SSA Sensor Tasking: Comparison of Machine Learning with Classical Optimization Methods

Bryan D. Little* and Carolin Frueh†
Purdue University, West Lafayette, Indiana, 47906

The object population in the space around the earth is subject to increase. With the advancements in sensor capabilities, it can be expected that at the same time, more of those objects will be detected. Although this allows for significant advances in understanding of the detectable objects and the expansion of object catalogs, it also leads to significant stress on the sensor systems and makes efficient sensor tasking a prime challenge. In order to solve sensor tasking as optimization problem, various methods exist. Classical methods rely that the problem can be formulated in a convex representation. Computationally intensive methods based on artificial intelligence such as machine learning have recently gained a lot of attention and are suitable for problems even when no convex formulation can be found. In this paper, the performance of a simple traditional greedy algorithm is compared to the performance of two machine learning algorithms: Ant colony and distributed Q-Learning. Ant colony optimization is a swarm optimization path finding methodology based on probabilistic principles; distributed Q-Learning aims at finding an optimal policy in maximizing the expected reward over the whole scenario. Both can be seen as a version of reinforcement learning. As an application case the observation of all known objects in the geosynchronous region with a ground-based sensor is used. The performance is evaluated in terms of number of objects that have been successfully tracked and the computational run time in a given observation cycle.

I. Introduction

A foundational element of Space Situational Awareness (SSA) is the build-up and maintenance of a catalog of the resident space objects (RSOs) orbiting the Earth. The object population in the space around the earth has been subject to constant increase since the beginning of the space age[1]. The U.S. Strategic Command uses the Space Surveillance Network to maintain a catalog of over 23,000 RSOs larger than ten centimeters[2, 3]. However, NASA and the European Space Agency (ESA) both estimate that there could be on the order of 100 million debris objects between 1 millimeter and 10 centimeters that are not currently maintained in a catalog[3, 4].

As sensor capabilities continue to improve it will lead to more and more objects being able to be detected[3, 4], potentially increasing the catalog by an order of magnitude. Although this allows for significant advances in understanding of all objects in the near earth region and the expansion of object catalogs, it also puts significant stress on the sensor systems, ranging in the best cases in the order of hundred sensors. As positional uncertainties of space objects increase over time, sensors are tasked with both, keeping track of all known objects (catalog maintenance) and also to detect unknown or previously lost objects (catalog build-up) at the same time.

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, and other observational constraints (e.g. during the night for optical observations) and sensor availability. Furthermore, the detectability of each object is time-dependent, based on observation geometry but also object specific properties, such as attitude motion, shape, and reflectivity of surface materials. This leads to a complex sensor tasking optimization problem.

In order to solve this optimization problem, various methods have been explored [5–14], ranging from heuristic assumptions to rigorous mathematical modeling. Recently, Linares and Furfaro have used reinforcement learning (actor/critic policy method) to develop an optimized strategy for a ground-based sensor (GBS)[6]. They have been able to show that for small populations and a single sensor, the method does reduce the trace of the position covariance

*PhD Student, School of Aeronautics and Astronautics, Purdue University.
†Assistant Professor, School of Aeronautics and Astronautics, Purdue University.

below a predefined level. Hill, et al., presented a method for scheduling a network of sensors by considering the orbit covariance of each RSO given a centrally generated task list[8]. They were able to show that using the uncertainty of each object to determine when to schedule individual observations, that the overall median position uncertainty of the catalog could be effectively reduced. Both of these methods are based on the assumption of assigning a single RSO for observation at each observation time.

In this paper, one formulation of the sensor tasking problem is used, balancing survey (detection of new objects) and follow-up (catalog maintenance), maximizing the detections based on maximum probability over the observation interval in a grid based approach that favors simultaneous detection over single detections [5], that has previously been evaluated and tested. Efficient ways of evaluating the sensor tasking are discussed.

Then, based on this formulation, three different optimization methods are compared. The first is a simple locally greedy algorithm which is not guaranteed to produce optimal results, but is computationally very efficient. The greedy algorithm chooses the best observation considering only the expected number and probability distributions, respectively, of previously unobserved RSOs at every step; this is akin to a steepest ascent algorithm, which assumes that local optimality ensures global optimality [15–18]. The performance of the greedy algorithm is compared to Ant Colony System (ACS) and Distributed Q-learning (DQL) reinforcement learning algorithms. Reinforcement learning is a class of machine learning algorithms that have been studied in many different disciplines, such as information theory, multi-agent systems, and swarm intelligence[15].

ACS is a meta-heuristic optimization methodology based on the probabilistic principles approximated by the natural behavior of ants and has been used to solve problems such as the Traveling Salesman Problem, the Quadratic Assignment Problem and the Job-shop Scheduling Problem[19–22]. The DQL algorithm allows multiple agents to explore the environment in order to discover an optimal policy by choosing state-action pairs at every time and collecting an associated reward (or penalty) [15, 23, 24]. The agent that finds the best policy (highest total reward) updates the values of the state-action pairs it used; the method continues to iterate until some stopping condition is met[15, 23, 24].

In order to compare the optimization strategies, a ground-based sensor (GBS) tasking scenario is used. For comparability, it is assumed that *a priori* information on the objects is available and only follow-up observations for catalog maintenance are to be tasked. Successful detection is based on a simulated truth object, that is sampled from the initial not the propagated probability density function (pdf). The optimization framework however has only knowledge of the first and second moment of the propagated pdf available for a tasking decision.

For the evaluation, the TLE catalog of geosynchronous objects has been used: It contains 1231 RSOs bounded by $r_p \geq 35,378$km and $r_a \leq 45,378$km, with no restrictions on inclination or right ascension of the ascending node at the time of evaluation*. Each object (in contrast to pure TLE data) is equipped with an orbital uncertainty, which is propagated in an Extended Kalman filter framework, in order to simulate a real world sensor tasking for catalog maintenance scenario. The performance of the different algorithms is evaluated in terms of the number of objects that have been successfully recovered in the sensor tasking of one observation cycle using the same ground truth scenario and the same object pdf sampling (fixed seed) in the tasking.

The paper is organized as the following. First, Frueh's formulation of the follow-up problem and the coordinate system that will be used is introduced. Next, the transformation of the RSO uncertainty into the measurement space is discussed as a means to weight the pointing directions. Finally, the greedy, ACS and DQL algorithms that are used to find solutions are discussed and results are shown for two different cases: a) RSO mean positions representing truth (no uncertainty), which is used as a best case albeit highly unrealistic scenario and b) optimization considering the RSO position uncertainty based on the first and second moment of the propagated object's pdf.

*Catalog used is from www.Space-Track.org for day 270, 2016 (September 26).

II. Sensor Tasking Problem Formulation

A. Cost Function

The problem is formulated as a maximization of the set of weighted viewing directions ($A = \{a_f\} \mid f = 1, 2, \ldots, m_g$) where the weighting represents the expected value of the RSOs to be observed[5]. The formulation given in equation 1 is specific to the follow-up problem for a catalog of known RSOs and will be applied to a catalog of objects in GEO. A more general formulation that includes surveillance for unknown or untracked objects has been presented by Frueh[5].

$$A = \max \sum_{f=1}^{m_g} \cdot \Big(\sum_{i=1}^{n} \mu(\hat{x}_i) \cdot p(\hat{x}_i) \cdot d(h, \hat{x}_i) \Big) \tag{1}$$

In equation 1, m_g represents the total number of viewing directions in a given observation window, which also represents the discretization of the problem with respect to time. n is the number of objects that are to be observed in the scenario. $\mu(\hat{x}_i)$ is the object specific value, quantifying the need for the new observations, it can be e.g. based on an uncertainty criteria or an information gain. $p(\hat{x}_i)$ is the probability of detection for object i. It may be based on the mean of the probability distribution function or on further moments of the distribution. $d(h, \hat{x}_i)$ is the probability that the object i falls within grid field h; the value for $d(h, \hat{x}_i)$ can be based on the cumulative distribution function value in a given viewing direction a_f.

Theoretically, a sensor has infinitely many viewing directions to which it can be pointed within the field of regard. In order to simplify the optimization problem, the field of regard is discretized into a grid of viewing directions based on the sensor field of view (FOV) and any sensor specific constraints, such as minimum elevation. The angular definitions for the allowable viewing directions are referred to as grid fields, and they depend on the coordinate system chosen for the sensor. This work uses the Local Meridian Equatorial (LME) coordinate system (τ_t, δ_t) for the ground-based sensor (GBS); the associated angles are defined below.

B. Viewing Directions

For the grid field definition, a coordinate frame decision has to to be made. Assuming angular measurements, sensor viewing directions in the ground-based scenario are defined in a topocentric spherical coordinate frame. In this paper, the local meridian equatorial (LME) system is chosen for the definition of the sensor viewing directions. The authors assume the observer position is perfectly known.

The LME system identifies the grid fields for the GBS in terms of the topocentric hour angle (τ_t) and declination (δ_t), given in equations 4-5, where θ_{lmst} is the observer's local sidereal time. The system is defined with the fundamental plane parallel to the Earth's equator and through the observer location and the principal direction given by the observer's local meridian[25]. The grid fields are found using equations 2-3, where $\bar{u}$ represents the pointing vector to the center of the grid field with respect to the sensor in inertial space.

$$\bar{\rho} = \rho \cdot \bar{u} = \bar{r} - \bar{R} \tag{2}$$

$$\bar{u} = (u_{\hat{x}}\ u_{\hat{y}}\ u_{\hat{z}})^T \tag{3}$$

$$\tau_t = \tan^{-1}\left(\frac{u_{\hat{x}} \sin\theta_{lmst} - u_{\hat{y}} \cos\theta_{lmst}}{u_{\hat{x}} \cos\theta_{lmst} + u_{\hat{y}} \sin\theta_{lmst}}\right) \tag{4}$$

$$\delta_t = \sin^{-1}(u_{\hat{z}}) \tag{5}$$

The LME system is closely related to the common Equatorial Vernal Equinox (EVE) system, where viewing directions are defined in terms of topocentric right ascension and declination. The advantage of LME over EVE is in the fact that the system is static with respect to the observer, which allows for the grid fields to be defined once and remain valid for all observation times instead of constant re-definition of the grid required in the EVE right ascension and declination system. In the first scenario, no orbital uncertainty is taken into account and it is assumed that the true positions of all objects are known to the optimizer. In this case, weightings of the grid fields are found by the number of

objects in each grid field direction. This comprises the upper performance bound in the presence of perfect knowledge for all optimizers. The realistic scenario includes orbital uncertainty.

Under inclusion of orbital uncertainty, the further advantage of the LME system the more common topocentric local meridian local horizon system (LMLH) is that it preserves the fact that a linearized transformation using a Jacobian is a better approximation in the LME than for the LMLH system.

This can be shown via Monte Carlo analysis by generating possible RSO positions based on the mean state, $\mathbf{X}$, and covariance, $\mathbf{P}^{-}$, of an RSO and transforming each particle into the measurement space. Assuming Gaussian uncertainties, the linearized measurement Jacobian is used to transform the covariance into the measurement space using equation 6. The squared Mahalanobis distances (MD) is calculated for each particle in the measurement space, with respect to the distribution $\mathbf{P}_z$, by equation 7.

$$\mathbf{P}_z = \mathbf{H}_f \cdot \mathbf{P}^{-} \cdot \mathbf{H}_f^T \tag{6}$$

$$MD = (\mathbf{Z}_p - \mathbf{Z}) \cdot \mathbf{P}_\mathbf{z} \cdot (\mathbf{Z}_p - \mathbf{Z})^T \tag{7}$$

If the transformation is perfectly linear the mean of the MD values for all particles with respect to $\mathbf{P}_z$ will be exactly equal to the dimension of the measurement space. Table 1 shows that the mean MD values for the LME measurement space is close, this is not the case when comparing to the identical transformation in the LMLH system. For advantages regarding computational speed in this paper, the use of linearized Jacobian transformation is preferred over other methods, such as unscented or Gaussian mixture transformation.

Table 1 The mean MD value is close to the linear value for the LME coordinate system.

	linear	τ_t, δ_t	β, h
Mean MD	2.0000	1.8408	106.8925

Using the LME system and linearized Jacobian transformation of the positional uncertainty from the Cartesian orbit space, weights for each grid field can be computed using the cumulative distribution function (cdf) values for each object with respect to each grid field; in every grid field, cdf value for each RSO is scaled by its current need to be observed, represented by $\mu(i)$. This leads to the value for each specific grid field.

In figure 1 a snapshot of the mean positions for the RSOs in the catalog are plotted against the grid for a field of view of $3^\circ x 3^\circ$ with a minimum elevation constraint of 12° (28 Sep 16, 1827 hrs, 48.7 mins; $\bar{R}^i_{obs} = [4618.43, -2714.08, 3465.96]^T$ km). Points that fall outside the grid represent RSOs that are not visible to the sensor (e.g., on the opposite side of Earth). It represents the first scenario where absolute truth knowledge is assumed. Figure 2 shows the scaled cdf weightings for the same observation scenario. This is used when orbital uncertainty is taken into account for the second scenario in this paper.

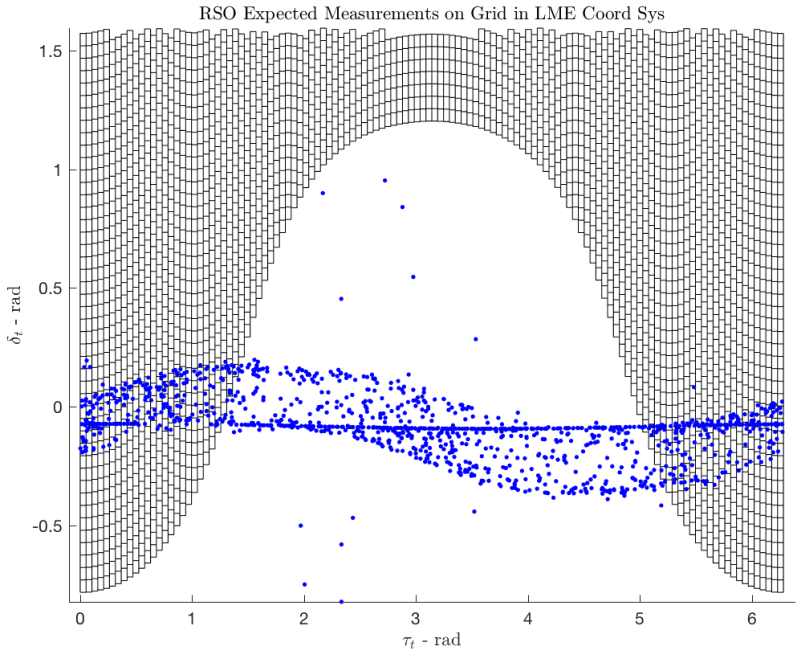


Fig. 1 Grid of pointing directions (black) for a ground based observer plotted against true positions of the catalog objects (blue) in (τ_t, δ_t) space.

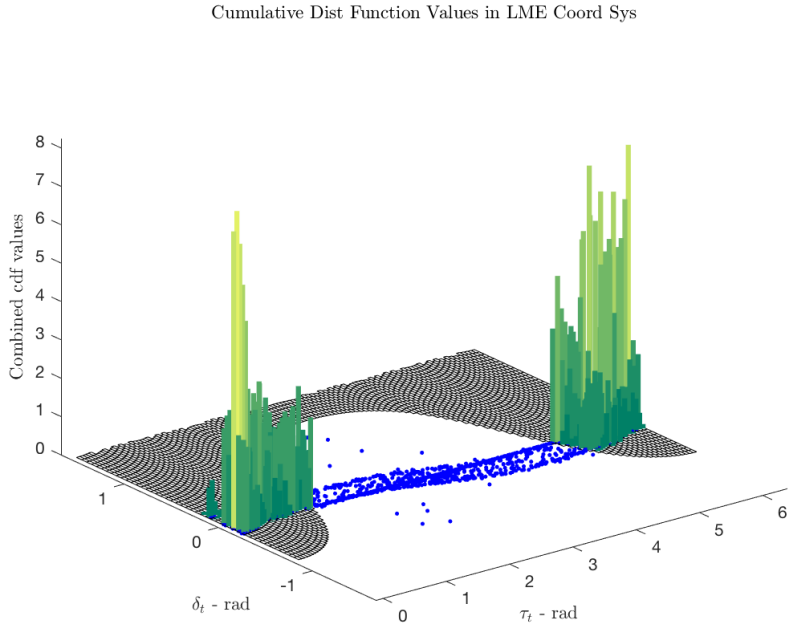


Fig. 2 Grid of pointing directions (black) for a ground based observer plotted against cdf values of catalog objects with uncertainty in (τ_t, δ_t) space.

III. Optimization

A. Convexity

Convex optimization is well investigated; convex systems have the advantage that optimal solutions are guaranteed to exist and that, in general, solutions can be found in a tractable manner [17, 18]. A problem being non-convex does not preclude use of convex optimization techniques, but it does imply that such methods will not generally produce optimal solutions.

The general convexity condition for a continuous function is given by equation 8 and holds for all convex continuous functions.

$$\lambda f(x) + (1-\lambda) f(y) \leq f(\lambda x + (1-\lambda) y) \tag{8}$$

The problem of sensor tasking for SSA is not a continuous problem, but can be modeled as a discrete process in two ways: a) the sensors are only capable of observing a small portion of the sky (the FOV) at a given observation time and b) the sensors will take an observation at only one pointing direction for each observation time[5, 26]. After an observation is taken, the sensor may be repositioned before taking another observation at a later time. So there is both a spatial and a temporal discretization of the observation window, which results in a high dimensionality for the sensor tasking problem [5, 6, 26].

In order to check for convexity, it becomes necessary to introduce the discrete convexity conditions in equations 9 and 10, where $x, y \in S$, $S \subseteq \mathbb{Z}^n$, $g : S \rightarrow \mathbb{R} \cup \{+\infty\}$ or $g : S \rightarrow \mathbb{Z} \cup \{+\infty\}$ depending on if the function, g, is real-valued or integer-valued[†]; these equations are derived from variations of equation 8 and taking into account constraints enforced by a discrete problem (see Murota, 2003)[18, 27]:

$$g(x) + g(y) \leq g((x - \alpha \mathbf{1}) \vee y) + g(x \wedge (y + \alpha \mathbf{1})) \tag{9}$$

$$g(x) + g(y) \leq g(x - \chi_u + \chi_v) + g(y + \chi_u - \chi_v) \tag{10}$$

Murota defines functions that obey equation 9 as $\mathrm{L}^\natural$ discrete functions, satisfying *translation sub-modularity* within the set of allowable points S, and functions that obey equation 10 as $\mathrm{M}^\natural$ discrete functions, satisfying the *exchange property* within the set of allowable points S[18, 27]. If the function g satisfies one or both of equations 9 and 10, then the problem belongs to the class of convex extensible discrete functions[18, 27].

In this paper, not general convexity condition is tested, but non-convexity is shown via providing a counter example. A full night scenario was run using the Purdue Optical Ground Station as the observer ($\bar{R}^{ECEF} = [-1434.14, -5161.33, 346596]^T$ km). Two-body propagation was used to generate the expected positions of 1231 GEO objects at each time step, separated by 128 seconds, during the window from 1826 hours, 44.7 seconds on 28 Sep 16 to 0514 hrs, 13.2 seconds on 29 Sep 16; the observations were simulated at the midpoint of each step with the first occuring at 1827 hours, 48.7 seconds. No uncertainty was considered with two-body positions being taken as perfectly known truth.

Three points (sensor tasking strategies) were generated using a greedy algorithm: x took the local optimal for each viewing direction, while y_1 and y_2 took the second and third best choices for each viewing direction, respectively. The point pairs (x, y_1) and (x, y_2) were then put into equations 9 and 10 to check for convexity. Table 2 shows the results of the convexity conditions for both pairs of points (the left sides of equations 9 and 10 are equal). In terms of $\mathrm{L}^\natural$ and $\mathrm{M}^\natural$ convexity, the point pairs fail to meet the convexity conditions, indicating that the problem is not convex.

In the following three different optimizers are used to solve the non-convex sensor tasking problem. The local optimal greedy algorithm used in this paper is only guaranteed to produce optimal solutions for a truly convex problem that is temporally decoupled. Neither is the case in the sensor tasking problem. Convexity is explicitly violated as shown above; temporal coupling is present as such that the previously observed objects are not sought to be observed again, hence leading to a coupling of the current tasking choice with the choices in the previous time steps. The ACS and DQL algorithms do not make this same assumption and are expected, with proper tuning, to produce more optimal solutions than the greedy algorithm.

[†]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 any element of S and every other element of S.

Table 2 The values for the left and right sides of Equations 9 and 10 are given for the pairs (x,y_1) and (x,y_2).

Measure	(x,y_1)	(x,y_2)
Left Side	717	664
L♮ Right Side	604	642
M♮ Right Side	487	501

B. Greedy

The greedy algorithm determines the local optimal for each viewing direction by simply choosing the grid field with the highest weighting at the current time step. When the uncertainty of the RSO positions is included in the weighting, this algorithm attempts to find the viewing direction that will contain the maximum combined probability of RSOs that need to be observed, $a_f = h^*$, where h^* is given by equation 11[28].

$$h^* = \arg\max_h \sum_{i=1}^{n} \mu(i) \cdot p(\hat{z}_f(i)) \cdot d(h, \hat{z}_f(i), \mathbf{P}_{z,f}(i)) \tag{11}$$

The greedy algorithm assumes that the local optimal choice at each step will lead to the global optimal solution[15, 17, 18]. As mentioned above, this explicitly neglects temporal coupling and the non-convexity of the problem, leading to a suboptimal solution of the problem. However, this allows it to be a very efficient algorithm, choosing the grid field with the highest value at the next step.

C. Ant Colony System

The Ant Colony System (ACS) is a specific type of Ant Colony Optimization introduced by Dorigo and Gambardella in 1997 as a way to solve a variety of optimization problems such as the Traveling Salesman Problem, the Quadratic Assignment Problem and the Job-shop Scheduling Problem [20, 21]. 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 [20–22, 24, 29].

The technique is based on sets of agents (ants) [20, 21]. Those agents explore the optimization cost function. In their exploration they are guided by two principles: the amount of pheromone left by previous agents, $\tau_{f,h}$, and a problem specific meta-heuristic value $\eta_{f,h}$ [15, 21, 24]. The pheromone itself is non-static but is chosen to evaporate. The evaporation, or the lessening of the pheromone number over time is ruled via the evaporation constant ρ, which is a free tuning parameter.

In this work the meta-heuristic is the grid field weighting; either the number of previously unseen RSO means, or the combined scaled cdf value. The two values are combined into the weighting factor, $w_{f,h}(t)$, given in equation 12[21]:

$$w_{f,h}(t) = \frac{[\tau_{f,h}(t)]^\alpha \cdot [\eta_{f,h}]^\beta}{\sum_{k=1}^{N} [\tau_{f,k}(t)]^\alpha \cdot [\eta_{f,k}]^\beta} \tag{12}$$

In equation 12, f and h are the current step and the grid field being considered, respectively, and t represents the current agent. The powers α and β are set to control the importance, or weighting, of the pheromone verses the meta-heuristic value [21]. The agents, guided by the weights $w_{f,h}(t)$, probabilistically choose to which grid field they will move; this allows the agents to solve the sensor tasking optimization problem as formulated in equation 1 by cooperatively assessing possible paths simultaneously.

Each set of agents passes through the entire set of observation instances m_g that make up the observation scenario over which the optimization takes place. In the simulations in this paper, the observation scenario comprises of one night. When a set of agents completes the analysis, they lay down pheromone on the viewing directions they chose based on the total value of the objects observed along their path. The pheromone values evolve based on equation 13 and 14, where ρ is the pheromone evaporation constant ($0 < \rho < 1$), $s(f,h)$ is the number of agents that chose the viewing direction defined by the time step, grid field pair (f,h), N_j is the expected value gained by each agent for choosing that viewing direction, and Q is a scaling factor and can be the total number of RSOs in the catalog, or could also be all visible objects during the night or at that particular instance [20].

$$\tau_{f,h}(t) = \rho \cdot \tau_{f,h}(t-1) + \Delta\tau_{f,h} \tag{13}$$

$$\Delta\tau_{f,h} = \sum_{j=1}^{s(f,h)} \frac{N_j}{Q} \tag{14}$$

As each set of agents completes their analysis, the value A, the optimization cost function equation 1, is determined for each agent and compared with the best value for the cost function A chosen by the previous sets of agents. It is possible to have more than one path that achieves the maximum value for A, and so all such paths will be saved until a higher value for the cost function A is found.

In this analysis, the ACS uses eight sets of agents to pass through the observation period in order to determine the best path, the pheromone importance is set to $\alpha = 1$ and the meta-heuristic importance is set to $\beta = 1.2$. The optimizer sets the number of agents as an integer multiple of the maximum amount of non-zero valued grid fields expected, and limits the number that can pass through a grid field at a given viewing direction to help ensure path diversity as the pheromone levels increase; the pheromone values are kept from growing uncontrollably by the pheromone evaporation constant, $\rho = \frac{1}{3}$, which is applied before the agents deposit their new pheromone values.

D. Distributed Q-Learning

Distributed Q-learning is a reinforcement learning algorithm that was introduced by Mariano and Morales, which extends the single step Q-learning [Watkins, 1989][15, 23, 24]. 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[24]. The value function $Q(s, a)$ represents the previously determined values for every state-action pair (s, a). The state action pair contains the reward r for a given action, a discount factor $gamma$, which de-weights the reward in future steps and the learning rate α, which counteracts the de-weighting function of the discount factor.

In order to allow for variation and exploring of the optimization, often a conservative so-called ϵ-greedy strategy is used. In this strategy, most of the time the agent chooses the predetermined best next action and only occasionally $(0 < \epsilon \ll 1)$ chooses to search other possible actions; equation 15 describes this strategy, where $\pi(s)$ is the choice of action a, and q is a random value from the uniform distribution over $[0, 1]$ [23]. This possibility for deviation and variation gives the method the flexibility to also handle non-convex problems. The less conservative epsilon is set, the more flexibility the method allows at the cost of increased run times.

$$\pi(s) = \begin{cases} \arg\max_a \hat{Q}(s, a) & \text{if } q \leq \epsilon \\ a_{random} & \text{otherwise} \end{cases} \tag{15}$$

The $Q(s, a)$ values are updated based on equation 16, where s is the current state, a is the chosen action, r is the reward, s' is the new state, γ is a discount factor $(0 < \gamma \leq 1)$ for future actions, and α is the learning rate (often $\alpha = 1$)[24]:

$$Q(s, a) \Leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)] \tag{16}$$

DQL is similar to ACS in that respect that there are multiple agents attempting to solve the problem during each iteration[23]. However, the agents are not updating the $Q(s, a)$ values with a reward at each step, as done in Q-learning; instead, each agent carries a copy of the values $(Q_{c^i}(s, a))$ which is updated according to[23]:

$$Q_{c^i}(s, a) \Leftarrow Q_{c^i}(s, a) + \alpha[\gamma \max_{a'} Q_{c^i}(s', a') - Q_{c^i}(s, a)] \tag{17}$$

Unlike ACS, the agents do not use a problem specific heuristic in choosing which actions to take [23, 24], nevertheless tuning parameters in the discount γ and learning rate α are also present in distributed Q-learning. Once all the agents complete an iteration, the best policy found during that iteration is used to update the $Q(s, a)$ values using equation 16[23]. For this work, the reward is the value of the cost function A, equation 1, chosen by the state-action pair.

In Q-learning and DQL, the initial values for $Q(s, a)$ are arbitrarily set and the agents only update them based on the rewards received [23, 24]. In other words, the agents begin from no understanding of the optimal policy, but generate the policy by balancing exploration of new state-action pairs with exploitation of previously explored "good" state-action pairs. The learning rate is set to the generic value $\alpha = 1$, the discount parameter is set to $\lambda = 0.75$, and the exploration parameter is set to $\epsilon = 0.3$ in our implementation, using 2000 iterations with 210 agents.

IV. Simulations

Each of the optimizers was applied to a ground-based sensor observation scenario. The GBS sensor is based on the Purdue Optical Ground Station[‡] and the observation time lasted for 10 hours and 47 minutes on the night of 28 September 2016 from astronomical sunset to sunrise.

In the chosen implementation of equation 1 of the optimization cost function, a probability of detection of one was assumed for all scenarios. For the scaling factor μ, immediate feedback is assumed. It is set to zero upon successful detection; otherwise the value is kept constant in the current implementation, effectively given all objects equal need to be observed at all times until successfully detected.

For the catalog of objects, the 1231 geosynchronous objects from the TLE catalog (bounded only by $r_p \geq 35,378$km and $r_a \leq 45,378$km) are propagated to 28 September 2016 at 0000 hours 00 seconds. The object states ($(X)_i = [\bar{r}, \bar{v}]^T$) are taken as the initial states for the propagation by the extended Kalman filter, and each is given an initial uncertainty of 50 kilometers along each position coordinate and 10 meters/second along each velocity coordinate. The state and covariance for each object is then propagated for 18 hours, 26 mins and 44.7 seconds to the beginning of the observation window. The ACS algorithm uses eight iterations with 1270 agents working through the problem for case 1, where absolute knowledge is present, and 1850 individual agents for case 2, where uncertainty is included. The difference is a result of the fact that there are more grid fields to choose from when considering the scaled cdf values computed from the object uncertainties than there were when only the true positions were considered. The pheromone importance is set to $\alpha = 1$ and the meta-heuristic importance is set to $\beta = 1.2$. Initially, all viewing directions (time step, grid field pairs) have an equal and small value of pheromone and the agents choose random paths through the viewing directions that have non-zero values. The path that achieves the best value is saved and each agent then deposits pheromone on the viewing directions along their path according to equations 13 and 14. The remaining iterations continue using equation 12 to choose viewing directions and replacing the best path when a higher value is achieved.

The DQL algorithm uses 2000 iterations with 210 agents for case 1; the number of agents is based on the number maximum number of non-empty grid fields expected during the observation window. The learning rate is set to $\alpha = 1$, the discount parameter is set to $\lambda = 0.75$, and the exploration parameter is set to $\epsilon = 0.3$. During the first iteration, the agents exclusively explore the viewing directions with non-zero values by randomly choosing actions to move from their current state to the next state. This allows the agents to generate a first improvement to the $Q(s, a)$ values. During the remaining iterations, the agents randomly choose between exploiting the $Q(s, a)$ values, or exploring other state-action pairs in attempting to find improved policies.

A. Case 1: Absolute Knowledge

The first implementation of the optimizers operates by assuming the states are perfectly known. In other words, no uncertainty is present in the object states. This reduces the problem to the observation of exact positions in the observation space. It represents an upper bound of the performance of each of the optimization algorithms in the current settings in the application of this scenario.

Table 3 contains the number of successfully observed objects by the follow-up strategies generated by each of the optimizers. The last column represents the number of objects that appear in the field of regard (above local horizon) at any point during the observation period and are hence theoretically observable during the observation time span. The last column allows for assessing how well each optimizer performs against the number that theoretically could be seen during the simulation.

Table 3 Number of successfully observed objects for each optimizer assuming absolute knowledge of the object positions at all times. The last column lists the total number of objects that are visible and therefore observable in the scenario.

Optimizer	Greedy	ACS	DQL	Possible
# obj	496	508	425	512

[‡]Lat: 33.0788° N, Long: 105.5286° W, Geocentric Radius: 6380.358 km

In Table 3 one can be seen that none of the optimizers manages to observe all 512 objects. However, it has to be noted that it is not guaranteed to be possible within the give time frame to observe all objects. ACS is performing best, observing all but 4 objects, greedy observes 12 objects less than the ACS, which still however amounts to 96.9 percent of the visible objects. DQL finds the least amount of objects with only 425, which amounts to 83 percent of the possible objects.

The DQL optimizer is currently under-performing compared to the other optimizers, however, it is not fully tuned yet and the results are expected to improve as the authors adjust the method. This algorithm was introduced as an alternative to the ACS, in part because of its lack of reliance on a problem specific meta-heuristic. The DQL begins with no knowledge of the solution space, and through iteration is able to generate a strategy to be employed by the sensor. This is different from the ACS, which at every step uses both previously gathered knowledge (pheromones) and a meta-heuristic (expected number of new objects) to guide the agents to the strategy solution. However, this also shows one of the problems with all learning algorithms, that they require tuning and it is not clear how that tuning translates to future nights, where no ground truth is available.

We can compare the viewing direction evolutions (grid fields at each step) and the number of RSOs observed at each step to see how the solutions of each optimizer differ. Figures 3 and 4 show the viewing direction evolutions for the Greedy and DQL algorithms; the size of the red $*$ depicts the number of new RSOs observed. In Figure 3, the dark blue (early in the window) grid fields are generally filled by the $*$, while the bright green (late in the window) only have a small red dot; as expected, the Greedy algorithm finds many objects early and fewer later. By comparison, Figure 4 has a more balanced number of the early and late grid fields with large and small $*$. This is also supported by Figure fig:six6, showing the number of observed objects over time. Whereas DQL has a steady increase, also in part due to the fact that it has more objects left to observe, Greedy has a steep increase in object numbers in the beginning and then flattens out.

Unlike the Greedy and DQL algorithms, the ACS algorithm produces three strategies that all achieve the same maximum number of RSOs observed for this scenario. Figure 5 shows the viewing direction evolutions for each of the three strategies generated by the ACS algorithm. Figure fig:six6 shows the object detection rate over time for the three ant colony solutions. It can be seen that two of them are very similar, whereas ASC_2 differs around time step 25, detecting fewer objects and then around time step 200 catches up with the other ant colony solutions. When comparing the ant colony solutions with the Greedy solution it can be seen that they perform very similarly until around step 160. Then Greedy reaches a plateau, where no new objects are detected. This is due to the fact that the greedy algorithm in its simple implementation misses out on objects that have set at this point but have been visible before but were not observed in the greedy strategy. It has also been shown that this can be overcome by including set times into the factor μ, boosting the performance of the Greedy algorithm, as shown in Frueh [5].

The algorithms are also compared based on their computation times as shown in table 4. As expected, the greedy algorithm is computationally the most efficient, taking only 15 minutes to completely assess the 10 hours and 47 minutes of observation time of the scenario. The DQL algorithm requires 1 hour and 57 minutes to complete the analysis; increasing the iterations or the number of agents is expected to improve the DQL results but will also increase the computation time required. The ACS algorithm is computationally expensive, requiring 5 hours and 30 minutes to compute the solutions; similar to DQL, the ACS algorithm could be adjusted to use more iterations or more agents to find better solutions, but the result would also be increased computational time.

Table 4 Computation times for each of the optimizers in the perfect knowledge scenario.

Optimizer	Greedy	ACS	DQL
Computation Time	15 mins	5 hrs 30 mins	1 hr 57 mins

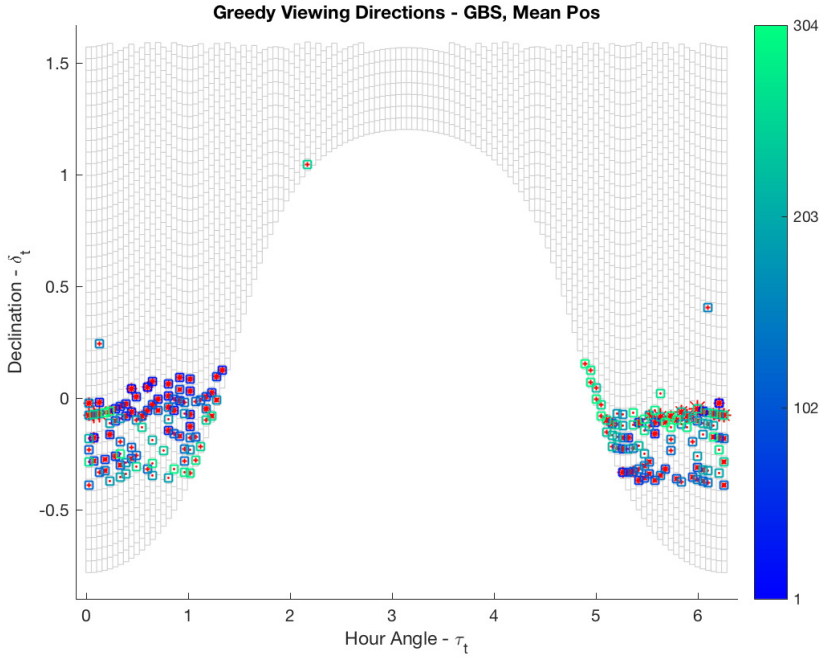


Fig. 3 Viewing Direction Evolutions for the Greedy algorithm. The grid field color is based on the step at which it is chosen and the size of the red $*$ indicates the number of RSO means observed.

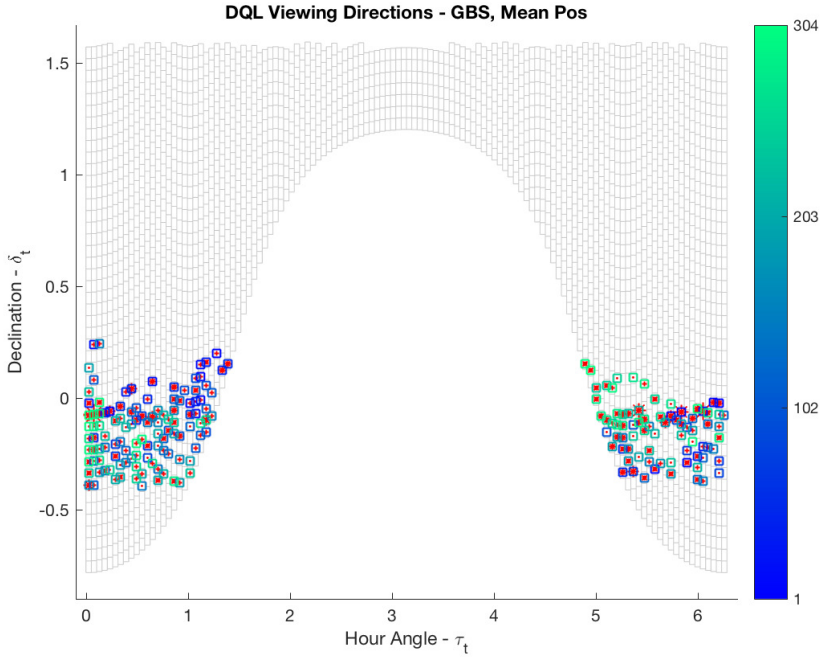
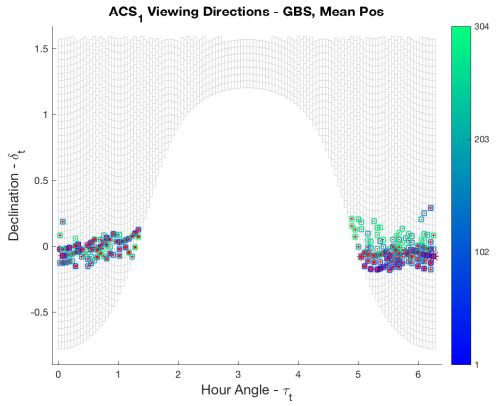
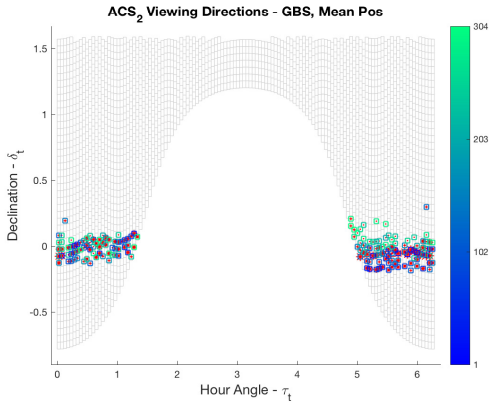


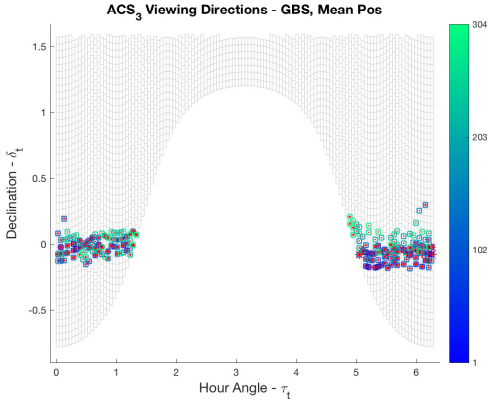
Fig. 4 Viewing Direction Evolutions for the DQL algorithm. The grid field color is based on the step at which it is chosen and the size of the red $*$ indicates the number of RSO means observed.



(a) First ACS strategy.



(b) Second ACS strategy.



(c) Third ACS strategy.

Fig. 5 The growth in the number of RSOs observed by the ACS algorithm strategies.

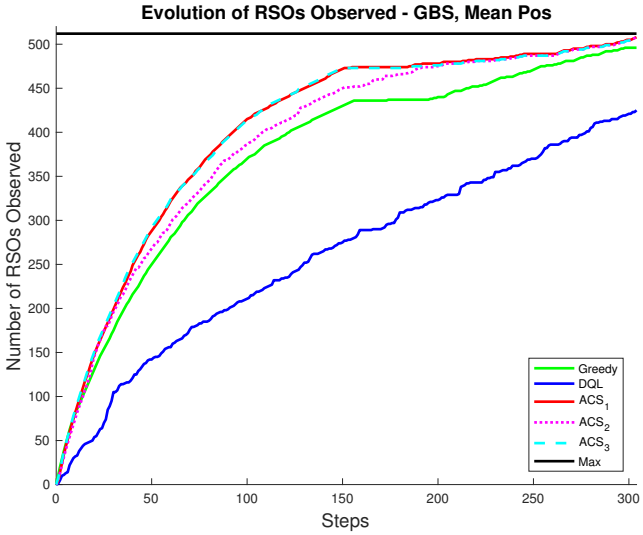


Fig. 6 The growth in the number of RSOs observed.

B. Case 2: Cumulative Distribution Function

In the second case, the more realistic setup is used that orbital uncertainty is present in the scenario. For each object the first two moments of the Gaussian probability density function (pdf) were propagated in an EKF framework, and at each observation time a true state was sampled from the pdf.

For each grid field, the cdf values are generated by integrating each object's probability density function over the entire FOV for each grid field at each step and are represented by the term $d(h, \hat{z}_f(i), \mathbf{P}_{z,f}(i))$ in equation 1. In order to determine successful observations represented in the scaling factor μ in the cdf implementation an immediate feedback via the sampled truth is assumed.

The total number of RSOs observed by the strategies by each optimizer are in Table 5. Again, in the last column the same number of objects is displayed that was visible and 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 linearizations in the object propagation and transformation into the observation space are present.

Table 5 Number of successfully observed objects for each optimizer in the presence of orbital uncertainty. The last column lists the total number of objects that are visible and therefore observable in the scenario.

Optimizer	Greedy	ACS	Possible
# obj	384	445	512

While the ACS algorithm again out performs the Greedy algorithm in the number of objects observed, it does so by a larger margin with the observation rates for the two algorithms being 75.0 and 86.9. The the viewing direction evolutions and number of RSOs observed at each step are plotted in Figures 7 and 8 for the greedy and ACS algorithms, respectively. By inspection, figures 7 and 8 appear to have less grid fields than figures 3 and 5. When considering the uncertainty, the Greedy algorithm only chose 128 unique grid fields for the 304 viewing directions, while the ACS algorithm chose 139 unique grid fields; by contrast with the previous case, the Greedy algorithm chose 185 and the ACS algoritm chose 174 unique grid fields when absolute knowledge was present.

The reasons for the decline in the number of unique grid fields are specific to each algorithm. Because the Greedy algorithm always chooses the highest scaled cdf value, if the grid field it chooses does not observe many of the RSOs that contributed to that scaled cdf value, the algorithm may end up selecting the same grid field in following steps to attempt to observe those RSOs that are still unobserved. The ACS algorithm, by contrast, is balancing the scaled cdf values and the pheromones resulting from previously generated strategies and may determine that revisiting certain grid fields at different points in the observation window is advantageous.

The growth rates of the RSOs observed have also changed from the previous case, as seen in figure 9; both algorithms have less steep growth at the beginning, while also not experiencing the plateaus seen in figure 6. It may still be likely to find an better solution using ACS by changing the parameters (number of iterations and/or number of agents), or perhaps using DQL and tuning it to the problem.

The computation time for the Greedy algorithm was significantly affected by the change to using the uncertainty, taking one hour and two minutes (nearly four times longer than using the true positions). Computing the combined scaled cdf values is responsible for the increased computation time. However, the ACS computation time remained nearly unchanged at 5 hours 47 minutes. The increase in the computation time for the combined scaled cdf values was offset by the change in the meta-heuristic calculation for the ACS. Using the true positions required each agent to carry a list of RSOs it had observed in order to check if it needed to see any of the RSOs in the next set of possible viewing directions; this was computationally expensive. With the scaled cdf values, the agents carried just the $\mu(i)$ values for every RSO, which were set to zero once an RSO was observed, using immediate feedback.

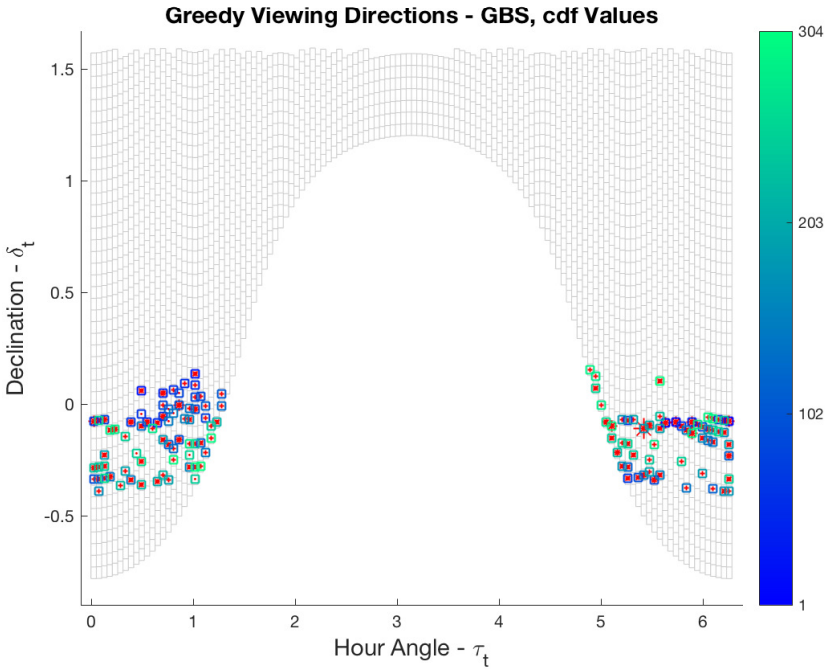


Fig. 7 Greedy algorithm viewing direction evolutions using uncertainty.

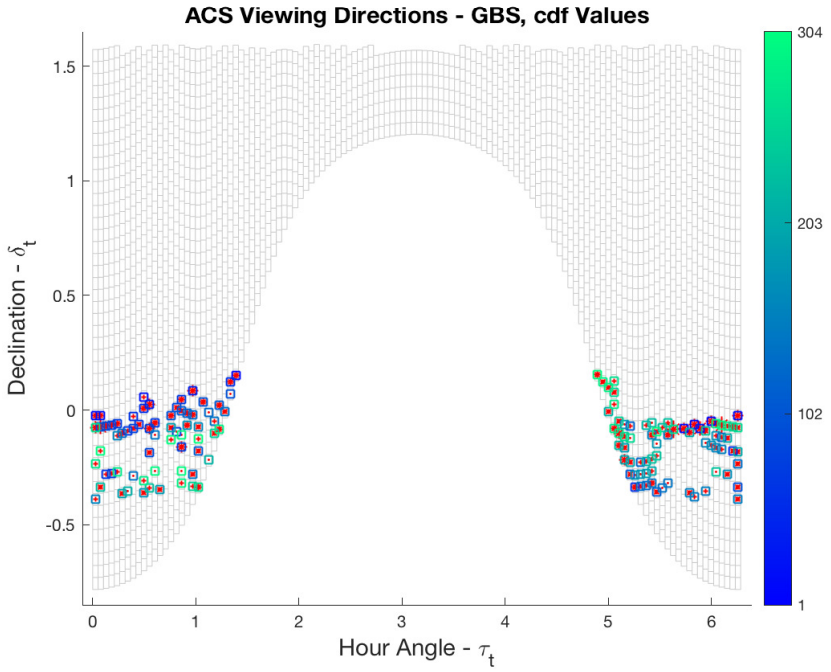


Fig. 8 ACS algorithm viewing direction evolutions using uncertainty.

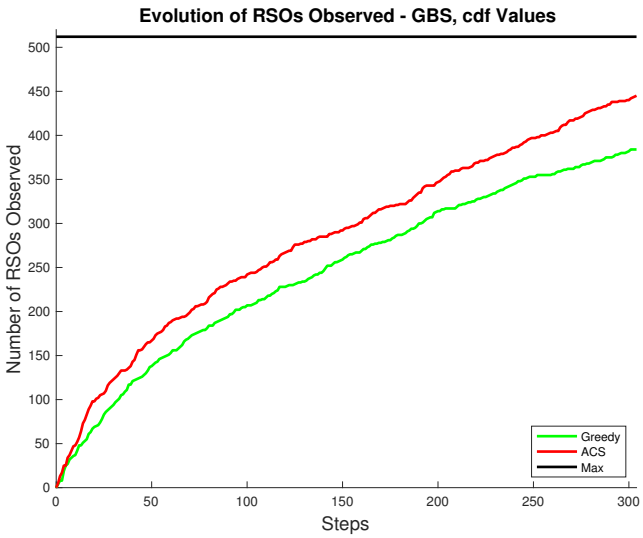


Fig. 9 The number of observed objects as a function of time for the different optimizers.

Table 6 Computation times for each of the optimizers in the scaled cdf values scenario.

Optimizer	Greedy	ACS
Computation Time	1 hr 2 mins	5 hrs 47 mins

V. Conclusions

This work presents a comparison between different optimizers to solve the sensor tasking problem. The Greedy, Ant Colony System and Distributed Q-learning optimization algorithms were introduced and used to generate sensor tasking strategies. The total number of Resident Space Objects tracked by the sensor and the computation time required are used to compare the three algorithms. For the evaluation a ground-based optical single sensor scenario is used, observing all geosynchronous objects in the USSTRATCOM catalog in a single night.

The Greedy algorithm assumes the problem is convex, which it is shown not to be, thus increasing the likelihood that the algorithm will find a less than optimal solution. The ACS algorithm uses a problem specific meta-heuristic, which aids the agents in solving the problem but also adds to the computation complexity. The DQL algorithm attempts to explore the entire problem, only exploiting state-action pairs that have previously resulted in improved policies; in the limit this should lead to finding the ideal solution, but significant tuning and computational time may be required. ACS and DQL both require tuning and no optimal tuning can be guaranteed.

Two different scenarios are investigated. One where the absolute truth of the object positions is known at all times. This represents an upper performance bound. In the second scenario, realistic orbital uncertainty was added. An extended Kalman filter framework was used for observation update and orbital propagation of the first two moments of the object uncertainty. Immediate feedback on successful observations was assumed.

The Greedy algorithm was shown to be the most computationally efficient in both scenarios, while the ACS algorithm was able to detect the most objects in both scenarios. The computational load was shown to increase for the Greedy algorithm from the case of absolute knowledge to the inclusion of uncertainty, while it remained nearly unchanged for the ACS algorithm. Greedy algorithm is still the most computationally efficient algorithm.

Acknowledgments

We would like to acknowledge that the support of this work via AFRL grant FA9451-18-C-0207 and the Purdue Military Research Initiative.

References

[1] ESA Space Debris Office, "ESA 's annual space environment report," Tech. Rep. 2, 2018. URL https://www.sdo.esoc.esa.int/environment{_}report/Space{_}Environment{_}Report{_}latest.pdf.

[2] USSTRATCOM, "Joint Space Operations Center," , apr 2018. URL http://www.stratcom.mil/Portals/8/Documents/JSpOCFactsheetFINALCAO.pdf?ver=2018-04-12-134128-903.

[3] ESA Space Operations, "Space Debris: The ESA Approach," , mar 2017.

[4] NASA, "NASA Technology Roadmaps: Introduction, Crosscutting Technologies, and Index," , jul 2015.

[5] Frueh, C., "Sensor Tasking for Multi-Sensor Object Surveillance," *7th European Conference on Space Debris*, Darmstadt, Germany, 2017.

[6] Linares, R., and Furfaro, R., "Dynamic Sensor Tasking for Space Situational Awareness via Reinforcement Learning," *Advanced Maui Optical and Space Surveillance Technologies Conference 2017*, Maui Economic Development Board, Maui, HI, 2017.

[7] Jaunzemis, A. D., Holzinger, M. J., and Jah, M. K., "Evidence-based Sensor Tasking for Space Domain Awareness," *Advanced Maui Optical and Space Surveillance Technologies Conference*, Maui Economic Development Board, Maui, HI, 2016.

[8] Hill, K., Sydney, P., Hamada, K., Cortez, R., Luu, K., Jah, M., Schumacher, P. W., Coulman, M., Houchard, J., and Naho'olewa, D., "Covariance-based network tasking of optical sensors," *Advances in the Astronautical Sciences*, Vol. 136, No. July, 2010, pp. 769–786.

[9] Schildknecht, T., "Optical surveys for space debris," , 2007. doi:10.1007/s00159-006-0003-9.

[10] Sharma, J., "Space Surveillance with the Space-Based Visible Sensor," *Space Control Conference*, edited by S. Andrews, 2000, pp. 115–124.

[11] Shell, J. R., "Optimizing Orbital Debris Monitoring with Optical Telescopes," Advanced Maui Optical and Space Surveillance Technologies Conference, sep 2010.

[12] DeMars, K. J., Hussein, I. I., Frueh, C., Jah, M. K., and Scott Erwin, R., "Multiple-Object Space Surveillance Tracking Using Finite-Set Statistics," *Journal of Guidance, Control, and Dynamics*, Vol. 38, No. 9, 2015, pp. 1741–1756. doi:10.2514/1.G000987, URL http://arc.aiaa.org/doi/10.2514/1.G000987.

[13] Musci, R., Schildknecht, T., and Ploner, M., "Orbit Improvement for GEO Objects Using Follow-up Observations," *Advances in Space Research*, Vol. 34, No. 5, 2004, pp. 912–916. doi:10.1016/j.asr.2003.01.019.

[14] Schildknecht, T., Hugentobler, U., and Ploner, M., "Optical Surveys of Space Debris in GEO," *Advances in Space Research*, Vol. 23, No. 1, 1999, pp. 45–54.

[15] Kaelbling, L. P., Littman, M. L., and Moore, A. W., "Reinforcement Learning : A Survey," *Journal of Artificial Intelligence Research*, Vol. 4, 1996, pp. 237–285.

[16] Neel, D. L., and Neudauer, N. A., "Matroids You Have Known," *Mathematics Magazine*, Vol. 82, No. 1, 2009, pp. 26–41. doi:10.4169/193009809X469020, URL http://openurl.ingenta.com/content/xref?genre=article{&}issn=0025-570X{&}volume=82{&}issue=1{&}spage=26.

[17] Boyd, S., and Vandenberghe, L., *Convex Optimization*, Cambridge University Press, 2004.

[18] Murota, K., *DISCRETE CONVEX ANALYSIS*, Society for Industrial and Applied Mathematics, 2003.

[19] Dorigo, M., and Blum, C., "Ant colony optimization theory: A survey," *Theoretical Computer Science*, Vol. 344, 2005, pp. 243–278. doi:10.1016/j.tcs.2005.05.020.

[20] Dorigo, M., Maniezzo, V., and Colorni, A., "Ant System: Optimization by a Colony of Cooperating Agents," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 26, No. 1, 1996, pp. 29–41.

[21] Dorigo, M., and Gambardella, L. M., "Ant colonies for the travelling salesman problem." *Bio Systems*, Vol. 43, No. 2, 1997, pp. 73–81. doi:10.1016/S0303-2647(97)01708-5.

[22] Merkle, D., Middendorf, M., and Schmeck, H., "Ant Colony Optimization for Resource-Constrained Project Scheduling," *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, Vol. 6, No. 4, 2002, pp. 333–346.

[23] Mariano, C. E., and Morales, E. F., "DQL: A New Updating Strategy for Reinforcement Learning Based on Q-Learning," *Machine Learning: ECML*, Vol. 2167, edited by L. De Raedt and P. Flach, Springer, Berlin, Heidelberg, 2001, pp. 324–335. doi:10.1007/3-540-44795-4_28, URL http://link.springer.com/10.1007/3-540-44795-4{_}28.

[24] Bianchi, R. A. C., Ribeiro, C. H. C., and Costa, A. H. R., "On the relation between Ant Colony Optimization and Heuristically Accelerated Reinforcement Learning," *1st International Workshop on Hybrid Control of Autonomous System*, 2009, pp. 49–55.

[25] Montenbruck, O., and Pfleger, T., *Astronomy on the Personal Computer*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1994. doi:10.1007/978-3-662-02982-4, URL http://link.springer.com/10.1007/978-3-662-02982-4.

[26] Herzog, J., "Cataloguing of Objects on High and Intermediate Altitude Orbits," Dissertation, University of Bern, 2013.

[27] Murota, K., "Recent Developments in Discrete Convex Analysis," *Research Trends in Combinatorial Optimization*, edited by W. Cook, L. Lovász, and J. Vygen, Springer, Berlin, Heidelberg, 2009, pp. 219–260. doi:https://doi-org.ezproxy.lib.purdue.edu/10.1007/978-3-540-76796-1_11.

[28] Patel, M., Sinclair, A. J., and Ho, K., "Information-Theoretic Target Search for Space Situational Awareness," *AAS/AIAA Space Flight Mechanics Meeting*, , No. January, 2018. doi:10.2514/6.2018-0725, URL https://arc.aiaa.org/doi/10.2514/6.2018-0725.

[29] Dorigo, M., Birattari, M., and Stutzle, T., "Ant colony optimization," *IEEE Computational Intelligence Magazine*, Vol. 1, No. 4, 2006, pp. 28–39. doi:10.1109/MCI.2006.329691, URL http://ieeexplore.ieee.org/ielx5/10207/4129833/04129846.pdf?tp={&}arnumber=4129846{&}isnumber=4129833{%}5Cnhttps://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4129846.