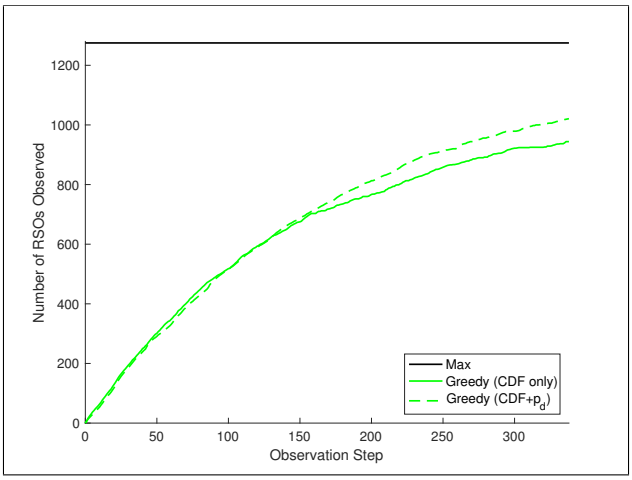
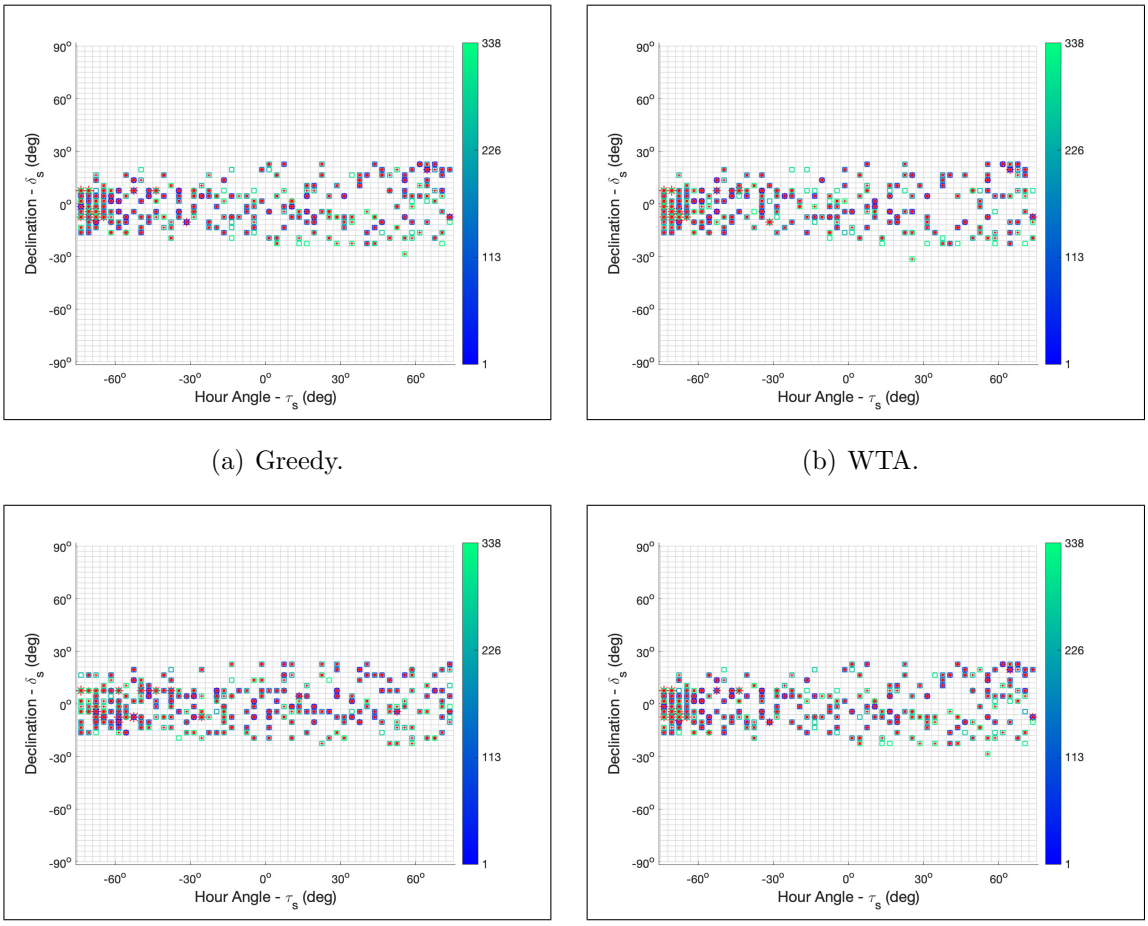


Figure 5.15. Comparing the greedy growth rates for RSOs observed: the solid line uses only the CDF values, the dashed line uses the CDF and p_d values.

selects grid fields in a wider range of δ_s than the other solutions. The same decrease in the number of grid fields selected on the left edge of the grid is also observed for the DQL solutions.

Table 5.16 again shows the number of viewing directions assigned by each solution broken down by how many RSOs were observed. The DQL solutions show very similar distributions to the previous scenario. The biggest changes for the greedy and ACO solutions appear in the number of viewing directions where zero and one objects were observed. For the WTA solution, the number of viewing directions where only two objects were observed decrease by 23 from the previous scenario, while the viewing directions where four and five were observed went up by 10 and 12, respectively. This shift in values matches with the increase in the total number of objects observed and with the performance of the WTA as shown in Figure 5.14.

In the case of the space-based sensor, a different optimizer produced the best solution in each case. The classical optimizers perform best in the absolute knowledge case, which is known to be the least realistic scenario. The DQL performs significantly better than ACO when considering the CDF; both learning algorithms outperformed



(a) Greedy.

(b) WTA.

(c) ACO_1.

(d) DQL_1.

Figure 5.16. Grid fields, and associated # of RSOs, for the solutions of each optimizer when uncertainty and p_d are considered in the optimization.

the classical optimizers. When the p_d was included, ACO produced the best solution followed by the greedy and DQL solutions.

5.3 Summary

The simulation scenarios in this section provide a comparison of the efficacy of each optimization algorithm for the sensor tasking problem. While the majority of

Table 5.16. The number of viewing directions assigned, in which a given number of RSOs were successfully observed.

	# of viewing directions which observed:										
# of objects	0	1	2	3	4	5	6	7	8	9	10
Greedy	25	54	65	71	55	31	19	13	2	2	1
WTA	39	46	62	65	57	29	23	12	2	2	1
ACO_1	26	47	70	79	41	35	22	12	2	3	1
DQL_1	27	52	65	70	56	32	18	13	2	2	1

SSA sensors are ground-based, the use of space-based sensors have advantages, so both types of sensors are examined. In each scenario, the algorithms are compared based on the number of RSOs observed by the solution, the summed weights of the viewing directions assigned, and the time required to generate the solutions; the effort required to tune the algorithms is also discussed. The algorithms were also compared based on the number of unique grid fields assigned, however, no clear advantage was found based on this comparison.

For the ground-based sensor, the ACO algorithm was consistently able to generate the solutions that observed the most RSOs. However, the WTA and DQL algorithms also performed well in each scenario. The greedy algorithm performed worse in the second scenario than in the third scenario, but was the least effective solution in each scenario.

For the space-based sensor, no single optimizer produced the best solution in all cases. The classical optimizers performed best for the absolute knowledge case, the DQL algorithm performed best when considering only the CDF values, and the ACO algorithm performed best when CDF and p_d are both considered. While the performance of the greedy and WTA algorithms decreased in the second and third scenarios, they were still able to produce good solutions. All of the optimizers performed better

when p_d was considered along with the CDF values, than they did when only the CDF values were considered.

In both the ground-based and space-based cases, the greedy algorithm was consistently, and significantly, more computationally efficient than the other algorithms. The computation times for WTA algorithm were longer than the greedy computation times, but shorter than the ACO and DQL algorithms. The ACO and DQL algorithms, while requiring longer computation times, also display more variability in the computation times than the classical optimizers. Additionally, the ACO and DQL algorithms require scenario specific tuning in order to generate the best solutions. The time and effort required to perform this tuning is hard to quantify, but it must be considered when comparing the overall efficiency of the algorithms.

6. PREDICTED MEASUREMENT PROBABILITY

The simulations in the previous chapter, as well as the sensor tasking and object filters discussed in Section 2.6, assume feedback is immediately available to the optimization algorithm, regarding which objects were observed [36, 49]. In a multi-sensor network, this implies real-time sharing of the information between all sensors and the optimization algorithm. It also requires that immediate updates of the object probability density functions (PDFs) are available with the obtained measurements of true objects as well as clutter detections [49].

This assumption of immediate feedback is an idealization. The observation processing for a single sensor approach, such as how many detections have been made in which locations on the CCD, is not instantaneous and may not be performed at the same location as the observer. The time delays required to transfer data between sensors, data processing locations, and the optimization algorithm mean there is some period where no updated catalog information is available, but the next observation may still be scheduled. In extreme cases, there may be no feedback during an entire observation period, such as an observation night, and all information is processed and communicated after that period. This creates the need for an efficient, optimized sensor tasking strategy for heterogeneous sensors without immediate measurement feedback.

The Predicted Measurement Probability (PMP) method is developed herein to handle situations where measurement feedback is not available. The development begins with a review of Bayes' rule in the problem with immediate feedback. Then the limitations for using Bayes' rule without feedback will be introduced. In both cases, the use of a Monte Carlo analysis is discussed to provide a point of comparison for the PMP method. Finally, the development of the PMP method will be explained and simulations using the method will be presented and discussed.

6.1 Bayes Rule - Immediate Feedback

When feedback is available, the application of Bayes' Rule:

$$p(x_{k+1}|Z_{k+1}) = \frac{p(z_{k+1}|x_{k+1})\ p(x_{k+1}|Z_k)}{p(z_{k+1}, Z_k)} \tag{6.1}$$

to a single-sensor, single-object estimation problem, with a state x_k and measurements $Z_k = (z_1, \ldots, z_k)$, leads to the optimal Bayes filter [74]. Such filters propagate the statistical moments to future times to represent the target distributions. Under linear dynamics, the Kalman Filter Prediction and update equations are derived from the Bayes' rule framework [49, 74, 75].

For single-sensor, multi-object systems, with $x_k(i)$ representing each object i, a measurement may be used to update a specific object, if association is possible, or to update the PDF of all objects simultaneously based on the probability that the measurement belongs to each object [74]. According to Mahler (2003), the Bayes filter for the multi-sensor, multi-object problem is too computationally complex to solve directly, requiring intelligent approximations and simplifications such as those used, for example, in the PHD filter [74, 76].

For the single-sensor, multi-object case, a simple example can be used to describe how Bayes' rule is used to update the PDFs. Figure 6.1 shows three objects with Gaussian PDFs, the Monte Carlo particles that represent the PDFs, and a FOV associated with an observation. The viewing direction is chosen to maximize the probability of seeing all three objects. Determining the viewing direction with the maximum probability is done in two ways: a) calculating the combined cumulative distribution function (CDF) values in the FOV for the three objects, or b) counting the number of Monte Carlo particles from the three objects, in the FOV.

The combined CDF value for the three objects is given by:

$$V = \arg\max_h \sum_{i=1}^{n} \int_h \frac{1}{\sqrt{(2\pi)^2 |\mathbf{P}_i|}} \exp\Big(- \frac{(\mathbf{X}_{i,h} - \widetilde{\mathbf{X}}_i)^T \mathbf{P}_i^{-1} (\mathbf{X}_{i,h} - \widetilde{\mathbf{X}}_i)}{2} \Big) \mathrm{d}h \tag{6.2}$$

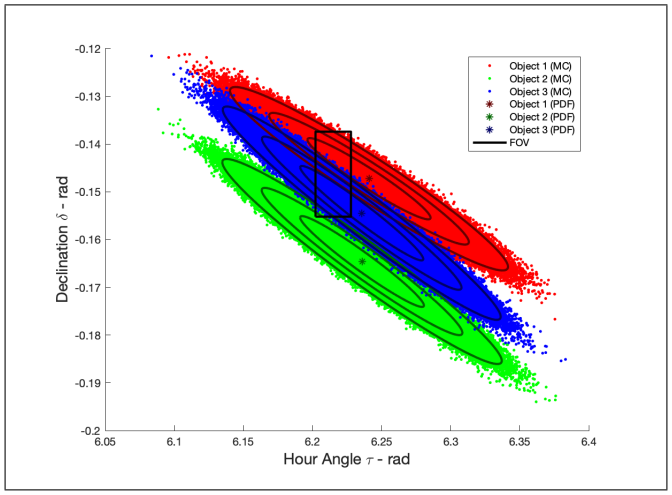


Figure 6.1. An example observation for three objects with Gaussian PDFs.

where the PDFs for each object ($\mathcal{N}(\widetilde{\mathbf{X}}_i, \mathbf{P}_i)$) are integrated over the rectangular FOV for each pointing direction h, and summed together to give a combined CDF value for each h. The h that results in the maximum V is chosen as the pointing direction for the given observation time; the states $\mathbf{X}_{i,h}$ represent the possible states of object i that fall within the FOV at pointing direction h. Assuming the measurement is processed in real time, the probabilities for each object could be updated based on the measurement taken [49, 74].

The second method eliminates the need for the integration of the PDF by using a Monte Carlo (MC) sampling of each object's distribution, where particles, p_j, represent the probabilistic states of the object. Each particle carries a weight, w_j, representing the probability of that particle being the true object. Because the weights represent the full PDF, it follows that $\sum_{j=1}^{N_p} w_j = 1$. These probabilities are updated based on Bayes' rule [49]. To get a true representation of an object's full PDF, very large numbers of particles are required [77]; often $N_p = 10^n$ particles is suggested, where n is the dimension of the state.

In Figure 6.1, the objects are each represented by one million MC particles. The initial weights are set based on the ideal MC scenario, $w_j = 1/N_p$ [78]; the weights are

subject to change over time as they are propagated by a dynamic filter. Propagating these particles over long periods would require re-sampling to discard poorly weighted particles and generate new particles with better weights [78, 79]. In this work, the propagation times are assumed short enough to not require re-sampling.

In the MC example, the viewing direction is chosen by summing the particle weights that fall within each grid field, h, and selecting the maximum value V, as given by:

$$V = \arg\max_{h} \Big(\sum_{i=1}^{n} \sum_{j=1}^{N_p} w_j(i) \cdot d_j(h, p_j(i))) \Big) \tag{6.3}$$

subject to:

$$\begin{aligned} \tau - \frac{1}{2}\text{FOV} - \tau_{p_j(i)} &\leq 0 \\ \tau + \frac{1}{2}\text{FOV} - \tau_{p_j(i)} &\geq 0 \\ \delta - \frac{1}{2}\text{FOV} - \delta_{p_j(i)} &\leq 0 \\ \delta + \frac{1}{2}\text{FOV} - \delta_{p_j(i)} &\geq 0 \end{aligned}$$

where $p_j(i)$ is the j^{th} particle representing object i, $w_j(i)$ is the particle's weight, and $d_j(h, p_j(i)))$ is one if all four conditions in equation 6.3 are met and zero otherwise. The conditions determine if the particle falls within the FOV, with (τ, δ) and $(\tau_{p_j(i)}, \delta_{p_j(i)})$ being the angles of the grid field center and the angles to the particles, respectively.

If immediate measurement feedback is assumed, either of the methods described can be used with Bayes' rule to produce desirable results. In the absence of measurement feedback, the probability that an object was observed (based on particles or distributions) is not available, so Bayes' rule cannot be employed. Not having these probabilities is the situation that is dealt with when there is a time delay in the feedback, between taking measurements and receiving the processed information. Since, the sensor is not required to stop and wait for measurements to be processed before moving to the next observation, a method is needed to account for a partic-

ular viewing direction where objects have potentially been observed. However, the objects move with respect to the grid fields, so simply making a grid field unavailable for future observations will not produce an optimal follow-up strategy.

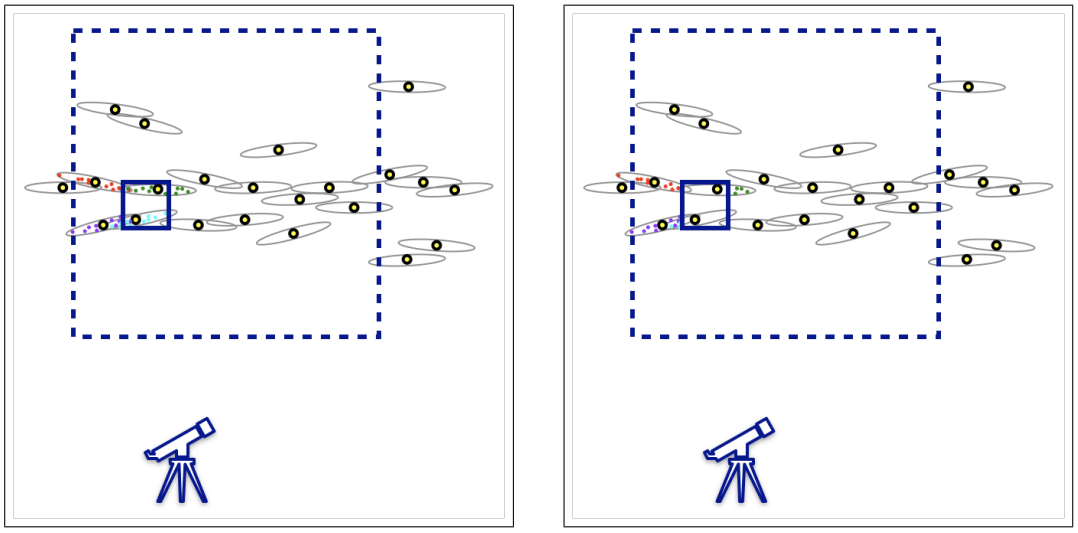
6.2 Monte Carlo Without Feedback

In the absence of immediate measurement feedback, a MC analysis is desirable for approximating the feedback, as the particles represent the true evolution of the PDF throughout the propagation; as long as re-sampling is not required, the particles will continue to represent the PDF. The particles are points within the state space that represent probable states for the objects. The particles are then transformed from the state space to the measurement space (by equations 2.22-2.23), and the optimization algorithm uses the particles to choose the viewing direction; those particles that contributed to selecting a viewing direction are known because they meet the conditions in equation 6.3. To handle the lack of feedback, the particles that contributed to the selection are removed from consideration by having their weights set to zero:

$$w_j(i) = \begin{cases} 0 & \text{if } d_j(h, p_j(i)) = 1 \\ w_j(i) & \text{if } d_j(h, p_j(i)) = 0 \end{cases} \tag{6.4}$$

Figure 6.2 shows an example of using feedback via a Monte Carlo analysis. In Figure 6.2(a), the optimization algorithm chooses the viewing direction for the sensor. The particles that led to that choice are then turned off as shown in Figure 6.2(b). Every particle is propagated to the future observation times, but those that previously contributed to selecting a viewing direction no longer contribute to V in equation 6.3.

The MC analysis will, theoretically, produce an optimal solution in the absence of measurements, as long as re-sampling is not required. Additionally, with enough particles, it accurately represents the behavior of the actual PDF in the presence of the non-linear orbit dynamics. However, it is computationally expensive because of



(a) Choice of Viewing Direction. (b) Remove Monte Carlo Particles.

Figure 6.2. When true measurement feedback is not available, Monte Carlo particles can provide a good approximation of the feedback.

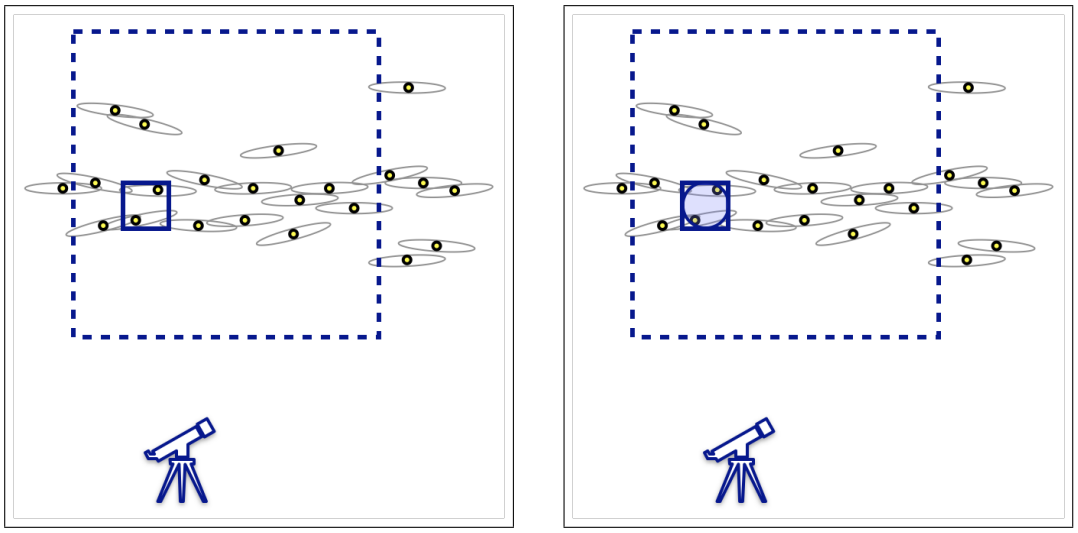
the need to propagate large numbers of particles for every object; assuming $N_p = 10^6$ for orbit dynamics, propagating a handful of objects is possible, but a few hundred objects can quickly result in days of computation for hours worth of observations.

6.3 Predicted Measurement Probability

To generate a feasible method without using MC particles, there must be a connection between the way the sensor operates and the way the targets move through the measurement space. The Predicted Measurement Probability (PMP) method seeks to estimate the measurement as an object with a PDF, which can then be used to provide feedback that predicts what was observed in the measurement.

Figure 6.3 shows the same objects in Figure 6.2, without using particles to represent the objects, but using a PMP approximate the measurement feedback. The PMP has a state, $\widetilde{\mathbf{X}}_{PMP}$, which has the same form as the states of the target objects, $\widetilde{\mathbf{X}}_i$. In this work, the PMPs are assumed to have multi-variate Gaussian PDFs,

$\mathcal{N}(\widetilde{\mathbf{X}}_{PMP}, \mathbf{P}_{PMP})$, to represent the viewing direction (FOV) as a distribution about the PMP state.



(a) Choice of Viewing Direction. (b) PMP represents the measurement.

Figure 6.3. When true measurement feedback is not available, a PMP is generated to provide a good approximation of the feedback.

Generating a PMP, with a PDF similar to the target objects being tracked, allows for propagation in the same EKF framework. The advantage over the MC method is, instead of propagating 10^6 particles for each target object, there is one PMP to propagate for each observation.

The influence of the MC particles is dealt with directly, by changing the particle weights. For the PMPs, the influence is dealt with in the same way as the target objects; CDF values are calculated for each PMP within each of the grid fields. Each PMP also has a value assigned, $\Upsilon(m)$, which represents the value of the observation for

which the PMP was generated; $\Upsilon(m)$ scales the CDF values for the PMP. Therefore, when using the PMPs, the value for choosing the next viewing direction is given by:

$$V = \arg\max_{h} \left(\sum_{i=1}^{n} \int_{h} f(\widetilde{\mathbf{X}}_i, \mathbf{P}_i) \mathrm{d}h - \sum_{m=1}^{p} \Upsilon(m) \int_{h} f(\widetilde{\mathbf{X}}_m, \mathbf{P}_m) \mathrm{d}h \right) \tag{6.5}$$

where:

$$f(\widetilde{\mathbf{X}}_i, \mathbf{P}_i) = \frac{1}{\sqrt{(2\pi)^2 |\mathbf{P}_i|}} \exp\left(-\frac{(\mathbf{X}_{i,h} - \widetilde{\mathbf{X}}_i)^T \mathbf{P}_i^{-1} (\mathbf{X}_{i,h} - \widetilde{\mathbf{X}}_i)}{2} \right) \tag{6.6}$$

where m represents a given PMP and p is the number of PMPs currently used (number of observations previously taken). Equation 6.5 is the same as equation 6.2, with the scaled CDF values for the PMPs subtracted to account for previous observations. The V that leads to assignment of the next viewing direction is also then used as the value for the new PMP, $\Upsilon(p+1) = V$.

6.3.1 Two Dimensional Problem

Before deriving the PMP method for the non-linear orbit problem in real space, a simplified two dimensional problem will be discussed. Consider a robot moving along a two dimensional space that is divided into a set of fixed viewing directions. The viewing directions are defined by the (x, y) coordinates of the center; thus, the position coordinates are directly measured in this scenario. Solutions will be generated by both MC analysis and the PMP method.

The initial mean state, $\widetilde{\mathbf{X}}$, and covariance, $\mathbf{P}$, of the robot are given by:

$$\widetilde{\mathbf{X}} = \begin{bmatrix} x \\ y \\ u \\ v \end{bmatrix} = \begin{bmatrix} 1.0 \\ 1.0 \\ 0.1\ s^{-1} \\ 0.2\ s^{-1} \end{bmatrix} \qquad \mathbf{P} = \begin{bmatrix} 0.09 & 0 & 0 & 0 \\ 0 & 0.09 & 0 & 0 \\ 0 & 0 & 10^{-5} & 0 \\ 0 & 0 & 0 & 10^{-5} \end{bmatrix} \tag{6.7}$$

x and y represent the position of the robot, and u and v represent the velocity. Like the sensor tasking problem, the velocities are not observable by the sensor.

The position coordinates are unit-less and the velocities are given in units of s^{-1}. The initial position uncertainty is sized such that the 3-σ interval is approximately one distance unit from the mean, as shown in Figure 6.4. The initial uncertainty of the robot's actual position is represented by MC particles and by the σ intervals of the position portion of the PDF.

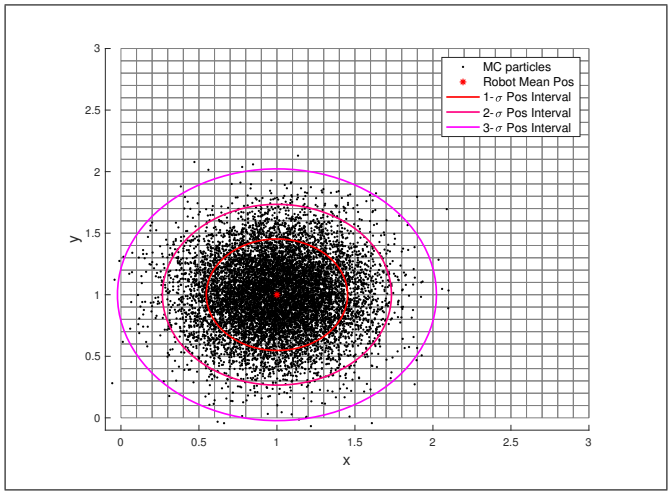


Figure 6.4. The initial position PDF and MC particles of the robot plotted against the grid of viewing directions across which it moves.

The viewing direction chosen, h, measures an area of the position space determined by the FOV size; in this example the FOV is 0.1×0.1 distance units. Using $n = 1$ (one object) in equation 6.3, for the MC particles, and equation 6.5, for the PMPs, viewing directions are chosen every five seconds for 100 seconds (20 observations). For the MC cases, 10,000 particles are used to represent the PDF; the weights of the particles are set to $w_j = 10^{-4}$, and the particle weights are updated based on equation 6.4.

For the PMP cases, the position of the PMP is set as the position of the center of the chosen viewing direction, (x_h, y_h), which is directly calculable in this simple case. Since the velocities are not measureable, the PMP velocities are initialized

as the velocity components (u_{rbt}, v_{rbt}) of the robot's mean state at the time of the observation. Thus, the state of a PMP at the time of generation is:

$$\widetilde{\mathbf{X}} = [x_h \quad y_h \quad u_{rbt} \quad v_{rbt}]^T \tag{6.8}$$

A covariance matrix is generated to complete the PDF for the PMP. The size of the FOV is known and the PMP PDF will represent the portion of the robot PDF that is contained within the FOV. Therefore, the position uncertainties are set to 3-σ_x = 3-$\sigma_y \approx 0.5 \cdot$ FOV, so the 3-σ interval of the position PDF lies on the edges of the FOV. With no way to measure the velocity or its uncertainty, the velocity covariance for the PMP is simply given the same values of the initial velocity uncertainties for the robot's covariance.

Four cases are tested for this two dimensional robot problem. In Case 1, the robot moves with linear velocity and no velocity uncertainty; the robot, MC particles, and PMPs all move with the same speed and direction. Case 2, the same linear velocity is repeated with velocity uncertainty included; here, the MC particles will move in slightly different directions or with slightly different speeds as compared to the robot PDF, while each PMP will be initialized with the velocity of the robot mean state. The last two cases use non-linear velocities: Case 3 with no velocity uncertainty, and Case 4 with velocity uncertainty. Since Cases 1 and 3 have no velocity uncertainty, the states can be reduced to only the position coordinates. The non-linear velocities used in Cases 3 and 4 are given by equations 6.9-6.10, where u and v are the current velocity components.

$$\dot{x}(t)_{non-linear} = u \cdot t + v \cdot \sin t \tag{6.9}$$

$$\dot{y}(t)_{non-linear} = v \cdot t + u \cdot \cos t \tag{6.10}$$

As the robot moves across the grid of fixed viewing directions, as shown in Figure 6.4, the viewing directions chosen for each observation are determined using a greedy

optimizer that selects the viewing direction which provides the most value. The values are determined by:

$$V = \arg\max_{h} \Big(\sum_{j=1}^{N_p} w_j \cdot d_j(h, p_j)) \Big)$$

subject to:

$$x_h - \frac{1}{2}\text{FOV} - x_{p_j} \leq 0$$
$$x_h + \frac{1}{2}\text{FOV} - x_{p_j} \geq 0$$
$$y_h - \frac{1}{2}\text{FOV} - y_{p_j} \leq 0$$
$$y_h + \frac{1}{2}\text{FOV} - y_{p_j} \geq 0$$

for the MC method, and:

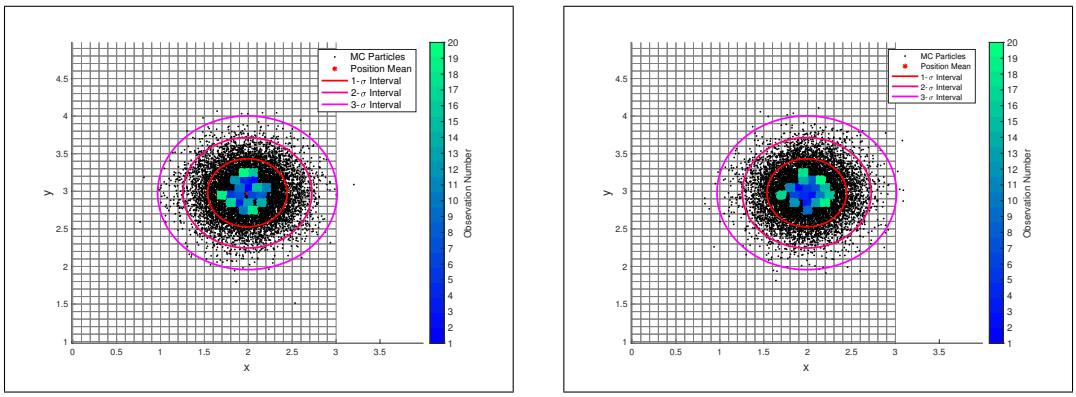
$$V = \arg\max_{h} \Big(\int_h f(\widetilde{\mathbf{X}}_{rbt}, \mathbf{P}_{rbt}) \mathrm{d}h - \sum_{m=1}^{p} \Upsilon(m) \int_h f(\widetilde{\mathbf{X}}_m, \mathbf{P}_m) \mathrm{d}h \Big)$$

for the PMP method. These equations are the same as equations 6.3 and 6.5, but modified for the robot problem. The observer is provided no feedback during the simulations.

Case by Case Comparison

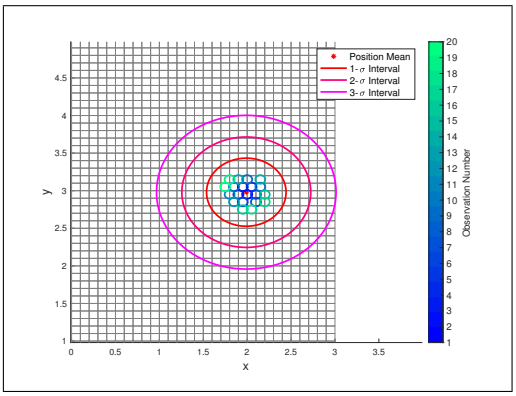
Figures 6.5-6.8 provide comparisons of the viewing directions chosen by two different MC representations and the PMP method, for each case of the robot problem. The order of the observations is indicated by the color (blue to bright green) given to the MC particles or the PMP that correspond to the viewing direction chosen for that observation. The MC particles are colored based on the first observation in which they are captured; they may be in viewing directions chosen at later times, but they are not counted as their weights are set to zero (equation 6.4) the first time they are observed. As expected, the three representations do not generate exactly the same observations in any of the four cases.

Case 1 is the most simple, since the MC particles remain in the same relative positions, with respect to the robot PDF, throughout the simulation. Figure 6.5 shows the observations for both methods are grouped in similar fashion about the mean position of the robot.



(a) Monte Carlo particles 1.

(b) Monte Carlo particles 2.

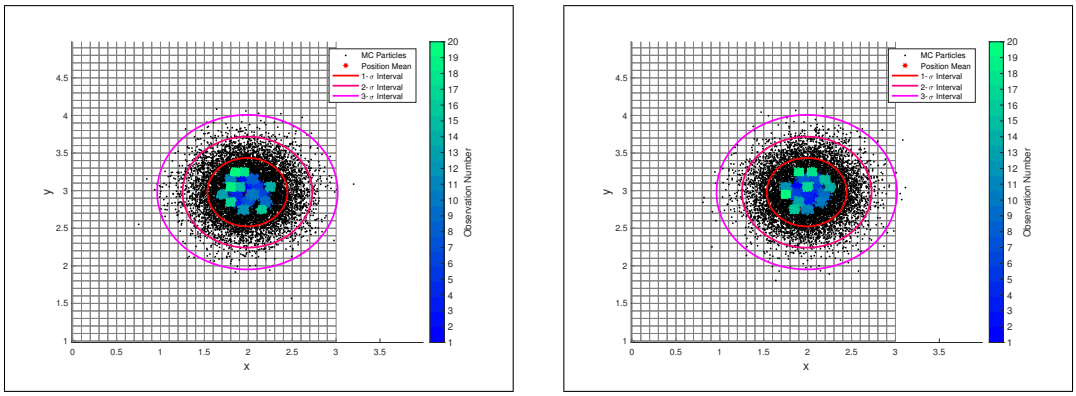


(c) Predicted Measurement Probability.

Figure 6.5. Case 1: All representations choose observations near the center of the robot's PDF.

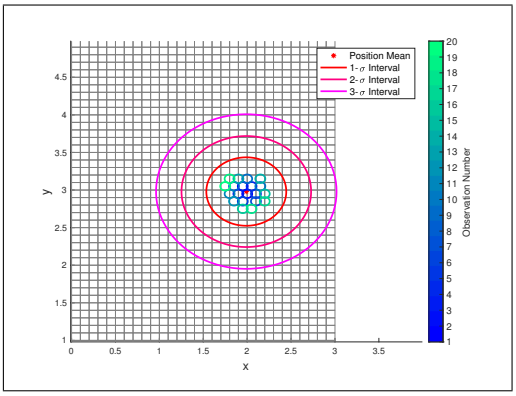
Case 2 includes a small uncertainty in the velocity, so the MC particles no longer remain in the same relative positions, with respect to the robot PDF, throughout the simulation. A small amount of process noise is added to all of the propagations to account for the changes in the robot PDF from the velocity uncertainty. Figure 6.6 shows the observations for each representation are still grouped in similar fashion

about the mean position of the robot. The distributions in 6.5(a) and 6.6(a) is different from that in 6.5(b) and 6.6(b), while those in 6.5(c) and 6.6(c) are the same; the MC method does handle the changes in the PDF better than the PMP method.



(a) Monte Carlo particles 1.

(b) Monte Carlo particles 2.

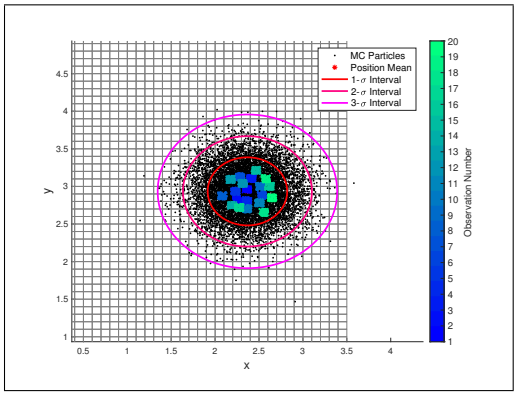


(c) Predicted Measurement Probability.

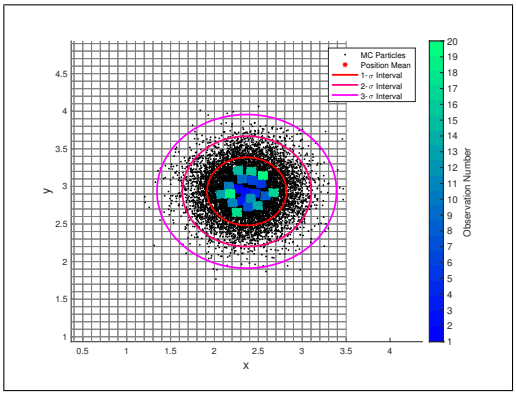
Figure 6.6. Case 2: The uncertainty in the velocity causes each MC representation to generate a different distribution of observations about the mean.

Case 3 introduces the non-linear motion without including velocity uncertainty. In Figure 6.7, the observations are all within the 1-σ position interval, but each MC representation has some particles near the mean that are not observed. The distribution of the PMP observations has changed from those in the linear cases,

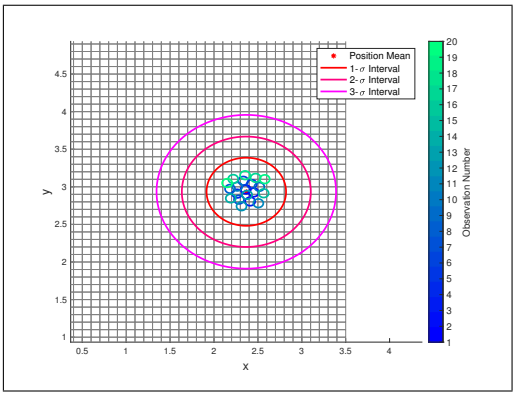
though they are still closely grouped around the mean. Because of the non-linear motion, there is more overlap in the observations produced by each method.



(a) Monte Carlo particles 1.



(b) Monte Carlo particles 2.

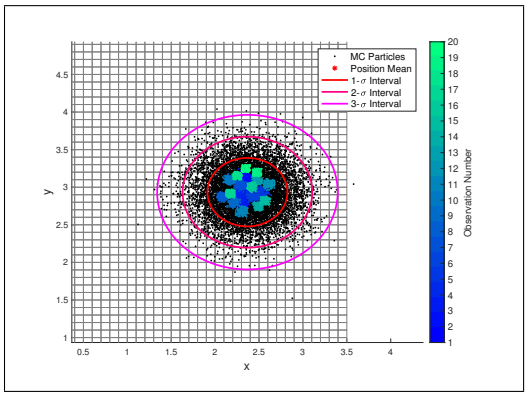


(c) Predicted Measurement Probability.

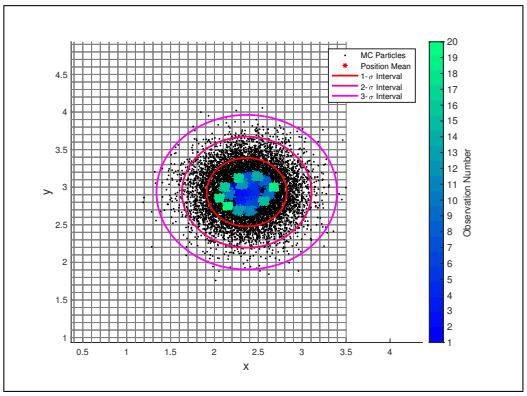
Figure 6.7. Case 3: The non-linear motion changes the distributions in each representation, though they remain close to the mean.

Case 4 introduces the velocity uncertainty along with the non-linear motion. Again, all observations are about the mean. Unlike Case 3, the observations generated by the MC representations do not leave as many noticeable gaps very close to the mean position of the robot. For the PMP method, the observations in Case 4 (6.8(c)) are different than those in Case 3 (6.7(c)); while the PMPs move with the

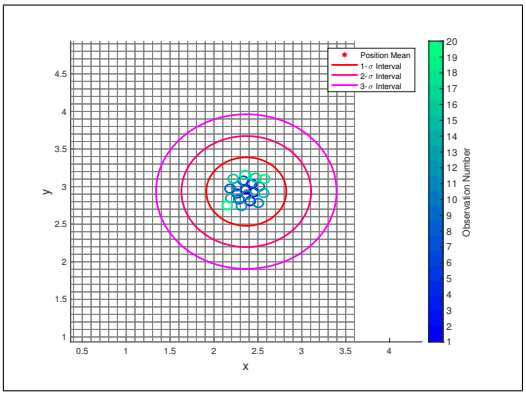
robot, the combination of non-linear motion and growing PDFs cause the noticeable difference in the location of the observations.



(a) Monte Carlo particles .



(b) Monte Carlo particles 2.



(c) Predicted Measurement Probability.

Figure 6.8. Case 4: The uncertainty with the non-linear motion causes more changes to the observation distributions, though they still remain near the mean.

These cases are limited representations of the problem. However, they show that the PMP method does not produce drastically different solutions from two different MC representations of the PDF. Therefore, the method, qualitatively, appears to be an effective approximation of the results found with a MC analysis.

Comparison of all Four Cases

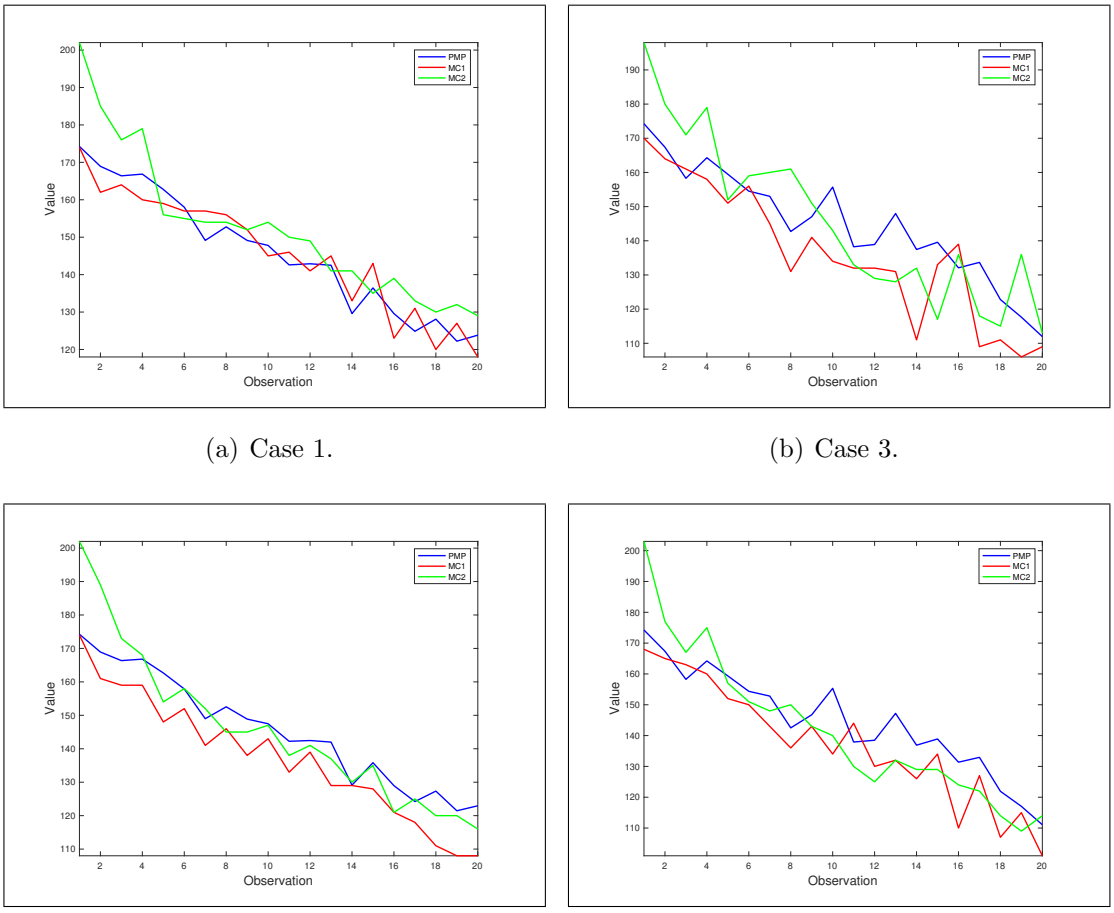
Figures 6.5-6.8 provide qualitative comparisons of the MC and PMP methods. In order to quantitatively compare the results, the values that each method used to assign the viewing direction at the observation times are compared. To do so, the number of MC particles is used as the value for that method, and the CDF value for the grid field chosen at each observation is multiplied by the total number of MC particles ($N_p = 10,000$).

Figure 6.9 shows the comparison of the values for the observations chosen by each method, for each case. Figure 6.9(a) (Case 1) shows the best agreement between the number of particles and the scaled CDF values chosen by the MC and PMP methods; this is expected as the motion is linear and consistent among the robot PDF, the MC particles, and the PMPs. Figure 6.9(c) (Case 2), presents larger differences as the MC values are generally lower than in Case 1; the different velocities of the MC particles are causing motion that leads to the lower MC values. Figures 6.9(d) (Case 4) and 6.9(b) show large changes in the observation values for both methods throughout the simulation. Figure 6.9(b) shows the largest differences between the methods at specific points (observations 14 and 17), indicating that the uncertainty in the particle velocities actually improves the solution.

In Figure 6.9, all cases show similar trends in the values of the observations; the highest values are found at the first observation and each successive observation finds a lower value. The qualitative and quantitative comparisons indicate that the PMP method should provide a similar ability to provide feedback to the optimizer, as that available with the MC method, while only propagating one object for each observation.

Time to Exhaust

The final analysis for the robot problem was to look at how the two methods performed by comparing the time to complete a task. A second simulation was run for



(a) Case 1.

(b) Case 3.

(c) Case 2.

(d) Case 4.

Figure 6.9. A comparison of the values for the viewing directions chosen by each MC representation and the PMP method, for each case.

the linear case with known velocity. Instead of using a fixed number of observations, a stopping condition was set up to "exhaust" the search. For the MC method this meant that observations were taken until all particles had been accounted for in choosing viewing directions. Because the PDF is infinite, the PMP method could theoretically continue forever. Thus, the stopping condition is set, such that, the simulation ends when the remaining CDF values to choose from are smaller than 10^{-4}; this is equivalent to the weight of an individual MC particle.

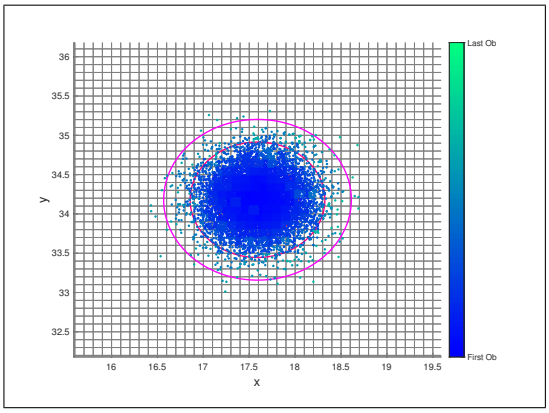
The same two cases of MC particles are run to compare with the case of the PMP method using CDF values. Table 6.1 shows the number of observations and computation time required to "exhaust" the search for the robot in each case. The MC methods use more observations, and consequently search more of the probable area where the robot may exist; this supports the understanding that a MC analysis should produce better results. However, they require significantly longer computation times due to the need to propagate 10,000 particles and the robot PDF. The difference in the number of observations required by the PMP method and each of the MC cases is less than 10% of the totals, while the PMP method requires over nine hours less computation time.

Table 6.1. The number of observations and computation time required to exhaust the linear problem with known velocity.

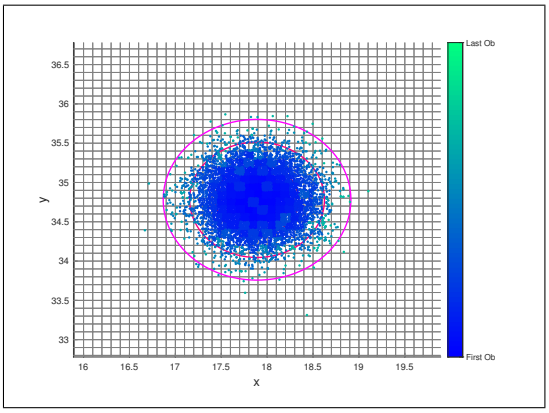
	PMPs	MC Particles 1	MC Particles 2
Observations	315	332	338
Compute Time	16 mins 22 secs	9 hrs 58 mins 8 secs	9 hrs 26 mins 31 secs

The cases for the MC method use more observations before exhaustion because the particles extend beyond, and are sparsely distributed as they approach, the 3-σ interval of the robot's PDF. This can be seen can be seen in Figure 6.10, which shows the PMPs and both sets of the MC particles at the point of exhaustion for each method. The wide dispersion of particles requires the MC method to assign more observations in order to measure every particle. By comparison, the PMP method only assigns viewing directions within the 3-σ interval, and assigns overlapping viewing directions, near the mean, toward the end of the simulation.

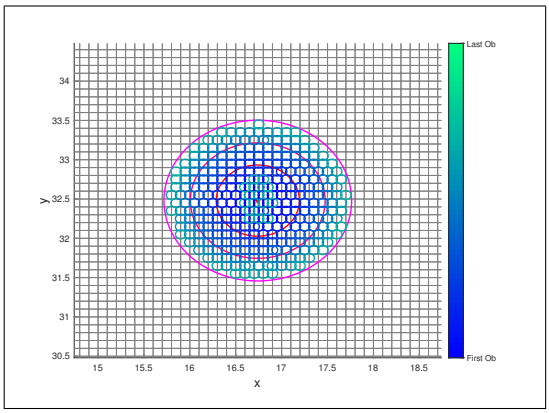
While the PMP method does not exactly match the MC representations in generating the viewing directions or in the time taken to exhaust the search, the patterns are similar. Based on this analysis, the method appears to be a plausible tool for use in providing predicted feedback. The rest of this chapter will develop the PMP method



(a) Monte Carlo particles 1.



(b) Monte Carlo particles 2.



(c) Predicted Measurement Probability.

Figure 6.10. The Monte Carlo particles and PMP observations when the methods are allowed to continue to exhaustion; the color indicates the order of the observations (blue to bright green).

for the sensor tasking problem and present simulation results for a two ground-based sensor scenario.

6.4 PMP in the SSA Sensor Tasking Problem

The definition of the PMP method for the SSA sensor tasking problem requires further development. In the two dimensional robot example, the position coordinates

were measured directly. In the sensor tasking problem, the measurements provide two angular quantities (e.g. τ and δ), which are related to the three dimensional position coordinates. To create the PMP for the sensor tasking problem, a full six dimensional state for the mean and an associated, suitable covariance must be generated.

The intent is for the PMPs to represent the observations that have been collected, which potentially contain known objects; without true measurement feedback, any of the objects could potentially have been observed, so the PMP should represent each of them, to some degree. Therefore, the quantities required to define the PMP state that are not directly observed $(\rho, \dot{\rho}, \dot{\tau}, \dot{\delta})$ must be estimated based upon the PDFs of the objects that contribute to that choice of viewing direction.

The following sections develop two methods for generating PMPs. First, a hypothesis for the full state of the PMP is presented in terms of the spherical coordinate frame, $\mathbf{X} = [\rho, \tau, \delta, \dot{\rho}, \dot{\tau}, \dot{\delta}]^T$. This hypothesis must be solved numerically for each of the target objects that contribute to choosing the viewing direction, which increases the computational load of the method. Next, to avoid a numerical solution, the generation of the PMP is broken into generation of the position and velocity vectors, separately; this results in an analytic method that can be efficiently calculated. Estimation of the PMP position vector is presented for both a single target object and the general case of multiple target objects. Finally, the generation of the covariance for the PMP is discussed, completing the generation of the PDF for the PMP.

6.4.1 Full State Estimation

The grid field, h, chosen for an observation provides two angles; for this development, the ground-based sensor is used with the LME coordinate frame angles, hour angle, τ, and declination, δ. The full state is generated in the spherical coordinates, where the four remaining coordinates $(\rho, \dot{\rho}, \dot{\tau}, \dot{\delta})$ are not observable by the sensor, and must be estimated to generate the PMP. The following development assumes a single target object PDF is being observed and the PMP PDF is hypothesized based on it.

The PMP that represents an observation is placed at the center of the FOV as shown in Figure 6.3. The initialization of the PMP state seeks to represent the most likely state of the target object given that its location is fixed by the angles τ and δ; i.e., as a simplification, the PMP represents the most probable state of the target object, given that its location is fixed to the center of the FOV. Assuming the target object PDF is Gaussian, the state located at the center of the FOV, with the highest probability, is defined as:

$$\{\rho, \dot{\rho}, \dot{\tau}, \dot{\delta}\} = \min\left(\frac{1}{2}(\mathbf{X} - \widetilde{\mathbf{X}}_s)^T \mathbf{P}\mathbf{z}^{-1}(\mathbf{X} - \widetilde{\mathbf{X}}_s)\right) \tag{6.11}$$

where $\mathbf{X}$ is the desired state of the target object given the angles (τ, δ), $\widetilde{\mathbf{X}}_s = [\rho_\mu, \tau_\mu, \delta_\mu, \dot{\rho}_\mu, \dot{\tau}_\mu, \dot{\delta}_\mu]^T$ is the mean state of the target object, and $\mathbf{P}\mathbf{z}^{-1}$ is the covariance of the target object expressed in terms of the spherical coordinates. Equation 6.11 minimizes the argument of the exponential term in the multi-variate Gaussian PDF, which produces the same result as maximizing the PDF.

The target mean state, $\widetilde{\mathbf{X}}$, and covariance, $\mathbf{P}$, are propagated by an Extended Kalman Filter, in the Cartesian frame. Equations 6.12-6.17 provide the relations for calculating the mean state in spherical coordinates:

$$\rho_\mu = \sqrt{(x_\mu - x_{st})^2 + (y_\mu - y_{st})^2 + (z_\mu - z_{st})^2} \tag{6.12}$$

$$\tau_\mu = \tan^{-1}\left(\frac{(x_\mu - x_{st})\sin\theta - (y_\mu - y_{st})\cos\theta}{(x_\mu - x_{st})\cos\theta + (y_\mu - y_{st})\sin\theta}\right) \tag{6.13}$$

$$\delta_\mu = \sin^{-1}\left(\frac{(z_\mu - z_{st})}{\rho_\mu}\right) \tag{6.14}$$

$$\dot{\rho}_\mu = \frac{(\dot{x}_\mu - \dot{x}_{st}) \cdot (x_\mu - x_{st}) + (\dot{y}_\mu - \dot{y}_{st}) \cdot (y_\mu - y_{st}) + \dot{z}_\mu \cdot (z_\mu - z_{st})}{\sqrt{(x_\mu - x_{st})^2 + (y_\mu - y_{st})^2 + (z_\mu - z_{st})^2}} \tag{6.15}$$

$$\dot{\tau}_\mu = \frac{(\dot{x}_\mu - \dot{x}_{st}) \cdot (y_\mu - y_{st}) - (x_\mu - x_{st}) \cdot (\dot{y}_\mu - \dot{y}_{st})}{(x_\mu - x_{st})^2 + (y_\mu - y_{st})^2} \tag{6.16}$$

$$\dot{\delta}_\mu = \frac{\sqrt{(x_\mu - x_{st})^2 + (y_\mu - y_{st})^2} \cdot \dot{z}_\mu}{\rho_\mu^2} - \frac{\{(\dot{x}_\mu - \dot{x}_{st}) \cdot (x_\mu - x_{st}) + (\dot{y}_\mu - \dot{y}_{st}) \cdot (y_\mu - y_{st})\} \cdot (z_\mu - z_{st})}{\sqrt{(x_\mu - x_{st})^2 + (y_\mu - y_{st})^2} \cdot \rho_\mu^2} \tag{6.17}$$

where, $x_{st},\ y_{st},\ z_{st},\ \dot{x}_{st},\ \dot{y}_{st}$ are the Cartesian coordinates of the observer state, the $\dot{z}_{st}$ component is assumed zero for a ground-based sensor, and $x_{\mu},\ y_{\mu},\ z_{\mu},\ \dot{x}_{\mu},\ \dot{y}_{\mu},\ \dot{z}_{\mu}$ are the coordinates for the mean state of the target object. In this work, the covariance in Cartesian space $\mathbf{P}$ is transformed via the 6×6 Jacobian matrix, $\mathbf{H}$, into the spherical coordinates, using a linearizing assumption:

$$\mathbf{Pz} = \mathbf{H} \cdot \mathbf{P} \cdot \mathbf{H}^T \tag{6.18}$$

The transformation errors via the linearization are assumed small for the full state. The partial derivatives that make up $\mathbf{H}$ are given in Appendix C.

Returning to equation 6.11, expanding the argument to be minimized results in an expression that is quadratic in each of the spherical coordinates:

$$\frac{1}{2}(\mathbf{X} - \widetilde{\mathbf{X}}_s)^T \mathbf{Pz}^{-1}(\mathbf{X} - \widetilde{\mathbf{X}}_s) = \tag{6.19}$$

$$\begin{aligned}
&\frac{1}{2}Pz^{\imath}_{1,1}\Delta\rho^2 + \frac{1}{2}Pz^{\imath}_{2,2}\Delta\tau^2 + \frac{1}{2}Pz^{\imath}_{3,3}\Delta\delta^2 + \frac{1}{2}Pz^{\imath}_{4,4}\Delta\dot{\rho}^2 + \frac{1}{2}Pz^{\imath}_{5,5}\Delta\dot{\tau}^2 + \frac{1}{2}Pz^{\imath}_{6,6}\Delta\dot{\delta}^2 \\
&\quad + Pz^{\imath}_{1,2}\Delta\rho\Delta\tau + Pz^{\imath}_{1,3}\Delta\rho\Delta\delta + Pz^{\imath}_{1,4}\Delta\rho\Delta\dot{\rho} + Pz^{\imath}_{1,5}\Delta\rho\Delta\dot{\tau} + Pz^{\imath}_{1,6}\Delta\rho\Delta\dot{\delta} \\
&\quad + Pz^{\imath}_{2,3}\Delta\tau\Delta\delta + Pz^{\imath}_{2,4}\Delta\tau\Delta\dot{\rho} + Pz^{\imath}_{2,5}\Delta\tau\Delta\dot{\tau} + Pz^{\imath}_{2,6}\Delta\tau\Delta\dot{\delta} \\
&\quad + Pz^{\imath}_{3,4}\Delta\delta\Delta\dot{\rho} + Pz^{\imath}_{3,5}\Delta\delta\Delta\dot{\tau} + Pz^{\imath}_{3,6}\Delta\delta\Delta\dot{\delta} \\
&\quad + Pz^{\imath}_{4,5}\Delta\dot{\rho}\Delta\dot{\tau} + Pz^{\imath}_{4,6}\Delta\dot{\rho}\Delta\dot{\delta} \\
&\quad + Pz^{\imath}_{5,6}\Delta\dot{\tau}\Delta\dot{\delta}
\end{aligned}$$

where, $\Delta\rho,\ \Delta\dot{\rho},\ \Delta\dot{\tau},\ \Delta\dot{\delta}$ are the differences between the desired coordinates and the mean coordinates (e.g., $\Delta\rho = \rho - \rho_{\mu}$); $\Delta\tau,\ \Delta\delta$ are the differences between the pointing angles and the mean angles for the object; and the $Pz^{\imath}_{i,j}$ terms are the elements of the inverted covariance matrix $\mathbf{Pz}^{-1}$.

Minimization is done in the usual way, taking the partial derivatives of the expression with respect to each of the desired coordinates and setting them equal to zero. Taking the partial derivatives leaves four equations which are linear in the desired

coordinates; because they are linear equations,the known values (e.g. $\tau,\ \delta,\ \rho_\mu$) may be separated from the desired values. The result is the following system of equations:

$$\begin{bmatrix} Pz^{\imath}_{1,1} & Pz^{\imath}_{1,4} & Pz^{\imath}_{1,5} & Pz^{\imath}_{1,6} \\ Pz^{\imath}_{1,4} & Pz^{\imath}_{4,4} & Pz^{\imath}_{4,5} & Pz^{\imath}_{4,6} \\ Pz^{\imath}_{1,5} & Pz^{\imath}_{4,5} & Pz^{\imath}_{5,5} & Pz^{\imath}_{5,6} \\ Pz^{\imath}_{1,6} & Pz^{\imath}_{4,6} & Pz^{\imath}_{5,6} & Pz^{\imath}_{6,6} \end{bmatrix} \begin{bmatrix} \rho_i \\ \dot{\rho}_i \\ \dot{\tau}_i \\ \dot{\delta}_i \end{bmatrix} =$$
$$\begin{bmatrix} Pz^{\imath}_{1,1}\rho_\mu + Pz^{\imath}_{1,4}\dot{\rho}_\mu + Pz^{\imath}_{1,5}\dot{\tau}_\mu + Pz^{\imath}_{1,6}\dot{\delta}_\mu \\ Pz^{\imath}_{1,4}\rho_\mu + Pz^{\imath}_{4,4}\dot{\rho}_\mu + Pz^{\imath}_{4,5}\dot{\tau}_\mu + Pz^{\imath}_{4,6}\dot{\delta}_\mu \\ Pz^{\imath}_{1,5}\rho_\mu + Pz^{\imath}_{4,5}\dot{\rho}_\mu + Pz^{\imath}_{5,5}\dot{\tau}_\mu + Pz^{\imath}_{5,6}\dot{\delta}_\mu \\ Pz^{\imath}_{1,6}\rho_\mu + Pz^{\imath}_{4,6}\dot{\rho}_\mu + Pz^{\imath}_{5,6}\dot{\tau}_\mu + Pz^{\imath}_{6,6}\dot{\delta}_\mu \end{bmatrix} - \begin{bmatrix} Pz^{\imath}_{1,2}\Delta\tau - Pz^{\imath}_{1,3}\Delta\delta \\ Pz^{\imath}_{2,4}\Delta\tau - Pz^{\imath}_{3,4}\Delta\delta \\ Pz^{\imath}_{2,5}\Delta\tau - Pz^{\imath}_{3,5}\Delta\delta \\ Pz^{\imath}_{2,6}\Delta\tau - Pz^{\imath}_{3,6}\Delta\delta \end{bmatrix} \tag{6.20}$$

If the matrix on the left side of equation 6.20 is invertible, a simple analytic solution is possible. Unfortunately, this is not generally the case, and a numerical method is required to find a solution. In order to find a computationally fast analytic expression, the computation of the position and velocity components for the PMP are separated.

6.4.2 Position Only PMP Hypothesis

Beginning with a single target object, a method for estimating the range to the PMP position is presented. The position of the PMP remains restricted to τ and δ, so only ρ must be estimated to fully define the position. Because the solution is only seeking an estimate for ρ, it can be accomplished without conversion into the spherical coordinates, but using the Cartesian position coordinates in terms of the spherical coordinates:

$$x = \rho\cos(\theta - \tau)\cos(\delta) + x_{st} \tag{6.21}$$

$$y = \rho\sin(\theta - \tau)\cos(\delta) + y_{st} \tag{6.22}$$

$$z = \rho\sin(\delta) + z_{st} \tag{6.23}$$

where the observer position coordinates are given by x_{st}, y_{st}, z_{st}.

Additionally, equation 6.11 simplifies to equation 6.24:

$$\rho = \min\left(\frac{1}{2}(\mathbf{X}_p - \widetilde{\mathbf{X}}_p)^T \mathbf{P}_p^{-1}(\mathbf{X}_p - \widetilde{\mathbf{X}}_p)\right) \tag{6.24}$$

where $\mathbf{X}_p$ represents the most probable position of the target object state at the given angles, while $\widetilde{\mathbf{X}}_p$ and $\mathbf{P}_p^{-1}$ are mean position and the position covariance marginalization, respectively. Leaving the coordinates in the Cartesian frame of propagation, removes the need to transform the covariance, that was required in the previous section.

Expanding the argument in equation 6.24 results in a single quadratic equation in ρ:

$$\frac{1}{2}(\mathbf{X}_p - \widetilde{\mathbf{X}}_p)^T \mathbf{P}_p^{-1}(\mathbf{X}_p - \widetilde{\mathbf{X}}_p) = D\rho^2 + E\rho + F \tag{6.25}$$

where:

$$\begin{aligned} D =& \frac{1}{2}P^{\imath}_{p(1,1)}\cos^2(\theta-\tau)\cos^2(\delta) + \frac{1}{2}P^{\imath}_{p(2,2)}\sin^2(\theta-\tau)\cos^2(\delta) + \frac{1}{2}P^{\imath}_{p(3,3)}\sin^2(\delta) + \\ & P^{\imath}_{p(1,2)}\cos(\theta-\tau)\sin(\theta-\tau)\cos(\delta) + P^{\imath}_{p(1,3)}\cos(\theta-\tau)\cos(\delta)\sin(\delta) + \\ & P^{\imath}_{p(2,3)}\sin(\theta-\tau)\cos(\delta)\sin(\delta) \end{aligned} \tag{6.26}$$

$$\begin{aligned} E =& \left[P^{\imath}_{p(1,1)}\cos(\theta-\tau)\cos(\delta) + P^{\imath}_{p(1,2)}\sin(\theta-\tau)\cos(\delta) + P^{\imath}_{p(1,3)}\sin(\delta)\right](x_{st} - x_\mu) + \\ & \left[P^{\imath}_{p(1,2)}\cos(\theta-\tau)\cos(\delta) + P^{\imath}_{p(2,2)}\sin(\theta-\tau)\cos(\delta) + P^{\imath}_{p(2,3)}\sin(\delta)\right](y_{st} - y_\mu) + \\ & \left[P^{\imath}_{p(1,3)}\cos(\theta-\tau)\cos(\delta) + P^{\imath}_{p(2,3)}\sin(\theta-\tau)\cos(\delta) + P^{\imath}_{p(3,3)}\sin(\delta)\right](z_{st} - z_\mu) \end{aligned} \tag{6.27}$$

$$\begin{aligned} F =& \frac{1}{2}P^{\imath}_{p(1,1)}(x_{st} - x_\mu)^2 + \frac{1}{2}P^{\imath}_{p(2,2)}(y_{st} - y_\mu)^2 + \frac{1}{2}P^{\imath}_{p(3,3)}(z_{st} - z_\mu)^2 + \\ & P^{\imath}_{p(1,2)}(x_{st} - x_\mu)(y_{st} - y_\mu) + P^{\imath}_{p(1,3)}(x_{st} - x_\mu)(z_{st} - z_\mu) + \\ & P^{\imath}_{p(2,3)}(y_{st} - y_\mu)(z_{st} - z_\mu) \end{aligned} \tag{6.28}$$

All of the values in equations 6.26-6.28 are known based on the target object's estimated state and covariance, the time of the observation, and the grid field chosen for the observation. The $P^{\imath}_{p(i,j)}$ terms are the elements of the inverted, marginalized covariance matrix, P_p^{-1}.

Taking the partial derivative of equation 6.25 with respect to ρ, setting it equal to zero, and solving for ρ results in the analytic solution:

$$\rho = -\frac{E}{2D} \tag{6.29}$$

which can be quickly calculated for any target object; E and D are given by equations 6.26-6.27. This provides a simple and efficient method for generating a position for the PMP.

6.4.3 Hypothesizing PMP Position and Velocity with Multiple Target Objects

The preceding development has assumed only one object. While an analytic solution was found for the range, and consequently the position, the velocity is unobservable by the optical sensor and thus a simple analytic solution is not possible. However, in the SSA sensor tasking problem many objects are present in the field of regard at any given time, and the PMP attempts to represent all possible objects that could have been observed. This fact is used to extend the solution for the position and to generate a solution for the PMP velocity vector.

Every object has a PDF, which is infinite, so all objects are contributing to all possible viewing directions. An estimated range for each object can be calculated by equation 6.29, but there must be some weighting to determine the influence each target object has on the range for the PMP. The CDF value for the object, found by integrating that object's PDF over the grid field, is used as the weight of the object in determining the PMP state. Since, the probability of seeing objects whose PDF is centered far from a given grid field is low, a cut-off value has been defined, where

objects with a CDF value that does not exceed the computer's machine precision (2.2204×10^{-16}) are considered to have zero weight.

This cut-off value limits the number of target objects for which equation 6.29 must be solved; instead of solving for all n target objects, only the k objects with a non-zero probabilities are considered. Equation 6.29 provides a ρ_i for each of the k objects, which are then combined to provide the range to the PMP as:

$$\rho_{PMP} = \frac{\sum_{i=1}^{k} w_i \cdot \rho_i}{\sum_{i=1}^{k} w_i} \tag{6.30}$$

where, w_i is the target weight. That is to say, the range for the PMP (ρ_{PMP}) is calculated as a weighted sum of the most probable target object ranges, given τ and δ. The result of equation 6.30 is combined with the pointing angles and the observer position at the time of the observation, to provide the ECI position vector, $\mathbf{X}_{p,PMP}$, using equations 6.21-6.23.

Because the observer cannot measure the velocity of the target objects, the PMP velocity cannot be solved for directly. As a simplification, the PMP velocity is generated by a weighted sum of the target object mean velocities, $\widetilde{\mathbf{X}}_{v,i}$:

$$\mathbf{X}_{v,PMP} = \frac{\sum_{i=1}^{k} w_i \cdot \widetilde{\mathbf{X}}_{v,i}}{\sum_{i=1}^{k} w_i} \tag{6.31}$$

where, the w_i are the same weights, for the same k objects, used in equation 6.30.

This method for generating the velocity estimate is not truly consistent with the position estimate; however, the PMP's position represents a weighted average of the contributing target objects in the grid field observed, while its velocity generally represents a hypothesis of the motion of those same target objects. Using a weighted sum of the target mean velocities produces a PMP velocity that achieves such a hypothesis.

6.4.4 Hypothesizing PMP Marginalized Covariances

The next step is to define the marginalized position and velocity covariances, which are used to initialize the PDF for the PMP; the initial PDF is defined to be Gaussian. The position uncertainties are defined in the range and the measurement angles (ρ, τ, δ) and rotated into the Cartesian space via:

$$\mathbf{P}_{x,y,z} = \mathbf{H} \cdot \mathbf{P}_{\rho,\tau,\delta} \cdot \mathbf{H}^T \tag{6.32}$$

where H is the 3×3 Jacobian matrix relating the spherical to the Cartesian coordinates.

The FOV of the sensor enforces strict limitations in the angles that can be observed. Using this fact, the angular variances are sized such that the volume of the associated 3-σ interval lies almost completely within the grid field being represented. Figure 6.11) shows an example of a PMP at the time of initialization; this PMP is based on the grid for the POGS sensor with a $3^\circ \times 3^\circ$ FOV. By placing the 3-σ interval in this way, the CDF value for the PMP PDF is ≈ 1 at the time of initialization, and within the grid field it represents; in other words, at the time of initialization, the PMP almost exclusively represents the grid field chosen for that observation.

While the angular limits are clearly defined, the range that the sensor can observe is not, though the target objects are limited to a given orbit regime. In order for the PMP PDF to capture a significant portion of the possible ranges in the GEO region, the range variance is defined so that the 3-σ interval extends along the pointing vector of the sensor, in both directions. Figure 6.12 shows the marginalized PDF volume for a PMP, the pointing direction that is aligned with the axis of the range variance, and the mean positions of the target objects. The width of the volume about the pointing direction is due to the angular variances, as shown in Figure 6.11, rotated into the ECI frame.

While the angle variances are based on the FOV and the range variance is based on the possible ranges to objects in the GEO region, there is no clear basis on which

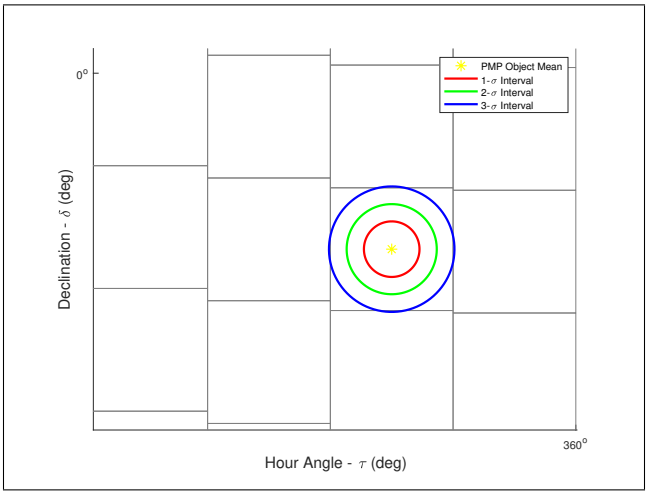


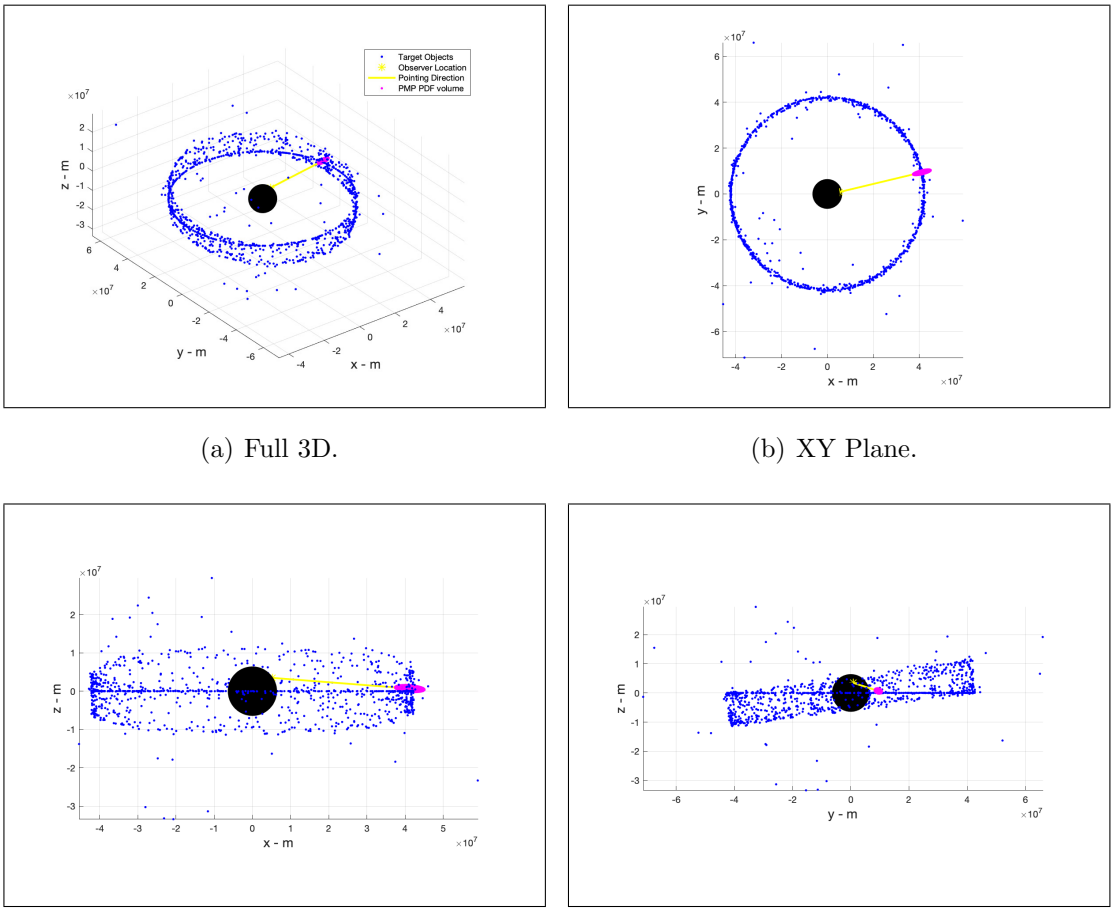
Figure 6.11. The 3-σ interval of the PMP's PDF touches the edges of the grid field at the time of initialization.

to define the velocity variances for the PMP. For this work, the velocity variances for the PMPs are initialized with the same initial velocity variances that the target object PDFs are initialized with at the start of the simulation; they are assumed to initially be independent and equal Gaussian terms in the Cartesian velocity coordinates, $\sigma_{vel} = \sigma_{\dot{x}} = \sigma_{\dot{y}} = \sigma_{\dot{z}}$. With the marginalized position and velocity covariances defined, the full covariance for the PMP is given as:

$$P(m) = \begin{bmatrix} P_{x,y,z}(m) & \mathbf{0} \\ \mathbf{0} & \sigma_{vel}^2 \cdot \mathbb{1} \end{bmatrix} \tag{6.33}$$

where, $P_{x,y,z}(m)$ is the marginalized position covariance in Cartesian coordinates obtained in equation 6.32, and $\mathbf{0}$ and $\mathbb{1}$ are the 3×3 zero and identity matrices, respectively.

As with all PDFs, integration over all possible values returns a probability of one. However, the PMP is representing the value of the observation, which is equal to the



(a) Full 3D.

(b) XY Plane.

(c) XZ Plane.

(d) YZ plane.

Figure 6.12. The 3-σ surface of the PMP PDF is shown with the point vector from the observer to the PMP mean position.

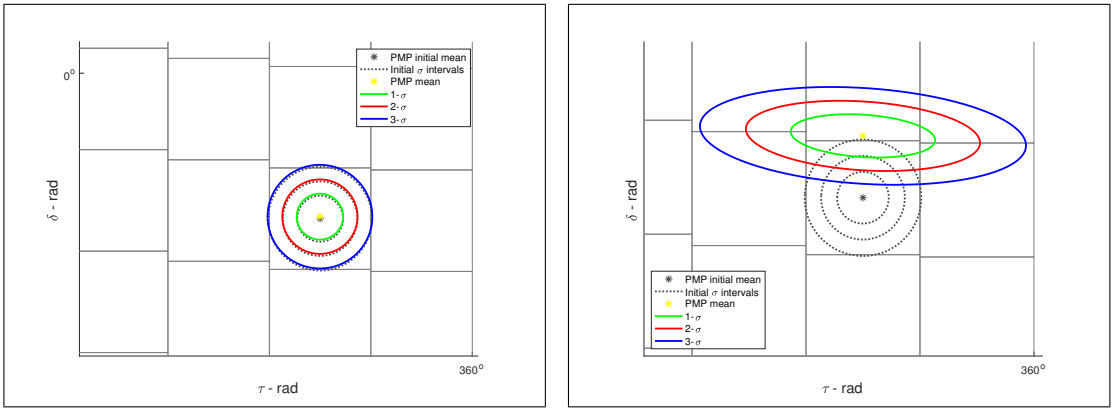
combined CDF values, scaled by the probability of detection $p_d(h, \widetilde{\mathbf{X}}_i)$, for each target object in the viewing direction. Therefore, the PMP will be assigned the value:

$$\Upsilon(m) = \sum_{i=1}^{n} p_d(h, \widetilde{\mathbf{X}}_i) \cdot d(h, \widetilde{\mathbf{X}}_i, \mathbf{P}_i) \tag{6.34}$$

where, $d(h, \widetilde{\mathbf{X}}_i, \mathbf{P}_i)$ is the CDF of object i for the viewing direction h. This value, $\Upsilon(m)$, represents the portions of all target object PDFs that contributed to the selection of the associated observation, m, and may be greater than one for grid fields

where many objects are potentially observed. When the PMP PDF is propagated forward, its CDF is calculated for all of the grid fields and scaled by this value. In this way the CDF calculation for the PMP, scaled by $\Upsilon(m)$, is subtracting out the predicted value of an already taken measurement.

The PMPs are propagated in the same filter as the target objects, experiencing the same non-linear effects common to all orbiting objects. In this way, the PDFs for the PMPs will evolve over time; like the target objects, it is assumed that a PMP PDF remains well approximated as Gaussian throughout the simulation. Figure 6.13 shows the evolution of the PMP from Figure 6.11; Figure 6.13(a) shows the object after it is propagated for 640 seconds (5 observations), and Figure 6.13(b) after propagation for 141 minutes (66 observations). In Figure 6.13(a), the PDF has shifted slightly, but remains well within the grid field in which it was derived, which discourages re-observation.



(a) After five observations.

(b) After 66 observations

Figure 6.13. A Predicted Measurement Probability object propagated to later observation times.

Figure 6.13(b) shows that after longer propagation times, the object's PDF shifts and changes shape with respect to the measurement space. This is how the PMP attempts to best approximate the average non-linear motion of the objects that contributed to its creation. As the PMP is spread over multiple grid fields, its impact on

any one of them becomes some fraction of the initial value of the observation based on the CDF of the PMP in the given grid field; the PMP CDF is restricted by the same cut-off value as the target object CDFs. This spreading of the PMP is still acting to diminish the portions of the object PDFs, but now those objects and the PMP have shifted in space and are no longer effecting just the grid field where the PMP was originated.

6.5 Simulation and Results

Analysis of the PMP method is based on several simulation scenarios with ground-based sensors. For all of the scenarios, the target objects simulated were selected from the publicly available Two-Line Element (TLE) catalog[1] published for 1 January 2019; all object were in the semi-major axis range, $37000 \leq a \leq 45000$ kilometers, with no restrictions on the other orbital elements (1138 total). The initial object states are generated by propagating the TLEs with SGP4 to four hours before the start of the observation window. Each object is then given state uncertainties of $\sigma_{pos} = 50$ kilometers and $\sigma_{vel} = 1$ meter per second. The initial state and covariance represent the first two moments of the PDFs for each object; the "true" states are sampled from these initial state-covariance pairs, and propagated along with the PDF representations. All remaining propagations are done using two-body motion in an Extended Kalman Filter framework.

The first scenario provides a comparison between the PMP method, as defined in sections 6.4.3-6.4.4, and a Monte Carlo simulation in a limited duration, single sensor observation window; both methods use the greedy optimizer described in section 4.3.1 to generate the solutions. The second scenario tests the PMP method with two sensors to generate pre-determined observation plans for each, without measurement feedback; the PMPs generated by each sensor are shared during the optimization of the observation plans. The observation plans are executed with no communication

[1]From https://www.space-track.org

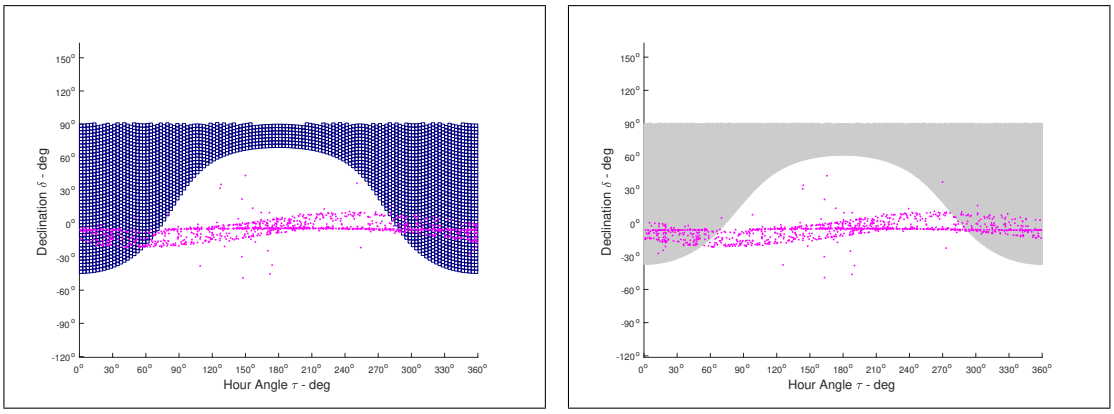
between the sensors during the actual observation window. The greedy and ACO optimizers are employed to generate the observation plans and performance is discussed. The use of these optimizers shows that the method can be used in classical and learning type optimization framework; the ACO was chosen because it was found to generally produce the best solutions for the ground-based sensor in Section 5.2.1. In the last scenario, the cooperative sensor tasking from the second scenario is compared to tasking generated for the sensors individually. Each sensor uses the PMP method to generate its observation plan with the greedy optimizer, but unlike the previous scenario, the PMPs are not shared between the sensors during the optimization. It is shown that despite the lack of communication between the sensors during the actual observation window, a cooperative tasking solution is still preferred over individual tasking solutions.

Two geographically separated observers, with different observation characteristics are used in this analysis. The first is the Purdue Optical Ground Station (POGS) (Lat: 33° N, Long: 106° W), which was introduced in section 5.2; again, it has 3° square FOV that is used to identify the grid fields that cover the FOR; the POGS sensor is used in all of the scenarios. The second sensor is based on the location of the Purdue Astronomy department's Cumberland Observatory (PACO) (Lat: 40° N, Long: 87° W)[2], and is modeled as having a rectangular FOV, $1.47° \times 1.02°$ ($\tau \times \delta$) for dividing the FOR into grid fields. Both sensors also have a minimum elevation constraint set to 12° above the local horizontal plane, which reduces the number of allowable pointing directions. The Local Meridian Equatorial (LME) system is used to define the measurement space, and measurement angles (τ, δ), for both sensors due to the advantages discussed in sections 2.2.1 and 3.2.1.

Each sensor's FOR is discretized based on its FOV as shown in Figure 6.14; the target objects discussed above are overlaid with the discretized FOR (grids) for each sensor. Each grid is fixed in hour angle τ and declination δ, and are only populated with the allowed viewing directions, satisfying the minimal elevation constraint.

[2]PACO position vector: $\bar{R}_2^{ECEF} \approx [262, -4846, 4139]^T$ km

White areas below the grids in Figure 6.14 represent viewing directions, which may contain objects, but are not possible to be chosen for the sensor. These directions are blocked by the Earth; the objects shown in these areas are behind the Earth.



(a) POGS grid and GEO objects.

(b) PACO grid and GEO objects.

Figure 6.14. Each sensors' grid and the catalog of objects are shown in each sensors' measurement space at a common epoch.

The sensors are not co-located, but they have overlapping FORs, which results in many of the objects being visible to both sensors. Figure 6.15 shows the same grids and objects from Figure 6.14, represented about the Earth in the ECI coordinate frame. The pointing directions that define the grid field centers are shown, presenting the significant amount of overlap in the FORs. Of the objects that are visible at some point during the observation interval, many objects will be visible to both sensors, while some will only be visible to one of the sensors.

In addition to the sensor locations and FOV sizes, the time between observations (from equation 4.8) are simulated to be different for the two sensors. For the POGS sensor, the time between starting two consecutive observations is $t_{obs} = 128$ seconds, while for the PACO sensor it is $t_{obs} = 188$ seconds. The number of images, exposure time, and readout time for the two sensors are the same, but PACO is modeled as having a slower re-positioning time, resulting in the longer time between observa-

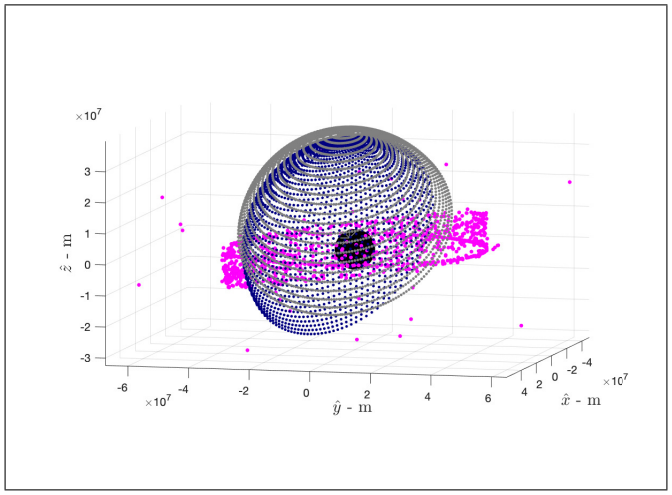


Figure 6.15. The overlapping FORs for the sensors are shown with the GEO object positions plotted in three-dimensional space. The grid field centers for POGS are shown in blue, and for PACO in gray.

tions. It is assumed that both observers are able to observe at every assigned time throughout the observation window, thus the number of temporal separation of the observations are fixed.

Another aspect of the two sensors having different FOV sizes, is that the angle uncertainties $(\sigma_\tau^2, \sigma_\delta^2)$ used to generate the PMP PDFs are unique to each sensor. Since the POGS FOV is square, the angle uncertainties are equal and given by $\sigma_\tau = \sigma_\delta = 7.8540 \times 10^{-3}$ radians. For PACO, the rectangular FOV leads to different values for variance of each angle, $\sigma_\tau = 3.8485 \times 10^{-3}$ and $\sigma_\delta = 2.6704 \times 10^{-3}$. The range variance is the same for both sensors, $\sigma_\rho = 1,333.33$ kilometers. The velocity uncertainties are set to $\sigma_{\dot{x}} = \sigma_{\dot{y}} = \sigma_{\dot{z}} = 1\ m/s$, which is the same velocity uncertainty used to initialize the covariances of the target objects before the start of the simulation.

6.5.1 Monte Carlo vs. PMP for the Sensor Tasking Problem

In order to compare the PMP method with a MC analysis for the sensor tasking problem, a smaller scale comparison is performed. Each method is used to provide

predicted feedback during a 90 minute window and a 6 hour window on the night of January 3, 2019; both windows begin at 19:35:16 Mountain Standard Time (Jan 4, 19 01:35:16 Universal time).

In order to perform such a comparison, the number of objects included in the catalog is reduced to 200 objects falling within the sensor's FOR at the beginning of the observation window. Additionally, for the MC analysis, each target object is only represented by 1000 particles; this is much less than the ideal 10^6 particles (for a six dimensional state space), but it was chosen to make the computation time required for propagation tractable.

Table 6.2 shows the results of the sensor tasking solutions generated using the MC and PMP methods for the two durations. The RSOs Observed columns provide the number of unique objects observed based on the solutions generated; only the first observation of an object was included in this value. The Computation Times columns record how long it took to generate the solutions for each method and each observation duration. Despite limiting the number of MC particles, the computation times are significantly longer than the PMP method. As expected, the MC method outperforms the PMP method in the 90 minute window, observing five more objects. The fact that the two methods find the same value for the longer window may be due to the limited number of MC particles not fully representing the target PDFs, and thus not fully representing the benefits of a Monte Carlo analysis, or it could also be due to the limited number of RSOs used in the analysis and the non-convex nature of the problem making it possible for the observer to reach the maximum with many different strategies [36].

While the shorter duration solution shows that the PMP method does not provide the same accuracy as the MC particles, the difference between the two methods is only five objects, 6.25% of the MC total; the predicted feedback from the PMP method provides an efficient method for generating solutions in the absence of measurement feedback. Next the PMP method is applied to two sensors with overlapping observation durations and fields of regard.

Table 6.2. The number of RSOs observed and computation times required to generate sensor tasking solutions using the Monte Carlo and Predictive Measurement Probability methods for observer feedback.

	RSOs Observed		Computation Time	
	90 Minutes	6 Hours	90 Minutes	6 Hours
Monte Carlo	80	160	9 hrs	36.5 hrs
PMP	75	160	90 secs	4 mins 47 secs

6.5.2 Cooperative Sensor Tasking Problem using PMP Method

This scenario simulates the two sensors operating on the night of January 2, 2019; it is assumed that weather conditions are good at both sensor locations, for the entire night. The simulation time frame for the PACO sensor runs from Jan 2, 2232 hours 19.50 seconds UT, to Jan 3, 1136 hours UT; the POGS sensor runs from Jan 3, 0047 hours 38.27 seconds UT, to Jan 3, 1320 hours 47.49 seconds UT. The times are based on local dusk and dawn conditions. The total observation times for the sensors are are 13 hours, 3 minutes for PACO and 12 hours, 33 minutes for POGS; the total observation is 14 hours, 48 minutes for both sensors together.

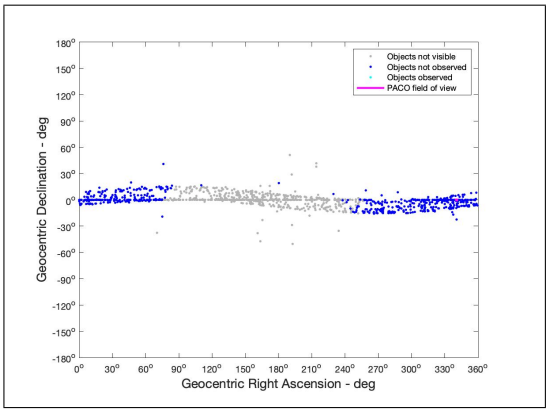
Over the entire simulation, a total of 603 observations are generated between the two sensors; 250 observations with PACO and 353 observations with POGS. A total of 1138 objects are selected from the TLE catalog, but as shown in Figure 6.14, many of these objects are not visible to either sensor. To reduce the computational load of the simulation, the catalog was reduced to only those objects whose PDFs generated a consequential CDF value (greater than machine precision, $\approx 2.2204 \times 10^{-16}$), for at least one of the sensors during the window. This reduced the catalog to 565 objects, mainly in geosynchronous region (GEO), with some objects in geosynchronous transfer orbits (GTO). The true states that determine if objects are seen, which are sampled from the initial states and covariances, are propagated to each observation

time, along with the mean states and covariances. The value of each viewing direction and the final determination of which RSOs are observed is dependent on the calculated p_d values for the objects; thus, this scenario is most like the third scenario for the ground-based sensor in Chapter 5.

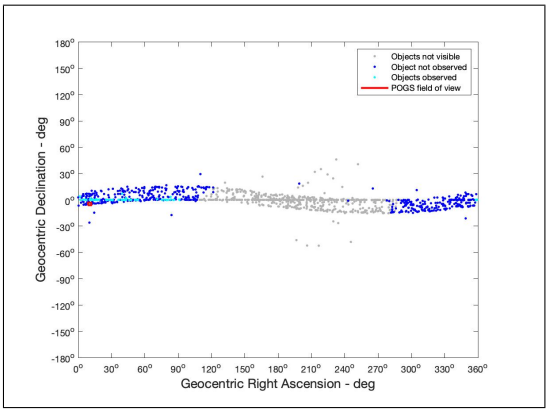
It is helpful to visualize the mechanics of the solutions generated, by looking at the evolution of objects observed in a common reference frame. Figure 6.16 shows four instances of observations being taken during the window with the sensors cooperating (generated using the greedy solution). Figures 6.16(a)-6.16(b) show the first observations taken by PACO and POGS, respectively, while Figures 6.16(c)-6.16(d) show the final observations taken by each sensor. The objects that have been observed are plotted in cyan, the objects not yet observed are in blue, and the sensor taking the observation is represented by pink (PACO) or red (POGS) rectangles representing the viewing direction at that time; the objects in grey are those objects from the TLE catalog that are not considered in the simulation. Of the 565 objects simulated, only 511 are actually within the combined FOR for the two sensors at some time during the observation window; the bands of unseen objects on the edges of Figures 6.16(c) and 6.16(d) include those objects that are never visible to the sensors, but whose PDFs are considered by the optimizers.

For this cooperative sensor tasking problem, the simple greedy and the ACO optimization algorithms from Chapter 4 are employed; ACO was chosen over DQL because it performed better for the ground-based sensor in terms of number of RSOs observed, despite the long computation times. Both optimization strategies consider all PMPs generated by previous observations, regardless of which sensor generated them, when choosing each viewing direction; thus, while the sensors do not communicate during the actual observation interval, the solutions are generated with predicted feedback from both sensors.

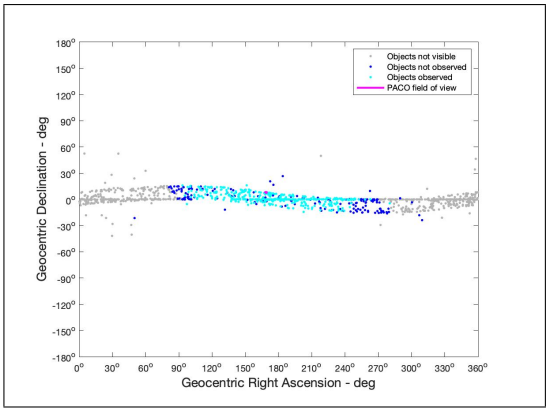
Using the ACO algorithm, each agent generates its own set of PMPs and does not consider the effect of other agents' PMPs when generating their own solution. Similar to when feedback is present, the influence of an agent's PMPs is incorporated



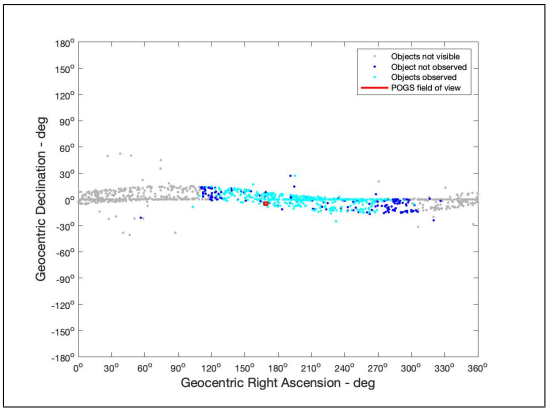
(a) First observation by PACO (obs #1).



(b) First observation by POGS (obs #45).



(c) Last observation by PACO (obs #553).



(d) Last observation by POGS (obs #603).

Figure 6.16. The 565 objects that could potentially be observed by the sensors are plotted over time. The angles are based on an inertial frame, causing the objects to drift in angle space as they orbit the Earth.

in the value of their solution through equation 6.5, which then gets incorporated into the pheromones in equation 4.25. Again, the successive sets of agents do not know which viewing directions were assigned by previous sets of agents, but again use the influence of the pheromones, that are now impacted by PMPs as well, to generate their solutions.

For this scenario, four cases of the ACO algorithm are compared: three sets of 20 agents, four set of 20 agents, five sets of 20 agents, and three sets of 50 agents. Because

of the probabilistic nature of the agents, each run of the ACO cases will produce slightly varied solutions; the random number generators are not seeded as they were for the simulations in Chapter 5, but use the CPU clock time of the computer to generate the random choices of the agents. The cases with three and four sets of 20 agents are each run twice to show the variability.

Each optimization case is compared based on the total number of objects (true states) observed by the solution, the A_v value of the solution, and the number of unique grid fields used in the solution. The unique grid fields are important in this case because, unlike the simulations in Chapter 5, the μ_i values are not reset during the optimization; the selection of unique grid fields shows that the PMP is effective in acting as the feedback and directing the algorithms to not repeatedly assign the same grid fields. The first column of Table 6.3 shows the total number of RSOs observed by each solution using the combined sensors; these are the number of unique RSOs observed at least once during the observation window. The % observed column represents the percentage of the objects observed based on the total number (511) that fall within the combined FOR at some point during the observation window. The A_v column represents the combined value of the viewing directions assigned by each solution. While the optimizers are not given feedback of the actual measurements during solution generation, the PMP provides an effective form of feedback, allowing the sensors to view approximately 75-81% of the objects in each solution. The ACO solutions are able to observe more objects than the greedy, but they require significant increases in the computation time [36]; the computation times are discussed in Appendix C.

Figure 6.17 shows the growth in the number of objects observed at least once during the complete observation window; the black line represents the 511 objects that could be observed. The trend of objects seen during the window shows that only the greedy solution is markedly different. The ACO solutions, despite differences in the number of iterations and the number of agents, are all very similar in their trends and their final values; running the simulations additional times may find better

Table 6.3. Comparing the number of objects observed and the percent of observable objects generated by the seven solutions.

	# Observed	% Observed
Greedy	385	75.34
ACO_{3x20} (1)	409	80.04
ACO_{3x20} (2)	417	81.60
ACO_{4x20} (1)	401	78.47
ACO_{4x20} (2)	403	78.86
ACO_{5x20}	403	78.86
ACO_{3x50}	414	81.02

solutions with four or five iterations of twenty agents and worse solutions with three iterations of 20 or 50 agents.

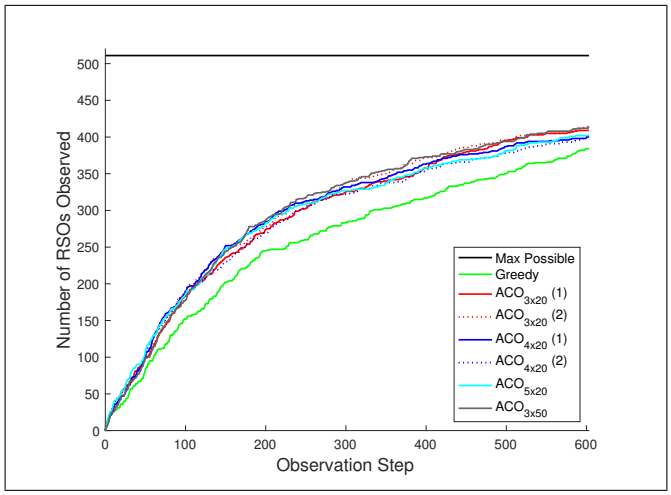


Figure 6.17. The growth in objects observed as the observation window progresses.

In Chapter 5, the value, A_v, is introduced as an additional comparison for each solution. The same comparison can be applied in this scenario, but the values now represent the summation of the weights, as calculated by equation 6.5, over the full solution. Table 6.4 shows the A_v value and the number of objects observed for each of the solutions using the PMP for feedback. In this scenario, the μ_i values are never set to zero, which means that an object could produce non-zero CDF values for many of the selected viewing directions during the observation period.

Table 6.4. The summation of the grid field values for each solution generated using PMP for feedback.

	# Observed	A_v
Greedy	385	1482.09
ACO_{3x20} (1)	409	1249.00
ACO_{3x20} (2)	417	1246.35
ACO_{4x20} (1)	401	1287.38
ACO_{4x20} (2)	403	1251.26
ACO_{5x20}	403	1293.12
ACO_{3x50}	414	1228.72

As with the immediate feedback case, the greedy solution is trying to find the maximum value at every observation time and this leads to the largest A_v value among the solutions. With the ACO algorithm, the agents are learning which grid fields lead to high values over the full solution without assigning the maximum value at every step; while the ACO solutions have lower A_v values than the greedy, they still find better solutions. However, among the ACO solutions, the highest A_v values do not lead to the most RSOs observed; the most RSOs are observed by the solutions with the two lowest A_v values. Additionally, the ACO_{4x20} (2) and ACO_{5x20} solutions observe the same number of RSOs, but differ by more than 40 in their A_v values.

The final comparison of the solutions illustrates how many unique pointing directions the solutions chose for each sensor. Since the pointing directions are based on the observer's Earth fixed location, any objects that are not geostationary will be moving with respect to the pointing directions. The PMP method does not force a unique pointing direction at every observation, but provides feedback to the optimizer which results in not repeatedly observing the same pointing direction. Previously observed pointing directions are likely to be re-selected by the optimizers after propagation of the PMPs away from those pointing directions. Additionally, the ACO optimizer includes influence from previously generated solutions, which influences the solution toward more diverse pointing direction selection.

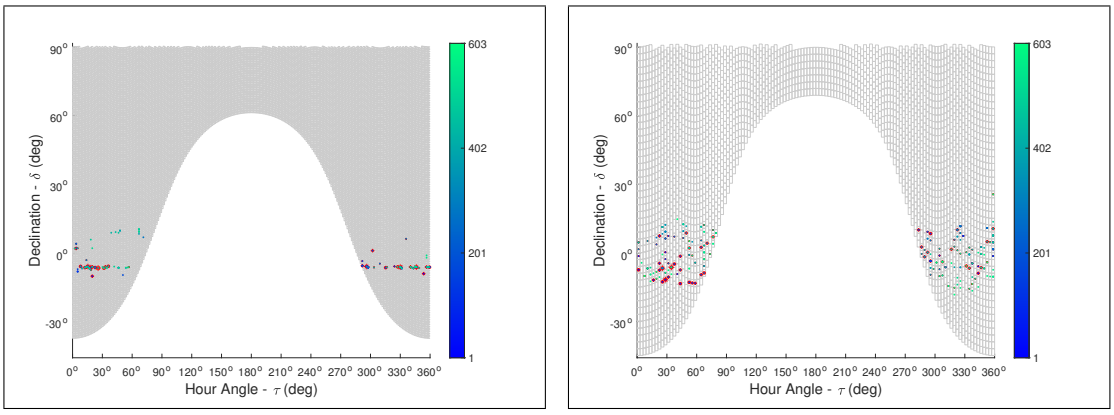
Table 6.5 provides a direct comparison of the number of unique pointing directions assigned by the solutions; the number of pointing directions are broken out by sensor in the center columns. Table 6.5 shows that all of the ACO solutions use more pointing directions than the greedy solution, both for the individual sensors and the combined totals. However, one is still not able to infer a direct relationship between selecting more unique grid fields and achieving a better solution. The solutions for ACO_{3x20} (1), ACO_{3x20} (2), and ACO_{3x50} use the most unique grid fields and observe the three highest number of RSOs; however, ACO_{3x20} (2) finds the most objects while using less unique grid fields than the other two solutions. Additionally, ACO_{4x20} (1) assigns more unique grid fields than ACO_{4x20} (2) or ACO_{5x20}, but observes two less RSOs. Selecting more unique grid fields is desirable, but it does not guarantee better solutions.

Figures 6.18-6.19 show the pointing directions chosen by the Greedy and ACO_{3x20} (2) solutions (rows 1 and 3 in table 6.5); the other ACO solutions are shown in Appendix C. Because there are two sensors, the sub-figures show pointing directions in each sensor's local measurement space. The color of the dots indicates to which observation time the viewing direction is associated, while the size of the red $*$ indicates the number of objects observed, for the first time, at the given viewing direction. Viewing directions where the blue/green dots are on top of a red $*$ indicate where

Table 6.5. Comparing the number of unique pointing directions (fixed to the observer) chosen by the solutions for each sensor.

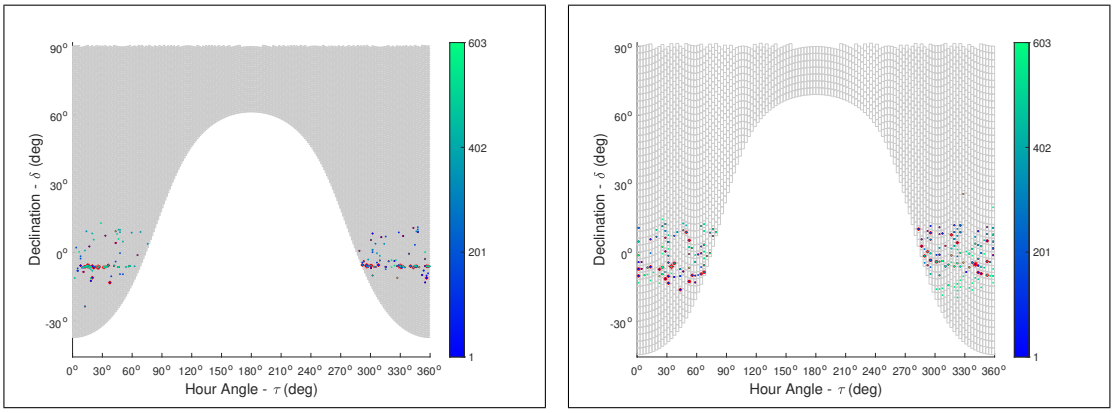
	PACO	POGS	Combined
Greedy	66	159	225
ACO_{3x20} (1)	131	196	327
ACO_{3x20} (2)	128	193	321
ACO_{4x20} (1)	116	187	303
ACO_{4x20} (2)	107	189	296
ACO_{5x20}	111	183	294
ACO_{3x50}	152	190	342

a grid field was chosen for more than one viewing direction. Dots that do not have a red $*$ collocated indicate viewing directions that did not observe any previously un-observed objects.



(a) PACO (b) POGS

Figure 6.18. The greedy viewing directions and numbers of objects seen at each step are shown for each of the sensors in their respective measurement grid.



(a) PACO

(b) POGS

Figure 6.19. The ACO_{3x20} (2) viewing directions and numbers of objects seen at each step are shown for each of the sensors in their respective measurement grid.

The pointing directions for the PACO sensor (Figures 6.12(a) and 6.14(a)) show a strong tendency to align viewing directions along the GEO belt ($\approx 0°$ inclinations), while those for the POGS sensor are more distributed about the GEO belt. This is due, in part, to the fact that PACO operates for two hours and 15 minutes before the beginning of the observations for POGS, and the objects mean states are most tightly grouped around the GEO belt. The tight grouping of the mean states leads to higher combined CDF values along the GEO belt, and the optimizers tend toward selection of the associated pointing directions. The assignment of the early PACO viewing directions along the GEO belt generates PMPs near the GEO belt, which help to encourage the viewing directions for POGS to be more diverse with respect to the entire GEO region.

6.5.3 Individual vs. Cooperative Sensor Tasking with PMP

Cooperatively tasking two sensors is expected to perform better than individual tasking each of the sensors. To check this, the same scenario used in the previous

section is used with the greedy optimizer, but with each sensor only being optimized based on its own set of PMPs. By not sharing the PMPs of both sensors in the algorithm, the optimizer is expected to choose overlapping viewing directions between the sensors, which will lead to a decrease in the number of unique objects observed by the combined sensor tasking solutions and a reduction in the number of unique grid fields assigned.

As a point of comparison for the individual sensor tasking solutions, the values of the greedy optimizer from the cooperative scenario, in the previous section, are used. While the solutions for the two sensors are individually generated, in order to make the comparison valid, the scenario uses the same times as in the previous scenario, with the same number of observations (250 for PACO, 353 for POGS). The performance of the individual tasking and the cooperative tasking are compared via the number of objects observed at least once, the number of objects observed twice or more, and the commonality of those sets of objects. The unique grid fields and the distribution of the viewing directions are also assessed.

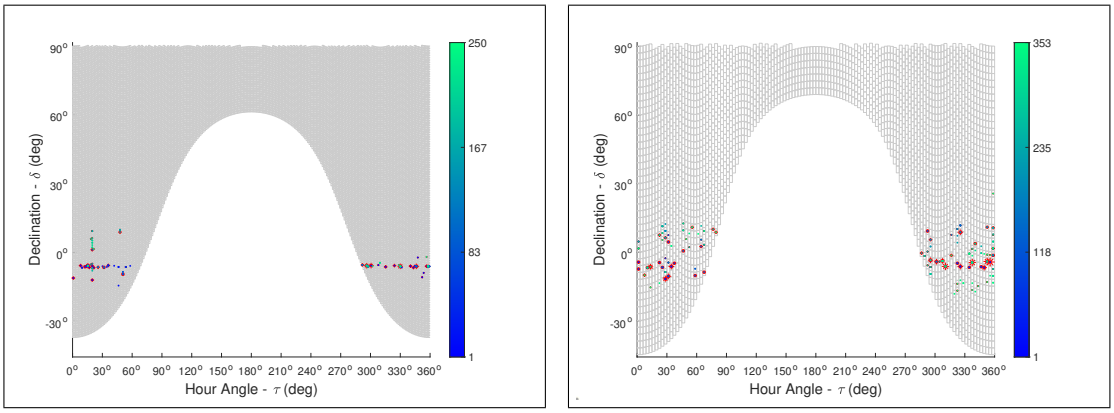
Table 6.6 shows the total number of objects observed by the sensors, individually and combined, and how many of the objects are uniquely seen by each sensor; the objects seen once also include all of the objects seen two or more times. The larger FOV and faster slew rate (more observations) for POGS allows for significantly more objects to be observed. However, the overlap in the fields of regard leads to a high number of objects being seen by both of the sensors when operating in ignorance. The unique objects denote those RSOs that are observed by one sensor but not the other sensor; because the FORs do not overlap completely, these unique objects may only be visible by one sensor. The combined value is computed as the sum of the number seen by each sensor individually, subtracting the number of objects that are seen by both sensors; in other words, all objects seen by one sensor, plus those objects uniquely seen by the other sensor (e.g. POGS total (291) + PACO Unique (33) = 324).

Table 6.6. Comparing the number of objects observed by each sensor, and the combined totals, when the sensors work individually.

	PACO	POGS	Combined
# Observed once	137	291	324
# Unique objects	33	187	-
# Observed twice +	81	223	242
# Unique objects	19	161	-

The number of unique grid fields assigned for each sensor also decreases when the optimization is done for the individual sensors. In the cooperative scenario, the PACO sensor was assigned 66 unique grid fields, while in the individual scenario the number was only 60. The POGS sensor saw a similar decrease from 159 unique grid fields in the cooperative case, to 106 in the individual case. This decrease in the number of grid fields can be seen by comparing Figure 6.20 with Figure 6.18. With the small difference in the number of grid fields assigned for the PACO sensor, there are only small differences between Figure 6.18(a) and Figure 6.20(a). For the POGS sensor, the larger difference in the number of unique grid fields assigned is also seen in the wider distribution of the grid fields in Figure 6.18(b) than in Figure 6.20(b).

Table 6.7 compares the combined number and percentages of objects seen when the sensors are optimized individually, and when they are optimized cooperatively. The cooperative sensor tasking optimization is able to observe 385 objects, while the individual sensor tasking optimization only observes a combined total of 324 objects; this is approximately a 12% increase in performance for the cooperative optimization. This corresponds to a 75% observation rate of all visible objects for the cooperative scenario, as compared to an observation rate of just over 63% when optimizing the sensors individually. The same pattern holds for the objects that are observed more than once; 242 individual objects were observed at least twice in the night by the



(a) PACO (b) POGS

Figure 6.20. The greedy viewing directions and numbers of objects seen at each step are shown for the individual sensor optimizations. The results are shown against each sensor's respective measurement grid.

individual sensors, as compared with 273 objects in the cooperative scenario. The increased numbers, when information is shared, indicates that PMPs provide effective feedback to the optimization algorithm, regarding each sensor's observations, even when no immediate observation processing is available. As expected, a common optimization for the sensors is preferred over employing the sensors completely independent of each other, even when pre-computed scenarios are used.

Table 6.7. Comparing the total number and percentage of objects observed when the sensors work individually and in cooperation.

	Individual Combined	Cooperative
# Observed once	324	385
% Observed once	63.41	75.34
# Observed twice +	242	273
% Observed twice +	47.36	53.42

6.6 Predictive Measurement Probability Summary

The PMP method has been introduced into the sensor tasking optimization problem for ground-based sensors observing objects in GEO. It is shown to be able to provide estimated feedback in the absence of true measurement feedback. In a limited single sensor scenario, the PMP is compared to a Monte Carlo analysis for providing feedback to the sensor tasking optimizer. The tasking optimization using the PMP method does not exactly match the performance (objects observed) of using a Monte Carlo analysis for providing feedback, but the computation time required by the PMP method is shown to be far superior. A cooperative two sensor scenario, where only the PMPs are shared between the sensors, is investigated and compared with the scenario where the individual sensors operate without considering the PMPs generated by the other sensor. The result shows that the cooperative scenario is superior to the individual optimizations. The improved performance of the sensors in the cooperative scenario indicates that sharing the PMPs from heterogeneous sensors provides the optimizer with effective feedback for creating a multi-sensor tasking solution.

7. SUMMARY

The optimization of sensor tasking is an area of extreme importance as the number of resident space objects continues to increase rapidly, while the number of sensors have historically increased at a much slower rate. There has been a significant amount of research related to the sensor tasking problem; from investigating the most opportune time to observe given objects, to minimizing the trace of the position covariances for a catalog of objects, to employing specific techniques for optimization. Much of the work has focused on one aspect, or one method for solving the problem.

The goal of this dissertation is the efficient generation of effective sensor tasking solutions for ground-based and space-based sensors, which are employed to observe objects in Geosynchronous Earth Orbits (GEO). This goal is achieved through the satisfaction of three objectives: 1) choosing a coordinate frame for each sensor that allows for the uncertainty in the orbits to be accurately represented in the measurement space, 2) comparison of four optimizers used to generate solutions to determine the efficiency and effectiveness of each, and 3) generation of an efficient method for optimization in the absence of measurement feedback. The first objective allows the optimization of the sensor tasking problem to be based on the probability density function (PDF) of every target object, as observed in the sensor's measurement frame, without inducing significant computational burden or detrimental information loss. The third objective provides the strategy for handling processing and communication delays in the sensor tasking process. The second objective, along with the first and third, provides the means of generating and analyzing the sensor tasking solutions in order meet the goal stated above.

7.1 Conclusions

The first objective is achieved by transforming the PDF of an object in GEO, along with a set of Monte Carlo particles representing the true PDF, into possible measurement spaces, as described in Chapter 3. The linear Jacobian matrices, which are the partials of the coordinates in one frame with respect to the other, are used to transform the PDF to the measurement spaces. By analyzing the Mahalanobis distances of each of the particles with respect to the transformed PDF, the measure of the transformation accuracy is determined. The Local Meridian Equatorial coordinate frame (τ, δ) is shown to be superior to the Local Meridian, Local Horizon coordinate frame (β, ζ); the LME coordinate frame provides a superior approximation of a linear transformation for the PDF from the Earth Centered Inertial frame into the measurement space. The Satellite Meridian Equatorial coordinate frames, which is analogous to the LME coordinate frame, is shown to be similarly effective as an approximation of a linear transformation for the PDF in the space-based sensor scenario.

The second objective is achieved through the analysis of three scenarios for each sensor, ground-based and space-based. The Greedy, Weapon-Target Assignment (WTA), Distributed Q-Learning (DQL), and Ant Colony Optimization (ACO) algorithms described in Chapter 4 are compared in terms of effectiveness and efficiency using the scenarios laid out in Chapter 5. The sensor tasking scenarios increase in complexity, beginning with perfect knowledge of the target positions and perfect detection, then using PDF representations of each object state with perfect detection, and finishing with PDF representations and a calculated probability of detection. In most cases, the ACO and DQL algorithms generate the most effective solutions; however, the solutions are not significantly better than those generated by the Greedy and WTA algorithms. The Greedy and WTA algorithms require significantly less computation time to generate solutions than do the ACO and DQL algorithms. Ad-

ditionally, the ACO and DQL algorithms require significant tuning prior to generating the solutions, while the Greedy and WTA algorithms require no such tuning.

The third objective is achieved through the development of the Predicted Measurement Probability (PMP). The PMP represents observations that sensors have performed, but for which no feedback is available, in order to guide future observation choices. The method represents feedback by generating PMPs, based on the observations assigned, which are propagated like the target objects; the PMPs provide the optimization algorithm with information about the previous observations by subtracting value from the grid field weights. In a single sensor scenario, the PMP method is shown to be more computationally efficient than a Monte Carlo analysis of each target object. In a two sensor scenario where no feedback is provided to the optimization algorithm, the PMP provides effective feedback that allows the optimizers to maximize the cost function for the sensor tasking problem.

7.2 Recommendations

There are still many questions that remain for generating optimal sensor tasking strategies in all situations. This work looked at four standard optimization methods, but there are many others that could be employed and potentially provide additional performance in terms of effectiveness, or efficiency, or both. This work also focused on objects in the GEO region, but the methods could be applied to other near Earth orbital regimes. Future work is likely to expand the complexity and types of the optimizers and investigating methods like parallel computing for reducing the computation time.

The cost function used in this work was simplified from the original formulation. Specifically, incorporating an object specific function that incorporates some measure of the desire or need to observe each object, or using the re-observation constraint to weight the viewing direction choices, will provide a more robust understanding of how well the optimizers perform. Additionally, the time between observations is

not required to be constant, the observation interval for each sensor could include downtime, and the number of sensors could be increased to stress the methods. Each of these changes provide an added dimension of realism in the sensor tasking problem.

Improvements in the PMP method are sought by relaxing the Gaussian assumption on the PDF of the target objects that are used to generate the PMPs. The uncertainty of orbit estimates is known to become non-Gaussian as the objects are propagated over time, especially in the Cartesian frame [53]. Since the PDF of the PMP represents the portions of the target object PDFs that map to the observation taken, using the non-Gaussian PDF of the target objects to generate the PMP object may lead to a more accurate representation of the observation. Additionally, a more robust velocity estimate for the PMP is sought.

The orbit representation of the target objects is another area for further investigation. Changing from the Cartesian frame to an orbital element frame will effect how the uncertainty grows over time, however the additional transformations required must be investigated to determine how they affect the transformation to the measurement space. Additionally, incorporating the orbital perturbations, discussed in Chapter 3, into the problem will provide a more realistic propagation of the objects.

Further investigation of the algorithms discussed in Chapters 4 and 5, and the advantage of parallel computing on the ACO, DQL, and WTA optimizers is expected to result in reduced computation times. As discussed in Chapter 5, the tuning of learning algorithms is specific to each problem; a Monte Carlo analysis could be used to determine the best tuning parameters for a given optimizer and scenario. Additionally, other Reinforcement Learning algorithms could provide better solutions through more robust learning or tuning methods.

REFERENCES

[1] ESA Space Operations. Space Situational Awareness Programme Overview, 2017.

[2] Erin Salinas. Space Situational Awareness is Space Battle Management, may 2018.

[3] David A. Vallado. *Fundamentals of Astrodynamics and Applications*. Microcosm Press/Springer, third edition, 2007.

[4] Dennis Roddy. *Satellite Communications*. McGraw-Hill Education, fourth edition, 2006.

[5] Space Operations. Space Debris: The European Space Agency Approach, mar 2017.

[6] National Air and Space Administration. National Air and Space Administration Technology Roadmaps: Introduction, Crosscutting Technologies, and Index, jul 2015.

[7] Carolin Frueh, Hauke Fiedler, and Johannes Herzog. Heuristic and Optimized Sensor Tasking Observation Strategies with Exemplification for Geosynchronous Objects. *Journal of Guidance, Control, and Dynamics*, 41(5):1036–1048, 2018.

[8] Richard H. Battin. *An Introduction to the Mathematics and Methods of Astrodynamics, Revised Edition*. American Institute of Aeronautics and Astronautics, Inc., 1999.

[9] William E. Wiesel. *Modern Orbit Determination*. Aphelion Press, 2003.

[10] Space Debris Office. European Space Agency's Annual Space Environment Report. Technical Report 2, European Space Agency, 2018.

[11] United States Strategic Command. Combined Space Operations Center, jul 2018.

[12] Johannes Herzog. *Cataloguing of Objects on High and Intermediate Altitude Orbits*. Dissertation, University of Bern, 2013.

[13] Edward P. Chatters and Brian J. Crothers. Space Surveillance Network. In Brian Tichenor, editor, *AU-18 Space Primer*, chapter 19, pages 249–258. Air University Press, 2009.

[14] Thomas Schildknecht. Optical surveys for space debris. *The Astronomy and Astrophysics Review*, 14(1):41–111, jan 2007.

[15] Alex M. Friedman and Carolin Frueh. Determining characteristics of artificial near-Earth objects using observability analysis. *Acta Astronautica*, 144:405–421, mar 2018.

[16] Victor G. Szebehely. *Adventures in Celestial Mechanics*. University of Texas Press, Austin, TX, 1989.

[17] O. Montenbruck, P. Steigenberger, and U. Hugentobler. Enhanced solar radiation pressure modeling for Galileo satellites. *Journal of Geodesy*, 89(3):283–297, 2015.

[18] Roshan Thomas Eapen. *Averaged Solar Radiation Pressure Modeling for High Area-to-Mass Ratio Objects in Geostationary Space*. Masters, Purdue University, 2017.

[19] Smriti Nandan Paul, Carolin Frueh, and Hauke Fiedler. Detection of Unknown Space Objects based on Optimal Sensor Tasking and Hypothesis Surfaces using Variational Equations : Example HAMR Objects. *Advances in Space Research*, submitted:1–41, 2019.

[20] Oliver Montenbruck and Thomas Pfleger. *Astronomy on the Personal Computer*. Springer Berlin Heidelberg, Berlin, Heidelberg, second edition, 1994.

[21] Carolin E. Frueh. AAE 590 Chapter 7, Orbit Propagation and Perturbations in the Near Earth Space, 2016.

[22] Oliver Montenbruck and Eberhard Gill. *Satellite Orbits: Models, Methods and Applications*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[23] Carolin E. Frueh. AAE 590 Chapter 4, Coordinate Systems and Time, 2016.

[24] Leonard Meirovitch. *Methods of Analytical Dynamics*. Dover Publications, Inc., 2003.

[25] David Vallado, Paul Crawford, Ricahrd Hujsak, and T.S. Kelso. Revisiting Spacetrack Report #3. In *AIAA/AAS Astrodynamics Specialist Conference*, pages 1–94, Keystone, Colorado, 2006.

[26] Wei Dong and Zhao Chang-yin. An Accuracy Analysis of the SGP4/SDP4 Model. *Chinese Astronomy and Astrophysics*, 34(1):69–76, 2010.

[27] T. Flohrer, H. Krag, H. Klinkrad, B. Bastida Virgili, and C. Früh. Improving ESA's collision risk estimates by an assessment of the TLE orbit errors of the US SSN catalogue, 2009.

[28] Francois Sanson and Carolin Frueh. Noise Estimation and Probability of Detection in Nonresolved Images: Application to Space Object Observation. *Advances in Space Research*, Submitted, 2018.

[29] Carolin E. Frueh. AAE 590 Chapter 3, Observations, 2016.

[30] Richard Linares and Roberto Furfaro. Dynamic Sensor Tasking for Space Situational Awareness via Reinforcement Learning. In *Advanced Maui Optical and Space Surveillance Technologies Conference 2017*, Maui, HI, 2017. Maui Economic Development Board.

[31] T. Schildknecht, U. Hugentobler, and M. Ploner. Optical Surveys of Space Debris in Geosynchronous Earth Orbit. *Advances in Space Research*, 23(1):45–54, 1999.

[32] T. Schildknecht, M. Ploner, and U. Hugentobler. The Search for Debris in Geosynchronous Eearth Orbit. *Advances in Space Research*, 28(9):1291–1299, 2001.

[33] Carolin Frueh. Sensor Tasking for Multi-Sensor Object Surveillance. In *7th European Conference on Space Debris*, Darmstadt, Germany, apr 2017.

[34] T. Schildknecht, R. Musci, M. Ploner, G. Beutler, W. Flury, J. Kuusela, J. de Leon Cruz, and L. de Fatima Dominguez Palmero. Optical Observations of Space Debris in Geosynchronous Earth Orbit and in Highly-Eccentric Orbits. *Advances in Space Research*, 34(5):901–911, 2004.

[35] Andris D. Jaunzemis, Marcus J. Holzinger, and Moriba K. Jah. Evidence-based Sensor Tasking for Space Domain Awareness. In *Advanced Maui Optical and Space Surveillance Technologies Conference*, Maui, HI, 2016. Maui Economic Development Board.

[36] Bryan D. Little and Carolin Frueh. SSA Sensor Tasking : Comparison of Machine Learning with Classical Optimization Methods. In *Advanced Maui Optical and Space Surveillance Technologies Conference*, pages 1–17, 2018.

[37] Kyle J. DeMars, Islam I. Hussein, Carolin Frueh, Moriba K. Jah, and R. Scott Erwin. Multiple-Object Space Surveillance Tracking Using Finite-Set Statistics. *Journal of Guidance, Control, and Dynamics*, 38(9):1741–1756, sep 2015.

[38] J. Geul, E. Mooij, and R. Noomen. Modelling and Assessment of the Current and Future Space Surveillance Network. *European Conference on Space Debris*, 7:4–9, 2017.

[39] T. Schildknecht, U. Hugentobler, and A. Verdun. Algorithms for Ground Based Optical Detection of Space Debris. *Advances in Space Research*, 16(11):47–50, 1995.

[40] Timothy P. Payne. New Deep Space Optical Search Strategies. In *Proceedings of the Fifth US-Russian Space Surveillance Workshop*, 2003.

[41] R. Musci, T. Schildknecht, and M. Ploner. Orbit Improvement for Geosynchronous Earth Orbit Objects Using Follow-up Observations. *Advances in Space Research*, 34(5):912–916, 2004.

[42] Andrea Milani, Giovanni F. Gronchi, Mattia De Michieli Vitturi, and Zoran Knežević. Orbit Determination with Very Short Arcs. I Admissible Regions. *Celestial Mechanics and Dynamical Astronomy*, 90(1-2):57–85, sep 2004.

[43] Keric Hill, Paul Sydney, Kris Hamada, Randy Cortez, Kim Luu, Moriba Jah, Paul W. Schumacher, Michael Coulman, Jeff Houchard, and Dale Naho'olewa. Covariance-Based Network Tasking of Optical Sensors. *Advances in the Astronautical Sciences*, 136(July):769–786, 2010.

[44] Zachary Sunberg, Suman Chakravorty, and Richard Scott Erwin. Information Space Receding Horizon Control for Multisensor Tasking Problems. *IEEE Transactions on Cybernetics*, 46(6):1325–1336, jun 2016.

[45] Richard Linares and Roberto Furfaro. An Autonomous Sensor Tasking Approach for Large Scale Space Object Cataloging. In *Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*, pages 1–17, 2017.

[46] R. Scott Erwin, Paul Albuquerque, Sudharman K. Jayaweera, and Islam Hussein. Dynamic sensor tasking for Space Situational Awareness. In *Proceedings of the 2010 American Control Conference*, pages 1153–1158. IEEE, jun 2010.

[47] Jayant Sharma. Space Surveillance with the Space-Based Visible Sensor. In S.E. Andrews, editor, *Space Control Conference*, pages 115–124, 2000.

[48] James R. Shell. Optimizing Orbital Debris Monitoring with Optical Telescopes. Advanced Maui Optical and Space Surveillance Technologies Conference, sep 2010.

[49] Mihir Patel, Andrew J. Sinclair, and Koki Ho. Information-Theoretic Target Search for Space Situational Awareness. In *2018 Space Flight Mechanics Meeting*, chapter AIAA SciTe. American Institute of Aeronautics and Astronautics, Kissimmee, Florida, 2018.

[50] Paul Maskell and Lorne Oram. Sapphire: Canada's Answer to Space-Based Surveillance of Orbital Objects. In *Advanced Maui Optical and Space Surveillance Technologies Conference*, Maui, HI, sep 2008.

[51] T. Flohrer, J. Peltonen, A. Kramer, T. Eronen, J. Kuusela, E. Riihonen, T. Schildknecht, E. Stöveken, E. Valtonen, F. Wokke, and W. Flury. Space-Based Optical Observations of Space Debris. In *Proceeding of the Fourth European Conference on Space Debris*, number ESA SP-587, pages 165–170, Darmstadt, Germany, 2005. European Space Agency.

[52] Sven K. Flegel and James Bennett. Normality in State Uncertainties from Orbit Determination Results Fitting Optical Measurements. In *Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*, 2018.

[53] Kyle T. Alfriend and Inkwan Park. When Does the Uncertainty Become Non-Gaussian. In *Advanced Maui Optical and Space Surveillance Technologies Conference*, 2016.

[54] Carolin Frueh. AAE 590 Chapter 8, Orbit Improvement/Filtering: Minimum Mean Square Error Estimation, 2016.

[55] P.C. Mahalanobis. On The Generlized Distance in Statistics. In *National Institute of Science*, volume 2, pages 49–55, India, 1936.

[56] R. De Maesschalck, D. Jouan-Rimbaud, and D.L. Massart. The Mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, 50(1):1–18, jan 2000.

[57] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[58] Roy H. Kwon. *Introduction to Linear Optimization and Extensions with MATLAB*. CRC Press, 2014.

[59] Kazuo Murota. *Discrete Convex Analysis*. Society for Industrial and Applied Mathematics, jan 2003.

[60] Kazuo Murota. Recent Developments in Discrete Convex Analysis. In W. Cook, L. Lovász, and J. Vygen, editors, *Research Trends in Combinatorial Optimization*, pages 219–260. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[61] Timothy S. Murphy and Marcus J. Holzinger. Generalized Minimum-Time Follow-up Approaches Applied to Tasking Electro-Optical Sensor Tasking. In *Advanced Maui Optical and Space Surveillance (AMOS) Technologies Conference*, 2017.

[62] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, may 1996.

[63] David L. Neel and Nancy Ann Neudauer. Matroids You Have Known. *Mathematics Magazine*, 82(1):26–41, feb 2009.

[64] Patrick A Hosein, James T Walton, and Michael Athans. Dynamic Weapon-Target Assignment Problems with Vulnerable C 2 Nodes. Report LIDS-P-1786, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, jun 1988.

[65] Robert A. Murphey. *Target-Based Weapon Target Assignment Problems*, chapter Target-Bas, pages 39–53. Kluwer Academic Publishers, 2000.

[66] Nimrod Lilith and Kutluyıl Doğançay. Reinforcement Learning-Based Dynamic Scheduling for Threat. In *16th European Signal Processing Conference*, pages 1–5, Lausanne, Switzerland, 2008. EURASIP.

[67] Reinaldo A.C. Bianchi, Carlos H.C. Ribeiro, and Anna H.R. Costa. On the Relation Between Ant Colony Optimization and Heuristically Accelerated Reinforcement Learning. In *1st International Workshop on Hybrid Control of Autonomous System*, pages 49–55, 2009.

[68] Carlos E. Mariano and Eduardo F. Morales. DQL: A New Updating Strategy for Reinforcement Learning Based on Q-Learning. In L. De Raedt and P. Flach, editors, *Machine Learning: ECML*, volume 2167, pages 324–335. Springer, Berlin, Heidelberg, 2001.

[69] Marco Dorigo and Luca Maria Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, apr 1997.

[70] Marco Dorigo, Mauro Birattari, and Thomas Stŭtzle. Ant Colony Optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39, 2006.

[71] Marco Dorigo, Vittotio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26(1):29–41, feb 1996.

[72] Marco Dorigo and Luca Maria Gambardella. Ant colonies for the travelling salesman problem. *Bio Systems*, 43(2):73–81, 1997.

[73] Daniel Merkle, Martin Middendorf, and Hartmut Schmeck. Ant Colony Optimization for Resource-Constrained Project Scheduling. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 6(4):333–346, 2002.

[74] Ronald P.S. Mahler. Multitarget Bayes Filtering via First-Order Multitarget Moments. *IEEE Transactions on Aerospace and Electronic Systems*, 39(4):1152–1178, 2003.

[75] Y. C. Ho and R. C. K. Lee. A Bayesian approach to problems in stochastic estimation and control. *IEEE Transactions on Automatic Control*, 9(4):333–339, oct 1964.

[76] Ronald P.S. Mahler. A Theoretical Foundation for the Stein-Winter "Probability Hypothesis Density (PHD)" Multitarget Tracking Approach. In *The 2000 MSS National Symposium on Sensor and Data Fusion*, pages 99–117, San Antonio, TX, 2000. Army Research Office Alexandria Va.

[77] Pushpa L. Gupta and R.D. Gupta. Sample Size Determination in Estimating a Covariance Matrix. *Computational Statistics and Data Analysis*, 5(3):185–192, 1987.

[78] Marcelo G.S. Bruno. *Sequential Monte Carlo Methods for Nonlinear Discrete-Time Filtering*, volume 6. Morgan & Claypool, jan 2013.

[79] Dan Crisan and Arnaud Doucet. A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on Signal Processing*, 50(3):736–746, mar 2002.

[80] Ronald J. Tallarida. *Pocket Book of Integrals and Mathematical Formulas*. Chapman & Hall/CRC, 3rd edition, 1999.

A. JACOBIAN MATRICES OF MEASUREMENT SPACES

The following sections provide the elements of the Jacobian matrices for the coordinate systems explained in section 3.2.3. Only the partial derivatives with respect to the position coordinates are included; the partial derivatives with respect to the velocity coordinates are all zero because there is no direct relationship between the velocity coordinates and the measurement angles.

To construct the Jacobian matrices, the expected position of the object is used, which provides an expected range; the EKF provides the expected, or most likely, state of the object. Equations A.1-A.3 provide the object state and the range vector definitions which are used in the following sections.

$$\mathbf{X} = [x \ y \ z \ \dot{x} \ \dot{y} \ \dot{z}]^T \tag{A.1}$$

$$\bar{\rho} = \begin{bmatrix} (x - x_{obs}) \\ (y - y_{obs}) \\ (z - z_{obs}) \end{bmatrix} = \begin{bmatrix} \rho_x \\ \rho_y \\ \rho_z \end{bmatrix} \tag{A.2}$$

$$\rho = \sqrt{\rho_x^2 + \rho_y^2 + \rho_z^2} \tag{A.3}$$

The equations for the angles of each of the measurement spaces (sections 2.2.1 and 2.2.2) may be expressed in terms of the range vector instead of the pointing vector. To generate the Jacobians, the range vector formulation of the measurement angles are used. To show this, equations 2.16-2.17 are expressed in terms of the range vector results in equations A.4-A.5. The other measurement angles may be similarly expressed in terms of the range vector components.

$$\alpha = \tan^{-1}\left(\frac{(y - y_{obs})}{(x - x_{obs})}\right) \tag{A.4}$$

$$\delta = \sin^{-1}\left(\frac{(z - z_{obs})}{\rho}\right) \tag{A.5}$$

The partials that form the Jacobian matrices are with respect to the position coordinates of the expected state (x, y, z); The position coordinates of the observer $(x_{obs}, y_{obs}, z_{obs})$ are treated as constants. From the differentiation rules for $\sin^{-1}$ [80], the partial of equation A.5 with respect to z, results in the following:

$$\frac{\partial \delta}{\partial z} = \frac{1}{\sqrt{1 - \left(\frac{(z - z_{obs})}{\rho}\right)^2}} \frac{\partial}{\partial z}\left(\frac{(z - z_{obs})}{\rho}\right)$$

$$= \frac{\rho}{\sqrt{(x - x_{obs})^2 + (y - y_{obs})^2}} \cdot \frac{\rho - (z - z_{obs})^2 \cdot \rho^{-1}}{\rho^2}$$

$$= \frac{\rho}{\sqrt{(x - x_{obs})^2 + (y - y_{obs})^2}} \cdot \frac{\rho^2 - (z - z_{obs})^2}{\rho^3}$$

$$= \frac{1}{\sqrt{(x - x_{obs})^2 + (y - y_{obs})^2}} \cdot \frac{(x - x_{obs})^2 + (y - y_{obs})^2}{\rho^2}$$

$$= \frac{\sqrt{(x - x_{obs})^2 + (y - y_{obs})^2}}{\rho^2}$$

Similar derivations are required for the remaining partials in the EVE system, as well as the other measurement spaces. The following sections include the partials of each ground-based and space-based measurement angles with respect to the expected range vector from the observer to the object.

A.1 Ground-Based Sensor

A.1.1 Equatorial Vernal Equinox System

The full set of partials for the EVE coordinate system, based on the angles in equations A.4-A.5, are provided in terms of the range vector components. The right ascension is independent of the object's ρ_z component, while the declination is dependent on all of the components.

$$\frac{\partial \alpha}{\partial x} = \frac{-\rho_y}{\rho_x^2 + \rho_y^2} \tag{A.6}$$

$$\frac{\partial \alpha}{\partial y} = \frac{\rho_x}{\rho_x^2 + \rho_y^2} \tag{A.7}$$

$$\frac{\partial \alpha}{\partial z} = 0 \tag{A.8}$$

$$\frac{\partial \delta}{\partial x} = \frac{-\rho_x\ \rho_z}{(\rho_x^2 + \rho_y^2 + \rho_z^2)\sqrt{\rho_x^2 + \rho_y^2}} \tag{A.9}$$

$$\frac{\partial \delta}{\partial y} = \frac{-\rho_y\ \rho_z}{(\rho_x^2 + \rho_y^2 + \rho_z^2)\sqrt{\rho_x^2 + \rho_y^2}} \tag{A.10}$$

$$\frac{\partial \delta}{\partial z} = \frac{\sqrt{\rho_x^2 + \rho_y^2}}{\rho^2} \tag{A.11}$$

A.1.2 Local Meridian Equatorial System

The angles for this system are given in terms of the range vector by equations A.12-A.13.

$$\tau = \tan^{-1}\left(\frac{\rho_x \sin\theta - \rho_y \cos\theta}{\rho_x \cos\theta + \rho_y \sin\theta}\right) \tag{A.12}$$

$$\delta = \sin^{-1}\left(\frac{(z - z_{obs})}{\rho}\right) \tag{A.13}$$

Equation A.12 includes the local mean sidereal time (θ), however, through the process of taking the partial derivatives θ drops out of all the terms. The resulting partials that make up the Jacobian matrix are:

$$\frac{\partial\tau}{\partial x} = \frac{\rho_y}{\rho_x^2 + \rho_y^2} \tag{A.14}$$

$$\frac{\partial\tau}{\partial y} = \frac{-\rho_x}{\rho_x^2 + \rho_y^2} \tag{A.15}$$

$$\frac{\partial\tau}{\partial z} = 0 \tag{A.16}$$

$$\frac{\partial\delta}{\partial x} = \frac{-\rho_x\ \rho_z}{\rho^2 \cdot \sqrt{\rho_x^2 + \rho_y^2}} \tag{A.17}$$

$$\frac{\partial\delta}{\partial y} = \frac{-\rho_y\ \rho_z}{\rho^2 \cdot \sqrt{\rho_x^2 + \rho_y^2}} \tag{A.18}$$

$$\frac{\partial\delta}{\partial z} = \frac{\sqrt{\rho_x^2 + \rho_y^2}}{\rho^2} \tag{A.19}$$

A.1.3 Local Meridian Local Horizon System

The LMLH system contains θ and the observer latitude (ϕ) in angle definitions as shown in equations A.20-A.21.

$$\beta = \tan^{-1}\left(\frac{\rho_x \sin\theta - \rho_y \cos\theta}{\rho_x \cos\theta \sin\phi + \rho_y \sin\theta \sin\phi - \rho_z \cos\phi}\right) \tag{A.20}$$

$$h = \sin^{-1}\Big(\rho_x \cos\theta \cos\phi + \rho_y \sin\theta \cos\phi + \rho_z \sin\phi\Big) \tag{A.21}$$

In this system, the trigonometric terms in θ and ϕ remain after the partial derivatives are taken, resulting in the following terms for the Jacobian:

$$\frac{\partial\beta}{\partial x} = \frac{\rho_y \sin\phi - \rho_z \sin\theta \cos\phi}{(\rho_x \cos\theta \sin\phi + \rho_y \sin\theta \sin\phi - \rho_z \cos\phi)^2 \cdot (\rho_x \sin\phi - \rho_y \cos\phi)^2} \tag{A.22}$$

$$\frac{\partial\beta}{\partial y} = \frac{\rho_z \cos\theta \cos\phi - \rho_x \sin\phi}{(\rho_x \cos\theta \sin\phi + \rho_y \sin\theta \sin\phi - \rho_z \cos\phi)^2 \cdot (\rho_x \sin\phi - \rho_y \cos\phi)^2} \tag{A.23}$$

$$\frac{\partial\beta}{\partial z} = \frac{\rho_x \sin\theta \cos\phi - \rho_y \cos\theta \cos\phi}{(\rho_x \cos\theta \sin\phi + \rho_y \sin\theta \sin\phi - \rho_z \cos\phi)^2 \cdot (\rho_x \sin\phi - \rho_y \cos\phi)^2} \tag{A.24}$$

$$\frac{\partial h}{\partial x} = \frac{(\rho_y^2 + \rho_z^2)\cos\theta \cos\phi - (\rho_x\ \rho_y \sin\theta \cos\phi + \rho_x\ \rho_z \sin\phi)}{\rho^2 \cdot \sqrt{\rho^2 - (\rho_x \cos\theta \cos\phi + \rho_y \sin\theta \cos\phi - \rho_z \sin\phi)^2}} \tag{A.25}$$

$$\frac{\partial h}{\partial y} = \frac{(\rho_y^2 + \rho_z^2)\cos\theta \cos\phi - (\rho_x\ \rho_y \sin\theta \cos\phi + \rho_x\ \rho_z \sin\phi)}{\rho^2 \cdot \sqrt{\rho^2 - (\rho_x \cos\theta \cos\phi + \rho_y \sin\theta \cos\phi - \rho_z \sin\phi)^2}} \tag{A.26}$$

$$\frac{\partial h}{\partial z} = \frac{(\rho_y^2 + \rho_z^2)\cos\theta \cos\phi - (\rho_x\ \rho_y \sin\theta \cos\phi + \rho_x\ \rho_z \sin\phi)}{\rho^2 \cdot \sqrt{\rho^2 - (\rho_x \cos\theta \cos\phi + \rho_y \sin\theta \cos\phi - \rho_z \sin\phi)^2}} \tag{A.27}$$

A.2 Space-Based Sensor

A.2.1 Satellite Orbit Radial System

These partials are based on equations 2.24 and 2.25, where the pointing vector components are again replaced by the range vector components. Like the LMLH system for the ground-based sensor, the angles in the SOR include a number of additional trigonometric terms; in this case, the terms are in the angles Ω, λ, and i. The resulting partial derivatives for the Jacobian matrix are:

$$\frac{\partial \vartheta_s}{\partial x} = \frac{-\cos i \cdot \rho_y - \cos\Omega \sin i \cdot \rho_z}{D1} \tag{A.28}$$

$$\frac{\partial \vartheta_s}{\partial y} = \frac{\cos i \cdot \rho_x - \sin\Omega \sin i \cdot \rho_z}{D1} \tag{A.29}$$

$$\frac{\partial \vartheta_s}{\partial z} = \frac{\cos\Omega \sin i \cdot \rho_x + \sin\Omega \sin i \cdot \rho_y}{D1} \tag{A.30}$$

$$\frac{\partial \varphi_s}{\partial x} = \frac{\sin\Omega \sin i \cdot (\rho_y^2 + \rho_z^2) + \cos\Omega \sin i \cdot \rho_x\ \rho_y - \cos i \cdot \rho_x\ \rho_z}{D2} \tag{A.31}$$

$$\frac{\partial \varphi_s}{\partial y} = \frac{-\cos\Omega \sin i \cdot (\rho_x^2 + \rho_z^2) - \sin\Omega \sin i \cdot \rho_x\ \rho_y - \cos i \cdot \rho_y\ \rho_z}{D2} \tag{A.32}$$

$$\frac{\partial \varphi_s}{\partial z} = \frac{\cos i \cdot (\rho_x^2 + \rho_y^2) - \sin\Omega \sin i \cdot \rho_x\ \rho_z + \cos\Omega \sin i \cdot \rho_y\ \rho_z}{D2} \tag{A.33}$$

where:

$$\begin{aligned} D1 =& (\cos^2\Omega + \sin^2\Omega \cos^2 i) \cdot \rho_x^2 + \ldots \\ & (\sin^2\Omega + \cos^2\Omega \cos^2 i) \cdot \rho_y^2 + (\sin^2 i) \cdot \rho_z^2 + \ldots \\ & 2(\cos\Omega \sin\Omega \sin^2 i) \cdot \rho_x\ \rho_y - \ldots \\ & 2(\sin\Omega \cos i \sin i) \cdot \rho_x\ \rho_z + \ldots \\ & 2(\cos\Omega \cos i \sin i) \cdot \rho_y\ \rho_z \end{aligned} \tag{A.34}$$

$$D2 = \rho^2 \cdot \sqrt{\rho^2 - (\sin\Omega \sin i \cdot \rho_x - \cos\Omega \sin i \cdot \rho_y + \cos i \cdot \rho_z)^2} \tag{A.35}$$

A.2.2 Satellite Meridian Equatorial System

Finally, the SME angles are defined in equations 2.28 and 2.29 with the angle θ_{sbs} given by equation 2.27; again the pointing vector is replaced by the range vector for the calculation of the partial derivatives. As discussed in Section 2.2.2, the definition is similar to the ground-based LME system, and partials with respect to the object position coordinates result in similar elements for the measurement Jacobian.

$$\frac{\partial \tau_s}{\partial x} = \frac{\rho_y}{\rho_x^2 + \rho_y^2} \tag{A.36}$$

$$\frac{\partial \tau_s}{\partial y} = \frac{-\rho_x}{\rho_x^2 + \rho_y^2} \tag{A.37}$$

$$\frac{\partial \tau_s}{\partial z} = 0 \tag{A.38}$$

$$\frac{\partial \delta_s}{\partial x} = \frac{-\rho_x \ \rho_z}{\rho^2 \cdot \sqrt{\rho_x^2 + \rho_y^2}} \tag{A.39}$$

$$\frac{\partial \delta_s}{\partial y} = \frac{-\rho_y \ \rho_z}{\rho^2 \cdot \sqrt{\rho_x^2 + \rho_y^2}} \tag{A.40}$$

$$\frac{\partial \delta_s}{\partial z} = \frac{\sqrt{\rho_x^2 + \rho_y^2}}{\rho^2} \tag{A.41}$$

B. ADDITIONAL OPTIMIZER RESULTS

In Chapter 5, the results were limited to the information that was relevant for understanding the results. This appendix provides some of the additional data that was left out of Chapter 5.

B.1 Propagation Times

In Chapter 5, the computation times reported were only for the generation of the solutions by the optimization algorithms. However, there were other calculations required for each sensor and scenario (e.g. calculation of the ground-based sensor position, propagation of the space-based sensor, and calculation of the CDF values). Table B.1 presents examples of the time required to perform these other calculations for each of the sensors and scenarios investigated in Chapter 5.

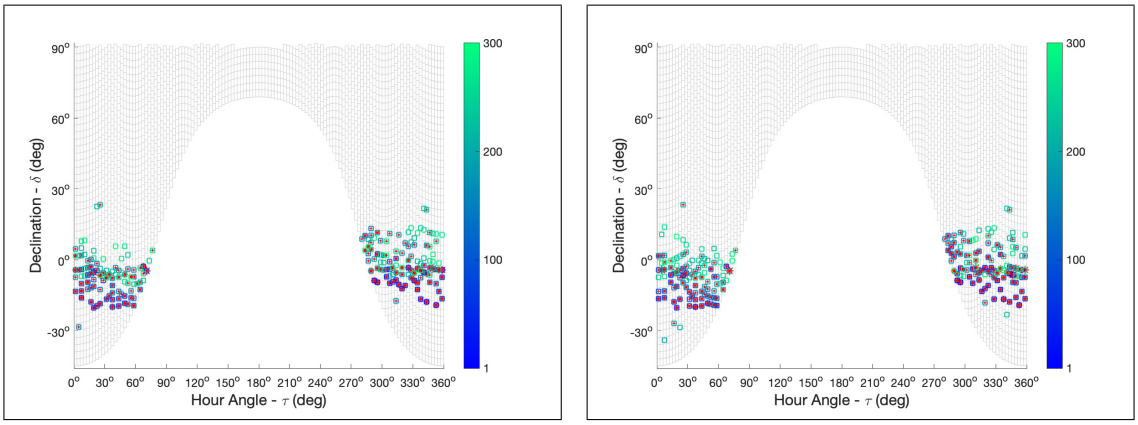
Table B.1. The computation times required for the scenarios, but that are not part of the optimization algorithm computations.

Ground-based	Computation Time
Absolute Knowledge	14 mins, 20 secs
Uncertain States	52 mins, 44 secs
Uncertain States, p_d	8 hrs, 15 mins
Space-base	-
Absolute Knowledge	13 mins, 39 secs
Uncertain States	1 hr, 2 mins
Uncertain States, p_d	2 hrs, 41 mins

The significant time increase for the ground-based sensor when the p_d is included is due to the need to calculate the seeing parameter for every grid field where the CDF values are non-zero. This means that for every object that could be observed by the sensor, at every time that an observation is to be scheduled, the multiple seeing parameters must be determined. Because the space-based sensor is outside the atmosphere, this extra calculation is not required and the increased computation time required for calculating p_d is much less.

B.2 Ground-based Sensor Results

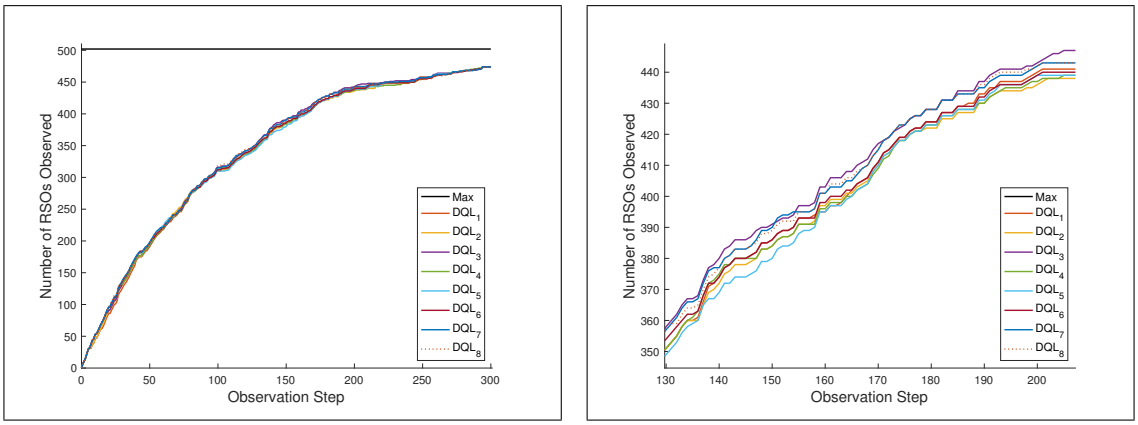
For the absolute knowledge case, the ACO algorithm generated four solutions. Figure B.1 shows the number of objects seen by viewing direction for the ACO_2 and ACO_3 solutions. These plots are very similar to the plots presented in Figure 5.3 in section 5.2.1.



(a) ACO_2. (b) ACO_3.

Figure B.1. Grid fields, and associated values (# of RSOs), for two solutions generated by the ACO algorithm when the mean states are assumed to be the true states.

In the uncertain states scenario, with $p_d = 1$, the DQL algorithm generate a total of eight solutions. Figure B.2 shows the growth rates of the eight solutions, along with a close up of the growth rates to highlight the uniqueness of the different solutions.

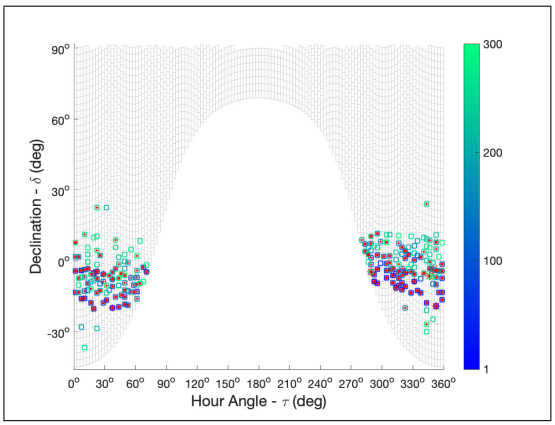


(a) DQL Growth Rates.

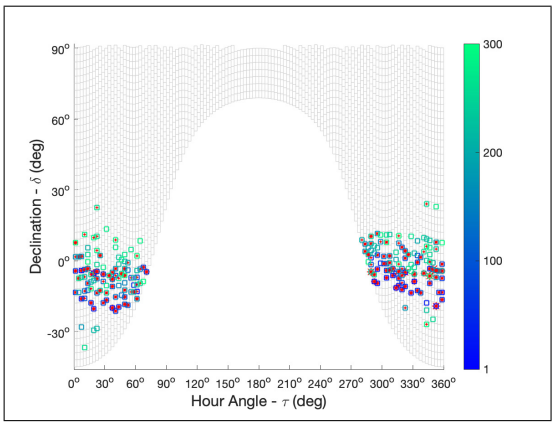
(b) Close Up.

Figure B.2. Growth rates for RSOs observed by each DQL solutions are distinguished by color and line style. The close up view on the right shows the solutions diverge at times due to the unique sets of viewing directions assigned.

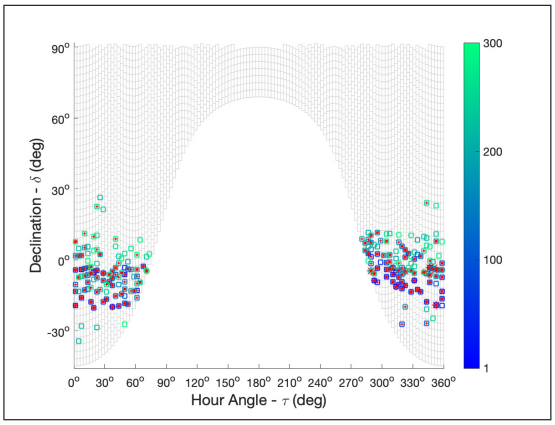
Figure B.3 shows the number of objects observed by viewing direction for the five solutions that were not included in Figure 5.7 in section 5.2.1; the plots are similar to those presented previously.



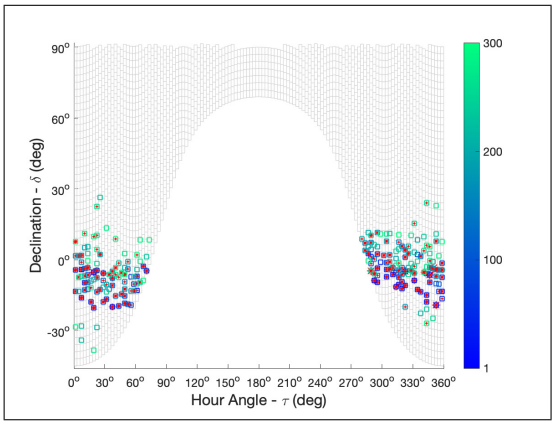
(a) DQL_4.



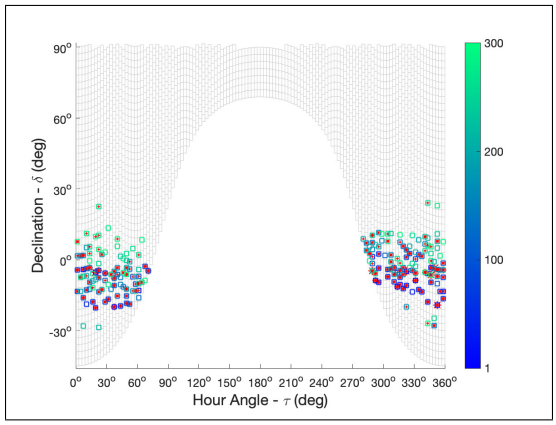
(b) DQL_5.



(c) DQL_6.



(d) DQL_7.



(e) DQL_8.

Figure B.3. Grid fields, and associated # of RSOs, for five solutions generated by the DQL algorithm when uncertainty is considered in the optimization.

B.3 Space-based Sensor Results

In the scenario with uncertain state estimates, the ACO algorithm generated two solutions. Figure B.4 presents the number of objects observed by viewing direction for the ACO_2 solution. The plot is similar to the ACO_1 plot found in section 5.2.2, Figure 5.13.

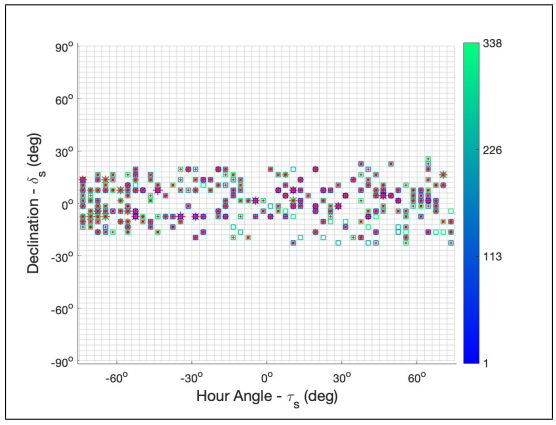


Figure B.4. Grid fields, and associated # of RSOs, for the second ACO solution generated when uncertainty is considered in the optimization.

Unique Grid Fields

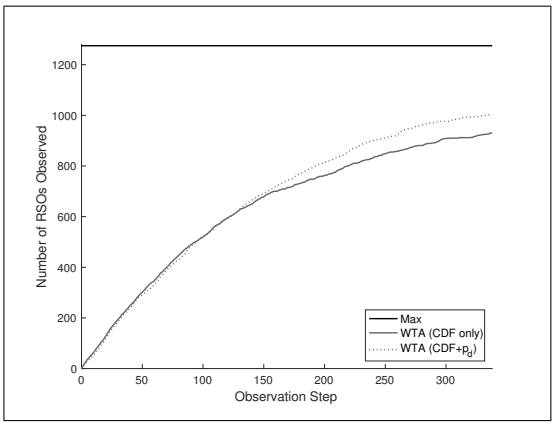
The number of unique grid fields chosen in a given solution is not directly linked with how well a solution will perform. This was determine during the ground-based results in Section 5.2.1, but the same is true for the space based results. Table B.2 shows the maximum number of unique grid fields selected and the number of RSOs observed for each optimizer solution, in each of the three scenarios, for the space-based sensor. Again, the number of grid fields is not clearly linked with better solutions; in the second scenario, the ACO selects 21 more unique grid fields than the greedy though they observe the same number of objects, while in the third scenario the greedy and WTA only differ by two unique grid fields but the WTA observes 18 more RSOs.

Growth Rate Comparisons

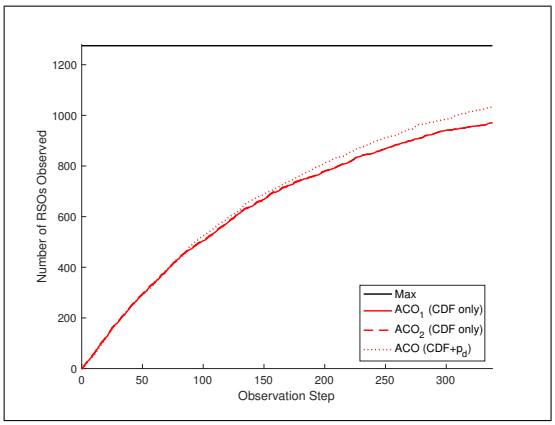
Table B.2. The number of unique grid fields used for the viewing directions is compared for each algorithm in all three scenarios for the space-based sensor. For DQL, only the maximum number of grid fields are shown, but the other solutions are within seven of the displayed value.

	Absolute Knowledge		Uncertain States		$p_d \neq 1$	
	# of $h_{g,f}$	# observed	# of $h_{g,f}$	# observed	# of $h_{g,f}$	# observed
Greedy	169	1180	197	822	209	816
WTA	226	1192	210	803	207	834
ACO	252	1189	218	822	230	822
DQL	170	1179	254	865	235	840

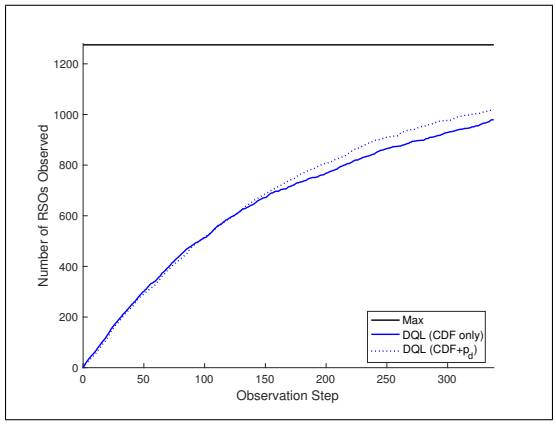
The space-based sensor solutions found more objects when including the p_d in the optimization then when only the CDF values were used. Figure 5.15 in section 5.2.2 shows that the greedy growth rates stays consistent longer when considering p_d. Figure B.5 shows the same is true for the other optimizers. In the space-based sensor scenario, the p_d plays a much more important role than in the ground-based sensor scenario.



(a) WTA.



(b) ACO.



(c) DQL.

Figure B.5. Comparing the WTA, ACO, and DQL growth rates for RSOs observed: the solid lines use only the CDF values, the dashed lines use the CDF and p_d values.

C. PREDICTED MEASUREMENT PROBABILITY

C.1 Jacobian for Full State Estimate

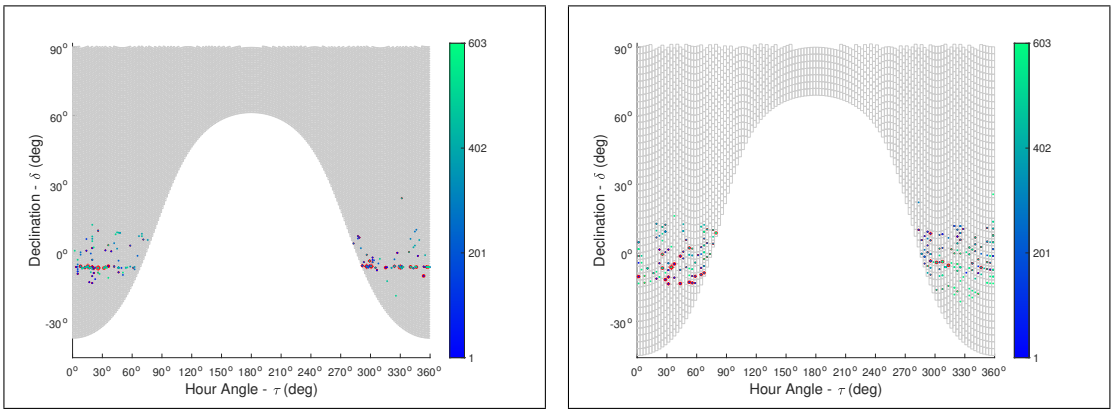
In Section 6.4.1, the full state estimate for the PMP object is performed in the spherical coordinates of the LME coordinate frame, for the ground-based sensor. Using this frame, which is the same as the frame of the measurements, allows the four required coordinates to be estimated directly. However, the objects' state and covariance are defined in the Cartesian ECI frame, and must be transformed into the spherical frame. The state is transformed by equations 6.12-6.17, while the transformation of the covariance into the spherical space requires the 6×6 Jacobian matrix, $\mathbf{H}$, given by:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial \rho}{\partial x} & \frac{\partial \rho}{\partial y} & \frac{\partial \rho}{\partial z} & \frac{\partial \rho}{\partial \dot{x}} & \frac{\partial \rho}{\partial \dot{y}} & \frac{\partial \rho}{\partial \dot{z}} \\ \frac{\partial \tau}{\partial x} & \frac{\partial \tau}{\partial y} & \frac{\partial \tau}{\partial z} & \frac{\partial \tau}{\partial \dot{x}} & \frac{\partial \tau}{\partial \dot{y}} & \frac{\partial \tau}{\partial \dot{z}} \\ \frac{\partial \delta}{\partial x} & \frac{\partial \delta}{\partial y} & \frac{\partial \delta}{\partial z} & \frac{\partial \delta}{\partial \dot{x}} & \frac{\partial \delta}{\partial \dot{y}} & \frac{\partial \delta}{\partial \dot{z}} \\ \frac{\partial \dot{\rho}}{\partial x} & \frac{\partial \dot{\rho}}{\partial y} & \frac{\partial \dot{\rho}}{\partial z} & \frac{\partial \dot{\rho}}{\partial \dot{x}} & \frac{\partial \dot{\rho}}{\partial \dot{y}} & \frac{\partial \dot{\rho}}{\partial \dot{z}} \\ \frac{\partial \dot{\tau}}{\partial x} & \frac{\partial \dot{\tau}}{\partial y} & \frac{\partial \dot{\tau}}{\partial z} & \frac{\partial \dot{\tau}}{\partial \dot{x}} & \frac{\partial \dot{\tau}}{\partial \dot{y}} & \frac{\partial \dot{\tau}}{\partial \dot{z}} \\ \frac{\partial \dot{\delta}}{\partial x} & \frac{\partial \dot{\delta}}{\partial y} & \frac{\partial \dot{\delta}}{\partial z} & \frac{\partial \dot{\delta}}{\partial \dot{x}} & \frac{\partial \dot{\delta}}{\partial \dot{y}} & \frac{\partial \dot{\delta}}{\partial \dot{z}} \end{bmatrix} \tag{C.1}$$

where $\mathbf{H}$ is deterministic and evaluated at the mean state of the object for the given observation step. The transformation is performed using equation 6.18, and $\mathbf{Pz}$ is the covariance in terms of the spherical coordinates.

C.2 Additional ACO Viewing Direction Plots

The viewing direction plots for the additional ACO solutions are shown Figures C.1-C.5. As discussed in section 6.5.2, Table 6.5, the number of unique grid fields varies across the solutions. However, the general distributions of the viewing directions are similar across all of the ACO solutions.

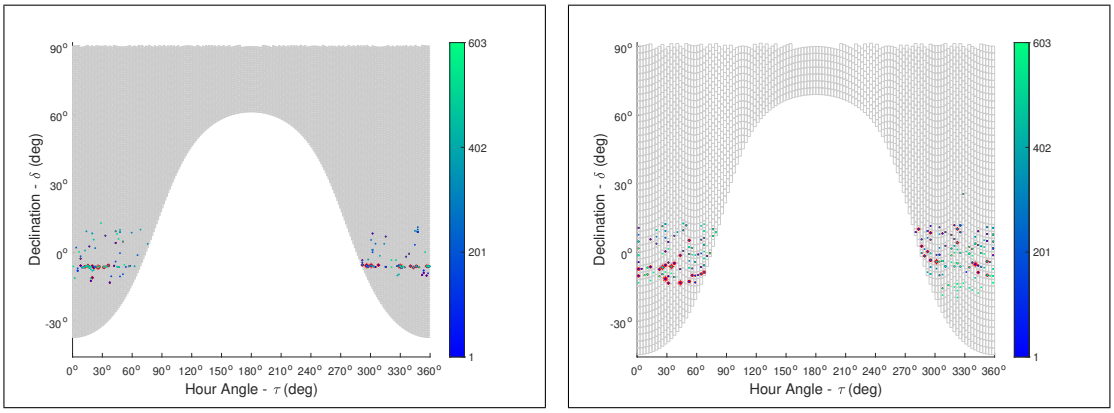


(a) PACO (b) POGS

Figure C.1. The ACO_{3x20} (1) viewing directions and numbers of objects seen at each step are shown for each of the sensors in their respective measurement grid.

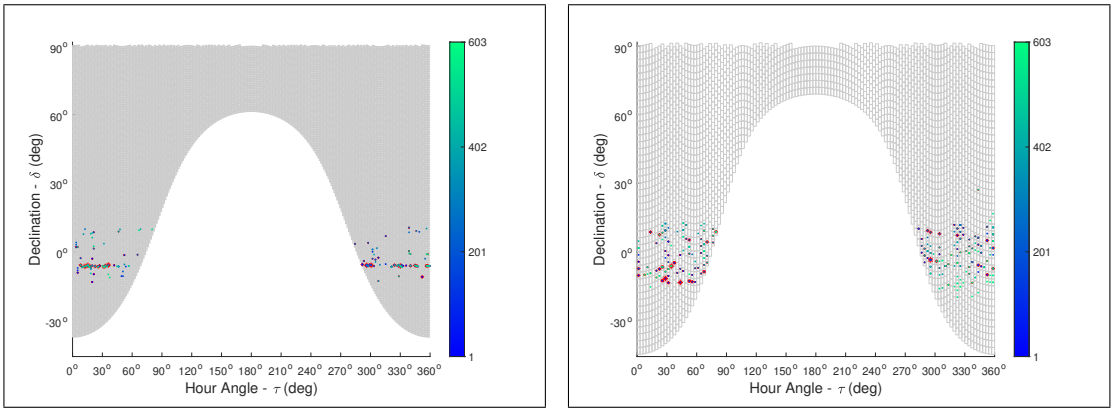
C.3 Computation Times for PMP Solutions

The addition of the PMP calculation to the optimization algorithms results in a significant increase in the computation time required. Additionally, the most of the calculations were performed for a two sensor scenario, so the number of propagations, CDF calculations, and viewing directions assignments were essentially doubled. Additionally, the PMP generation, PMP propagation, and calculation of the CDF values for the PMP were added to the solution generation computations. Because the focus of Chapter 6 was the efficacy of the PMP method, and the author knew the com-



(a) PACO (b) POGS

Figure C.2. The ACO_{4x20} (1) viewing directions and numbers of objects seen at each step are shown for each of the sensors in their respective measurement grid.

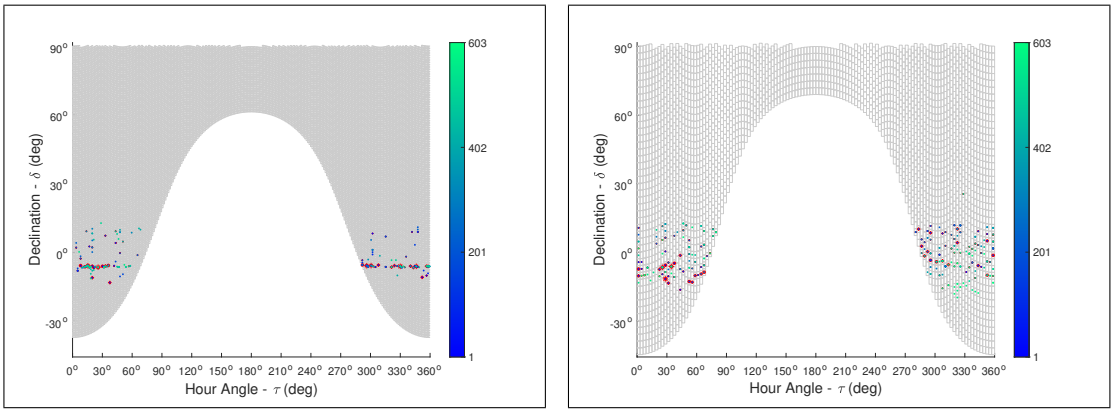


(a) PACO (b) POGS

Figure C.3. The ACO_{4x20} (2) viewing directions and numbers of objects seen at each step are shown for each of the sensors in their respective measurement grid.

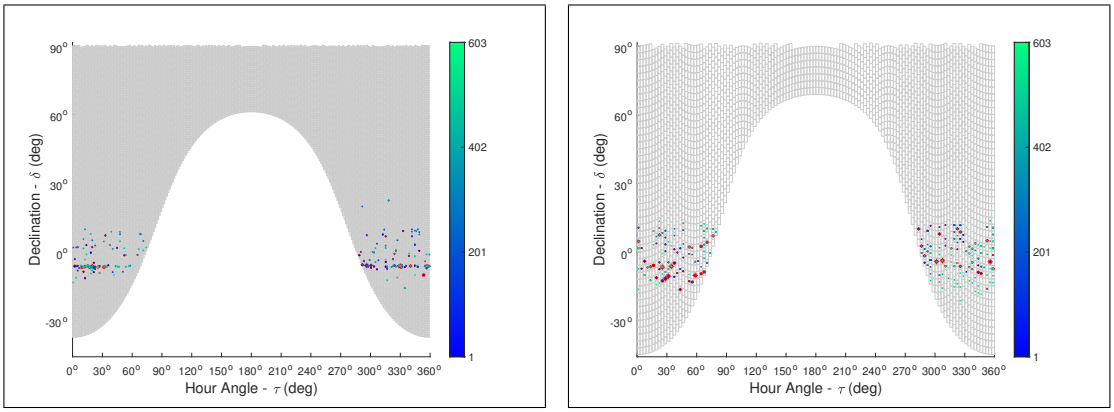
putational burden would be high, the computation times were not included in the analysis.

The computation times were recorded, and are contained in Table C.1. The first two rows show the single sensor, greedy optimizer computation times. While the



(a) PACO (b) POGS

Figure C.4. The ACO_{5x20} viewing directions and numbers of objects seen at each step are shown for each of the sensors in their respective measurement grid.



(a) PACO (b) POGS

Figure C.5. The ACO_{3x50} viewing directions and numbers of objects seen at each step are shown for each of the sensors in their respective measurement grid.

PACO sensor generated less observations (250 vs. 353 for POGS), having smaller grid fields causes the number of CDF calculations at every step is much higher than for the POGS sensor.

Table C.1. Looking at the computation times for the solutions generated during the PMP analysis shows that the method, and using two sensors, generates significant increases in computational complexity.

	Computation Time
PACO only	7 hrs, 0 mins
POGS only	4 hrs, 47 mins
Greedy	9 hrs, 44 mins
ACO_{3x20} (1)	37 hrs, 58 mins
ACO_{3x20} (2)	38 hrs, 22 mins
ACO_{4x20} (1)	51 hrs, 2 mins
ACO_{4x20} (2)	49 hrs, 22 mins
ACO_{5x20}	63 hrs, 53 mins
ACO_{3x50}	51 hrs, 13 mins

D. OPTIMIZER TUNING METHODS

The Ant Colony Optimization and the Distributed Q-learning algorithms require tuning in order to generate solutions. This appendix describes the tuning used in this work for each of the algorithms. The tuning of these algorithms was performed in order to produce good solutions for comparison with the other algorithms. The author does not assume that the values determine represent those which guarantee the best solutions from each optimizer. The time required to perform the tuning of these algorithms is an additional parameter that must be considered in the evaluation of the algorithms' efficiency.

D.1 Ant Colony Tuning

As discussed in section 4.4.1, the ACO algorithm requires the user to define the following values:

1. Number of agents
2. Number of iterations
3. Heuristic importance
4. Pheromone importance
5. Pheromone evaporation rate

D.1.1 Single Sensor Simulations

These parameters will be introduced in terms of the tuning for the single sensor simulations. While the values used for the ground-based and space-based sensors are different, the steps taken to tune the algorithm are the same. Tuning for the two

sensor simulations used in the PMP development are slightly different and will be discussed in section D.1.2, below.

Because the ACO agents probabilistically make choices at every step, the random number generator in MATLAB is seeded for these simulations. This ensures that the same solution is generated if all of the parameters are held constant. This allows for the effects of changing each parameter to be isolated, as long as only one parameter is changed.

Number of Agents

The number of agents is entirely user defined, with no references found in the literature regarding how to choose this value. For the initial single sensor scenarios (see Chapter 5), the author chose to relate the number of agents to the number of non-zero valued grid fields at the first observation time of the scenarios. In each case nearly 2,000 agents were assigned; this large number of agents allows the algorithm to explore a large number of possible solutions through the problem.

The agents are not directly controlled within the algorithm, but are allowed to sample any viewing direction choice based on the weights provided by equation 4.28. Excessive sampling will lead to extremely large pheromone values that would skew the solutions to that choice of viewing direction; i.e., while the agent choices are probabilistic, if they tend towards greedy choices in the first few iterations, the resulting pheromone values will encourage limited exploration in later iterations. In order to avoid one viewing direction choice becoming overly sampled, this author chose to limit the number of agents allowed to choose a given grid field at each time by introduce the parameter K_{lim}. K_{lim} forces the agents to explore more than just the highly weighted grid fields. In these scenarios, the number of agents is larger than the number of non-zero grid fields, so the value of K_{lim} must not be too small, or the agents will run out of choices; however, it cannot be set too large or its effect will be lost. In this work, the author chose to set the value to $K_{lim} = 10$, based on the number of agents used in the absolute knowledge cases (sections 5.2.1 and 5.2.2); the

cases with uncertainty are less restrictive (K_{lim} could be larger), but the same value was chosen.

Number of Iterations

The number of iterations may be set to achieve some maximum value, to determine when solutions stop improving, or based on a user defined maximum number. In this work, the number of iterations was set to limit the number of iterations; the maximum value expected is not easily determined, and it is possible for the best solutions in early iterations to be worse than a previous iteration. Holding all other parameters constant, the number of iterations was adjusted for the absolute knowledge case with the ground-based sensor. The author found that eight iterations provided the algorithm enough time to generate good solutions, without forcing the computation times to become too long. This value was then set for all of the other solutions in both the ground-based and space-based sensor simulations.

Heuristic and Pheromone Importance

The heuristic (β) and pheromone (α) importance values allow the algorithm to be tuned to balance the influence of each of these inputs. Initially, these values are set by the author and solutions are generated. The values are then adjusted individually and the solutions are recalculated to determine the effect. Through this evaluation it was determined that $\alpha = 1$ was the best choice for the pheromone importance; this is also a commonly used value in the literature for ACO [69, 71]. The value for β was found to differ in each simulation. The author used increments of ± 0.1 to adjust β until a "best" solution was generated.

Pheromone Evaporation Rate - ρ

The final parameter to set is the pheromone evaporation rate, $0 < \rho < 1$; this reduces the pheromone from older solutions before the newer solutions apply their pheromone. If ρ is too high, the pheromone values can become large and may dominate the solution; if ρ is too small, the pheromone may evaporate too quickly and the solutions will be dominated by the heuristic.

The author initially set the evaporation rate to $\rho = 1/2$ [69]. After deciding on the pheromone importance value of $\alpha = 1$, the ρ value was adjusted; the values tested were $1/2,\ 1/3,\ 1/4,\ 1/5$. It was determined that $\rho = 1/4$ produced good solutions in the scenarios simulated for this work.

D.1.2 Two Sensor Simulations

In the two sensor simulations, the parameters $\alpha,\ \beta,$ and ρ are set based on previously determined values for the single ground-based sensor case. These simulations are assessing how well the PMP method was working within the ACO algorithm, and the previously determined values are assumed to be good enough for that evaluation. Additional tuning, specific to this problem, is expected to be able to generate better solutions.

The number of agents used is set to a fixed number instead of being dependent on the number of non-zero grid fields; due to the different field of view sizes for the two sensors, the number of grid fields are different. There are also more than twice the number of viewing directions to assign between the two sensors. The initial values for the iterations and agents were set to small values (three iterations and twenty agents), while the PMP code was being developed; the lower values ensured that any errors in the code could be identified more quickly than if larger values were used (e.g. eight iterations of 2000 agents).

When the solution generated with these smaller values still resulted in better solutions than the greedy optimizer, it was decided to assess the probabilistic nature of the ACO algorithm. Therefore, the random number generator used for generating the agent choices was not seeded, so that each run generates a unique solution. Additionally, four different iteration-agent parameter sets were tested, $[3x20,\ 4x20,\ 5x20,\ 3x50]$.

The results of the two sensor simulation indicate that good solutions may be possible in the single sensor cases with less agents and/or less iterations. However, this was not explicitly tested in this work.

D.2 Distributed Q-Learning

As discussed in section 4.4.2, the DQL algorithm requires the user to define the following values:

1. Number of agents
2. Number of iterations
3. Learning Parameter
4. Discount Parameter
5. Exploration Parameter

The DQL was only applied to the single sensor simulations.

Number of Agents

As with the ACO, the number of agents can be chosen by the user. The number of agents is directly related to how many of the state-action pairs are evaluated during each iteration. An optimal solution is only generated if every state-action pair is visited infinitely many times; larger numbers of agents provide a better approximation of infinitely visiting every state-action pair. The number of agents is also directly tied to the computation time required to generate a solution (i.e. more agents, longer computations).

For this work, the number of agents is set to 2,000 for the ground-based sensor and 3,000 for the space-based sensor; these values are close to the number of agents used by the ACO algorithm in the same scenarios. Attempts were made to use less agents, but the solutions generated did not produce good results. Once the agents were increased to these values the solutions improved.

Number of Iterations

Again, like the ACO, the number of iterations can be set based on a desired value, the solutions no longer improving, or a set number of iterations. The author chose the latter as the maximum value is unknown and solutions for one iteration will often do worse than a previous iteration; this is true with DQL because the choices made during exploration are highly random. Additionally, the number of iterations also plays a part in trying to evaluate every state-action pair infinitely many times. The numbers of iterations tested started small, 50, and were increased until the final value of 500 was reached. At that point the solutions generated were doing well, and the computation times were not extremely long.

Learning and Discount Parameters

The learning (α) and discount (γ) parameters are tied to how quickly the values of the state-action pairs grow within the value function. Both values are set between zero and one, but the lower the value the slower the value function changes, which can lead to long convergence times. However, setting these values too high can result in premature convergence. The author chose to begin with values of $\alpha = 1$ and $\gamma = 0.5$ and adjust them individually until "good" solutions were generated. In doing so, many combinations of the two parameters were tested, before the final values were selected; those values, $\alpha = 0.9$ and $\gamma = 0.7$, were used for each of the simulations.

Exploration Parameter

The exploration parameter, ϵ, determines how likely the agents are to try actions that are not the maximum value choice. This value is generally set to a small value, so the author chose to follow this same practice [67, 68]. The value was set to $\epsilon = 0.1$ for the ground-based sensor, while the setting it to $\epsilon = 0.05$ was found to work better for the space-based sensor.

The author also tried to vary the value between 0.7 and 0.9, decreasing the value when better solutions were not being found (thus increasing the exploration) and increasing it a new best value was found. This did not improve the overall performance of the algorithm and was eventually abandoned.

D.2.1 Summary

The assumption that the learning optimizers (ACO and DQL) would generate better solutions than the classical optimizers (greedy and WTA) was the initial focus for the tuning of these algorithms. Some effort to produce the best solutions was performed, but the tuning is not assumed to be the optimal tuning for the scenarios investigated.

VITA

Bryan attended the University of Washington from 2000 to 2004, graduating with a BS in Aerospace Engineering. From 2004 to 2007 he worked in GPS acquisitions for the US Air Force at Los Angeles AFB, before attending the Air Force Institute of Technology where he earned his MS in Astronautic Engineering in 2009. From 2009 to 2012 he worked at the Air Force Research Laboratory, Maui Space Surveillance and Supercomputing Center. In 2012 he was assigned to Peterson AFB where he worked at Headquarters Air Force Space Command, before beginning his PhD program at Purdue University in 2016.