Machine Learning for Space Domain Awareness Sensor Scheduling

Neil K. Dhingra, Cameron DeJac, Clayton McGuire
Auria

ABSTRACT

We present our novel approach using machine learning (ML) methods to train sensor scheduling algorithms for Space Domain Awareness (SDA). Since the full SDA sensor scheduling problem is NP hard, methods for identifying globally optimal sensor schedules do not scale well with the problem dimension. Rather than modifying traditional combinatorial optimization techniques to run at operational speeds, we train an ML algorithm to match their performance in less time. Sensor Scheduling for Space Domain Awareness (SDA) is a large, complicated, time-varying, and NP hard problem even under restrictive assumptions. As a result, solving full SDA sensor scheduling problems to global optimality is intractable, especially when scheduling algorithms must satisfy operational constraints such as those on runtime or a need to maintain an 'anytime' property. Conventional approaches often require identifying problem characteristics that can be exploited to deploy algorithms that create high quality schedules while satisfying operational constraints. We have employed such approaches successfully for scheduling SDA sensors and other space operations. However, identifying and exploiting these characteristics can be manually intensive and require expert knowledge. Our approach in this paper, in contrast, is to use machine learning (ML) techniques to train a scheduling algorithm that satisfies operational constraints and accomplishes this mission without explicitly exploiting problem characteristics. Rather, the ML training process is designed to learn these problem characteristics implicitly through the training data. We use supervised learning techniques to develop an algorithm that performs well on representative subproblem benchmarks, for whom optimal solutions are known, and use reinforcement learning to extend and further optimize its performance on the large and complex full SDA sensor scheduling problem. By limiting problem size, complexity, or both, we have identified several relevant and representative subproblems over which we can design optimal schedules. We generate high quality schedules offline using different methods that can solve these problems to global optimality. We then use supervised ML to initialize an ML scheduling algorithm using the optimal schedules as training data. Finally, we use reinforcement learning to further improve that algorithm's performance for the full SDA sensor scheduling problem while maintaining performance on the benchmark subproblems. The framework used to train this sensor scheduling algorithm is also extensible to other problem cases which are even less amenable to combinatorial methods. For example, many combinatorial problems are represented by mixed integer linear programs which require that the FOM is a linear function of the optimization variables. The ML framework developed in this paper imposes no such constraint on the form of the FOM. This framework can also be extended to support human feedback for the FOM. This is desirable in cases where operator preferences are hard to characterize mathematically, but where operators can score schedules to provide feedback on how well they do. We deploy the ML algorithm in our SDA sensor scheduling software, Heimdall, for evaluation. Heimdall runs several algorithms in parallel and chooses the one with the best FOM for execution. This framework enhances Heimdall's SDA sensor scheduling algorithm suite and facilitates the continued improvement of the other Heimdall SDA sensor scheduling algorithms. Heimdall has been deployed in support of SDA sensor scheduling for several customers and will support the forthcoming Deep Space Advanced Radar Concept (DARC) system.

1. INTRODUCTION

Sensor Scheduling for Space Domain Awareness (SDA) is a large, complicated, time-varying, and NP hard problem even under restrictive assumptions. As a result, solving full SDA sensor scheduling problems to global optimality is intractable, especially when scheduling algorithms must satisfy operational constraints such as those on runtime or a need to maintain an 'anytime' property. The conventional approach is to simplify expensive combinatorial optimization techniques to run on operational time scales and satisfy other constraints, often by identifying problem characteristics that can be exploited for performance. We have employed such approaches successfully for scheduling

SDA sensors and other space operations. In contrast, our approach in this paper is to use machine learning (ML) techniques to train a scheduling algorithm that satisfies operational constraints and accomplishes this mission. We generate high quality schedules offline, use supervised ML to initialize the scheduling algorithm with those schedules as training data, and then employ reinforcement learning to further optimize the algorithm.

The ML algorithm we design runs sequentially to iteratively add tasks to a sensor schedule. It takes as inputs the problem – schedule requirements, sensor configurations, object characteristics, slew times between objects, required dwell times, etc. – as well as the state of the schedule – time since last observation for each object, expected estimate covariances, etc. – in order to determine which observation to schedule next and when that observation should occur. Performance is measured by an objective function or Figure of Merit (FOM) that quantifies how well a schedule accomplishes all the schedule objectives, e.g., desired revisit rates on each object in a catalog, orbit estimation error covariances better than a given threshold, etc. The output of the algorithm is a distribution over objects and times that is used to determine which task is scheduled next and where it is inserted into the existing schedule. Once a task has been added, the algorithm takes the updated schedule as input and selects the next task to schedule. To initialize the algorithm using supervised learning, we first identify several tractable subproblems of the full SDA scheduling problem which are amenable to analysis via integer linear programming techniques, convex relaxations, exhaustive search, and/or other formal methods. It is possible to solve these subproblems to optimality, albeit not on operational timescales.

To avoid overfitting to the simpler benchmark problems and facilitate transfer of the techniques to more complicated problems, we employ reinforcement learning using several instances of the full SDA scheduling problem to refine the ML algorithm. The algorithm is adjusted to improve the FOM of schedules designed for these full problems.

The framework used to train this sensor scheduling algorithm is also extensible to other problem cases which are even less amenable to combinatorial methods. For example, many combinatorial problems are represented by integer linear programs which require that the FOM is a linear function of the optimization variables. The ML framework developed in this paper imposes no such constraint on the form of the FOM. This framework can also be extended to support human feedback for the FOM. This is desirable in cases where operator preferences are hard to characterize mathematically, but where operators can score schedules to provide feedback on how well they do.

We train different variants of the ML algorithm. For example, we investigate using reinforcement learning without the supervised learning initialization. We deploy these different variants of the ML algorithm in our SDA sensor scheduling software, Heimdall, for evaluation. Heimdall runs several algorithms in parallel [1, 2, 3] and chooses the one with the best FOM for execution. We employ this framework to compare instances of the ML algorithm against one another and against traditional approaches.

This framework enhances Heimdall's SDA sensor scheduling algorithm suite and facilitates the continued improvement of the other Heimdall SDA sensor scheduling algorithms. Heimdall has been deployed in support of SDA sensor scheduling for several customers and will support the forthcoming Deep Space Advanced Radar Concept (DARC) system.

Literature Review

Scheduling for SSA/SDA has been a very active area of research. Many scheduling approaches center estimation error covariance and determine the best sensors and timing of observations to achieve track accuracy objectives or to optimize information gain from measurements [4, 5, 6, 7, 8, 9]. Of particular interest has been the tradeoff between sensor accuracy and the cost of sensing [7, 2]. Some work has leveraged Commercial Off-the-Shelf Products combined with customized SSA/SDA-specific components (e.g., the approach of Heimdall) to create integrated solvers [5, 10, 11]. Others employ generalized nearest neighbor-based approaches [9, 12] or other meta-heuristics such as those based on pricing [13]. Often, slew time is neglected.

In [14], the authors pose SSA/SDA sensor tasking as an optimization problem and design an approach to solve it optimally. They tailored their approach for new object search missions in [15] and for multi-sensor coordination (i.e., stereo collects in the two sensor case) in [16].

Machine Learning has also been applied for SDA scheduling. The authors in [17] compare classical greedy and weapon-target assignment algorithms to ant colony optimization and distributed Q learning for SDA sensor scheduling. In [18], the authors use actor-critic reinforcement learning to perform scheduling for an SDA catalog considering track accuracy but not slew times.

Recently, we have begun formal analyses of SSA/SDA scheduling by considering how different CONOPS or algorithms affect the quality of sensor schedule achievable, the sensitivity of solution quality to particular algorithms [1], and the impact of track covariance-based tasking on sensor burden [2].

Such formal analysis is important because scheduling is a notoriously tricky problem and intuitive heuristics may not lead to good solutions. For example, in a traveling salesperson problem (TSP), a greedy nearest neighbor approach can yield the *worst* possible solution [19]. The TSP and vehicle routing problem (VRP) have been very active areas of study in the Operations Research field. Many of these problems are directly applicable to SSA/SDA sensor scheduling. VRPs [20] can be used to design slew paths for multiple sensors and VRPs with time windows (VRPTWs) [21] can be used to enforce viewability constraints. The Time Dependent Team Orienteering Problem with Time Windows (TDTOPTW), also known as the Tourist Trip Design Problem (TTDP) concerns multiple sensors, multiple targets of which not all are observed, time windows, and time dependence [22, 23], but do not typically deal with recurring tasks. Periodic VRPs (PVRPs) can be used to enforce revisit constraints [24, 25, 26, 27, 28] when revisit rates are strictly specified. Recently, Flexible PVRPs (FPVRPs) have been formulated to allow the planner the leeway to add extra visits if that will help find a better solution [29, 30]; the authors have shown that the benefit of this leeway cannot be bounded in general. The central aim of this paper is to leverage tools in the TSP/VRP operations research literature to inform analysis of domain-specific SSA/SDA scheduling algorithms.

Outline

In Section 2, we formulate the SSA/SDA planning and scheduling problem. We describe our approach in Section 3. Next, we discuss the algorithmic architecture of our SSA/SDA sensor scheduling software Heimdall in Section 4. In Section 5, we present results. Section 6 provides concluding remarks. Finally, Appendix A discusses our Heimdall software in more detail from a software and UI/UX perspective.

2. PROBLEM FORMULATION

2.1 Background

Effective SDA requires data collection that balances the monitoring of multiple heterogeneous objects by multiple heterogeneous sensors to satisfy multiple related objectives. Broadly speaking, the successful surveillance of any given object requires that tasking result in observations that are 1) of sufficient quality and 2) of sufficient quantity and density/regularity. High-quality data on objects are required so that they can be filtered for high-accuracy orbit determination and/or track accuracy that may be used to generate downstream data products, such as those related to object characterization and conjunction assessment. In addition, data must be collected at a sufficient cadence to prevent one from losing track of the object. For example, one would often like the time between observations of an object to be small enough such that even if the object maneuvers with some maximum ΔV immediately after the former observation, it would not have left the sensor's field of view as it performs the latter observation.

Requirements on data quality and quantity are coupled with one another, with the object, and with the sensors collecting data. Observations with better sensors will result in better data products and better tracks and, for objects less prone to maneuvers, may mean that less frequent observations are required to maintain a high-quality track. Tasking requirements can change with time; higher quality tracks may be required near potential conjunctions. Moreover, the value of information provided by a particular sensor is also dependent on factors such as the timing of data collection and existing knowledge about the object because the orientation and size of the sensor noise covariance ellipse in relation to the estimated covariance of the state estimate changes with the sensor mode, observation geometry, and other factors. Finally, effective tracking of space objects should be balanced with the search for new, previously unobserved space objects.

2.2 Mathematical Problem Specification

The scheduler must allocate, order, and time observation tasks for each sensor. We note that this is more complicated than most periodic routing problems because the scheduler has the freedom to allocate extra observations if that would allow it to lower the overall cost of the problem. This may seem unnecessary, and indeed the scheduler is incentivized to *not* overallocate observations if they are not needed. Nevertheless, such overallocation can be used to move slack time from a time when the sensors are undertasked to a time when they are overtasked; see [30, Thereoms 1, 2] for results in an analogous scenario that show that there is no general bound on the improvement such flexibility can buy.

In addition to meeting tasking requirements, the scheduler attempts to minimize some objective function or figure of merit. This can vary depending on the application and the operator CONOPS. Some operators prefer that the sensor fill its downtime with more tasks, i.e., the schedule will exceed the requirements on tasking (e.g., to collect more frequently than is required) if possible. This is typically the case for cheaper or less exquisite systems. Other operators prefer to minimize the energy expended by the system given that the requirements are met. This is often the case for expensive systems where the additional wear-and-tear on the sensors imposed by excess tasking is costly from a mission perspective. In this paper, we focus on the latter objective. However, the same framework and approach can be used to understand the case where excess tasking is desired.

2.2.1 Problem Data

We consider a catalog of n objects in the set $\mathbb{O} := \{o_0, o_1, \ldots, o_{n-1}\}$ to be observed by a set of m sensors in the set $\mathbb{S} := \{s^0, \ldots, s^{m-1}\}$. Observations are to be planned by sensors s^j on objects o_i during the planning period $[0,T]$.

Object o_i may be observed at a time $\tau \in P_i^j$ during which it is visible; P_i^j may consist of several potentially disjoint viewability periods $P_i^j := \bigcup_l [\underline{p}_{i,l}^j, \bar{p}_{i,l}^j] \subseteq [0,T]$. Observations on object o_i by sensor s^j at time τ take $d_i^j(\tau)$ seconds. Slewing sensor s^j from being positioned to perform an observation on object o_i at time τ to being positioned to perform an observation on object o_k takes $t_{ik}^j(\tau)$ seconds. The catalog has certain requirements on tasking. Each object o_i must be viewed every r_i seconds by any sensor.

We note that the problem data for $\mathbb{O}, \mathbb{S}, d_i^j, t_{ik}^j$, and r_i are determined by rigorous astrodynamics calculations. The estimation error in Heimdall is calculated by the SSA Enhancement module [1, 2] using an EKF or an Epoch State Filter.

2.2.2 Decision Variables

For each sensor s^j, the scheduler assigns the tour $A^j := a_0^j, a_1^j, \ldots$ with timing $T^j := \tau_0^j, \tau_1^j, \ldots$ where each assignment $a_i^j \in \mathbb{O}$ is an object to be observed by sensor s^j starting at time τ_i^j.

We consider optimization variables related to the start of each task $t \in \mathbb{R}^n$ and the acceleration incurred by the motors between each pair of tasks $a \in \mathbb{R}^{n-1}$. For convenience, we denote the kth task on object o_i as $a'_{i,k}$ to be observed at time $\tau'_{i,k}$. Note that $a'_{i,k}$ and $\tau a'_{i,k}$ merely re-index the assignments and start times per object for convenience.

2.2.3 Problem Statement

The full problem considered in this paper is

$$\begin{aligned}
&\text{minimize} && \sum_{i,j} d_i^j + t_{a_i,a_{i+1}}^j && && \text{(1a)}\\
&\text{subject to} && \tau_{i+1}^j \geq \tau_i + d_i^j(\tau) + t_{a_i,a_{i+1}}^j(\tau) && \forall i,j && \text{(1b)}\\
& && \tau_i^j \in P_i^j && \forall i,j && \text{(1c)}\\
& && \tau'_{i,k+1} \leq \tau'_{i,k} + r_i && \forall i,k. && \text{(1d)}
\end{aligned}$$

Colloquially, the problem is to minimize the effort of system (measured by slew time) while meeting requirements on object revisit rates. The objective function (1a) measures the total time the system uses to slew and dwell. Of course, this objective can be generalized with different weighting on slewing, dwelling, particular objects, etc. or it could be changed to encourage overcollection rather than efficient minimal collection.

Constraint (1b) enforces that between each successive set of tasks for each sensor, there is enough time for the sensor to perform the collection and slew. Constraint (1c) ensures that collections are performed on objects when they are feasible, due to viewability, weather, etc. Finally, constraint (1d) enforces the recurrence interval between successive collections on each object.

3. INTELLIGENTLY INITIALIZED RL

We describe our hybrid supervised learning-RL method.

3.1 Neural Network Scheduler

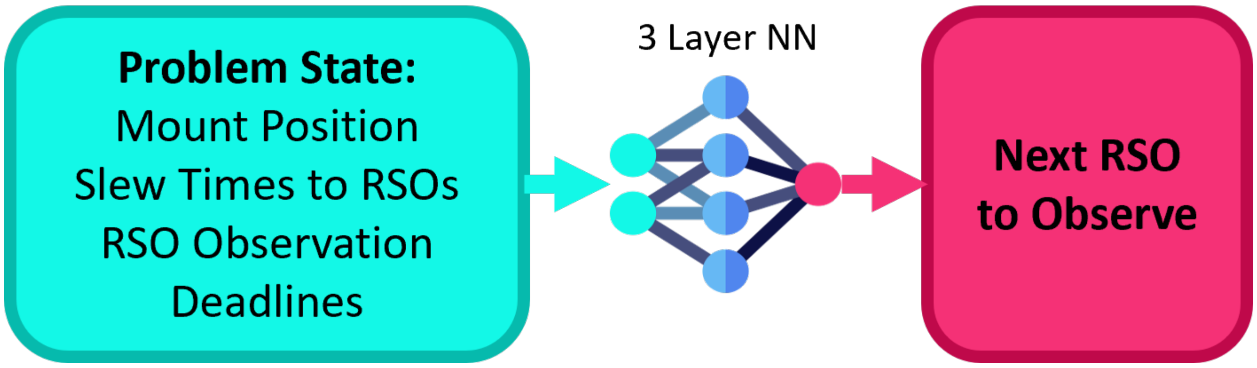


Fig. 1: NN Scheduler.

We employ a Neural Network (NN) as the ML scheduler. It takes as input the current pointing position, the slew times to each other RSO in the catalog, and the time until the next observation on each RSO must be performed. Its output is a probability distribution over which RSOs in the catalog to schedule next. The neural network itself has three linear layers with elementwise hyberbolic tangents connecting them. The first layer is of the size of the observation space – $3n$ – and the latter two are of the size of the output space – n. A final softmax layer normalizes the output to the probability simplex.

3.2 Supervised Learning Problem

Instead of directly targeting problem (1), we consider the traveling salesperson problem with time windows (TSPTW). This problem is the same as the traditional TSP with time window constraints on each city.

We consider these time windows to be from the current time to some point in the future. This TSPTW represents scheduling the next observations on all the objects in the catalog in problem (1) but does not consider second, third, etc. observations on those objects.

To initialize the NN, we sample instances of the TSPTW problem and solve them with Google OR-Tools [31, 32]. We sample random deadlines for the RSOs uniformly distributed from 0 to $\frac{2}{3}nT_{\max}$ where $T_{\max}$ is the largest slew time between object sin the catalog. We choose starting point. If the VRPTW algorithm finds a feasible tour, we store the inputs and the first RSO in the tour as the output using one-hot embedding.

Once we have gathered training and test data, we use supervised learning to fit the NN to match the VRPTW output. We perform the supervised learning in PyTorch [33, 34] using the AdamW optimizer, an MSE loss function, and epochs of size 128.

3.3 Reinforcement Learning Extended Problem

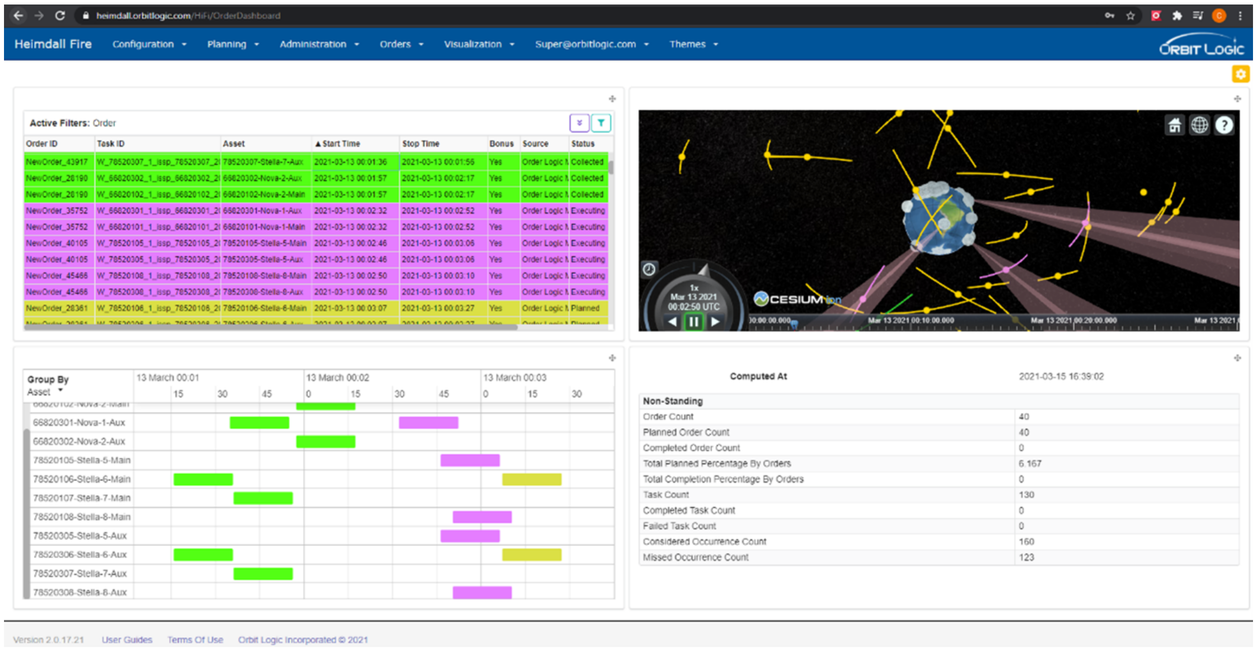
We set up the RL environment in Gymnasium [35]. The rewards for the actions are penalties on the slew time and on violating the maximum interval between successive observations on an object. The violation penalty is larger than the largest slew time to ensure that a slew and observation is always preferable to idleness.

The basic REINFORCE framework [36] is used to train the NN to adapt to the new problem. Relative to the TSPTW, the problem in the RL environment imposes a penalty on slewing, meaning that there is an incentive to wait when ther is not a pressing need to observe an RSO.

During RL training, the NN output probability distribution is sampled to determine which RSO to schedule next, imparting a large amount of variability on performance during training and enabling exploration of the solution space. During deployment, the RSO with the largest probability from the NN output is chosen to be scheduled next.

4. HEIMDALL

The framework described in this paper has guided the testing and refinement of Orbit Logic's Heimdall software. As a software tool, Heimdall provides a user-friendly interface in which operators can interact with orders/plan requirements, configure sensors/target objects, automatically leverage Orbit Logic's planning and scheduling algorithms to generate sensor schedules, visualize and validate these schedules in multiple ways. Heimdall generates coordinated, optimized observation schedules for the full set of available ground and space-based sensors for SDA observations. It leverages Orbit Logic's STK Scheduler scheduling algorithms, Orbit Logic's Collection Planning and Analysis Workstation (CPAW) scheduling algorithms and tools, and domain-specific Orbit Logic algorithms tailored for SDA. Moreover, several users can access the software at once and with heterogeneous permissions on what information they can view and what functions they can perform. More details on the software architecture and features of Heimdall may be found in Section A. The central Heimdall dashboard is shown in Fig. 2.



Active Filters: Order

Order ID	Task ID	Asset	▲Start Time	Stop Time	Bonus	Source	Status
NewOrder_43917	W_78520307_1_issp_78520307_2	78520307-Stella-7-Aux	2021-03-13 00:01:36	2021-03-13 00:01:56	Yes	Order Logic	Collected
NewOrder_28190	W_66820302_1_issp_66820302_2	66820302-Nova-2-Aux	2021-03-13 00:01:57	2021-03-13 00:02:17	Yes	Order Logic	Collected
NewOrder_28190	W_66820102_1_issp_66820102_2	66820102-Nova-2-Main	2021-03-13 00:01:57	2021-03-13 00:02:17	Yes	Order Logic	Collected
NewOrder_35752	W_66820301_1_issp_66820301_2	66820301-Nova-1-Aux	2021-03-13 00:02:32	2021-03-13 00:02:52	Yes	Order Logic	Executing
NewOrder_35752	W_66820101_1_issp_66820101_2	66820101-Nova-1-Main	2021-03-13 00:02:32	2021-03-13 00:02:52	Yes	Order Logic	Executing
NewOrder_40105	W_78520105_1_issp_78520105_2	78520105-Stella-5-Main	2021-03-13 00:02:46	2021-03-13 00:03:06	Yes	Order Logic	Executing
NewOrder_40105	W_78520305_1_issp_78520305_2	78520305-Stella-5-Aux	2021-03-13 00:02:46	2021-03-13 00:03:06	Yes	Order Logic	Executing
NewOrder_45466	W_78520108_1_issp_78520108_2	78520108-Stella-8-Main	2021-03-13 00:02:50	2021-03-13 00:03:10	Yes	Order Logic	Executing
NewOrder_45466	W_78520308_1_issp_78520308_2	78520308-Stella-8-Aux	2021-03-13 00:02:50	2021-03-13 00:03:10	Yes	Order Logic	Executing
NewOrder_28361	W_78520106_1_issp_78520106_2	78520106-Stella-6-Main	2021-03-13 00:03:07	2021-03-13 00:03:27	Yes	Order Logic	Planned

Computed At	2021-03-15 16:39:02
Non-Standing	
Order Count	40
Planned Order Count	40
Completed Order Count	0
Total Planned Percentage By Orders	6.167
Total Completion Percentage By Orders	0
Task Count	130
Completed Task Count	0
Failed Task Count	0
Considered Occurrence Count	160
Missed Occurrence Count	123

Fig. 2: Heimdall Dashboard Page with the light color theme

Heimdall competes multiple planning algorithms against each other in parallel and chooses the plan with the best resulting FOM or objective value. These algorithms may run in different threads. Algorithm execution times of longer algorithms will not block those of quicker algorithms. Some of these algorithms are anytime algorithms, so you can always stop computation early and have a valid plan.

The flow of each particular Heimdall algorithm consists of an initial solver followed by local search, as shown in Fig. 3. Some initial solvers employ greedy methods to ensure the rapid generation of a valid plan. For more computationally intensive approaches, the initial solver is often a coarse, holistic planner that may be sophisticated but approximate. This step is an avenue to leverage advanced but specialized approaches so that Heimdall can leverage combinatorial optimization techniques, convex relaxations, etc. even though they rely on problem structure.

This is possible because the local search step enforces constraints to high fidelity. Local search iterates between two steps: polishing, which adjusts task timing to obtain an optimized, deconflicted, and validated plan given a particular task ordering, and schedule adjustment, which considers different changes to task ordering in order to improve the result of the ensuing polishing step. The outcome of each polishing step is suitable for transmission to mission systems for execution.

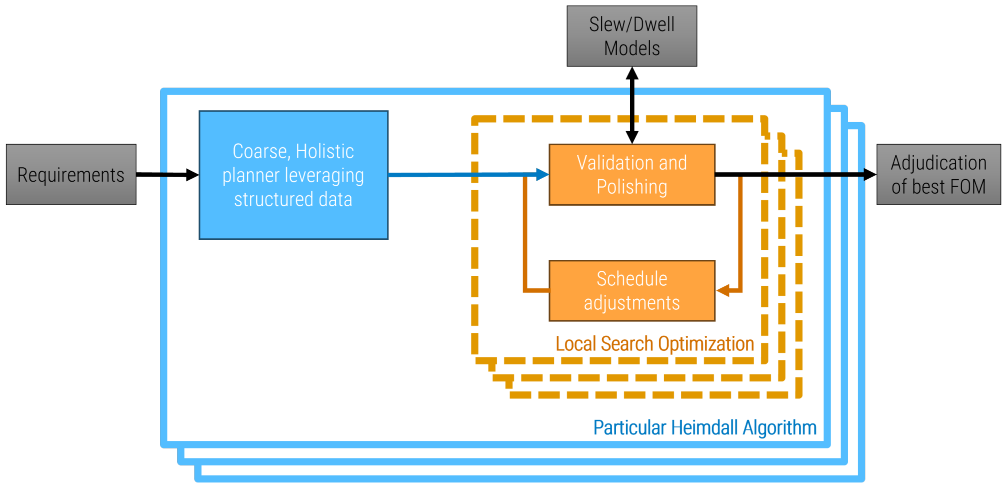

Slew/Dwell Models
Requirements
Coarse, Holistic planner leveraging structured data
Validation and Polishing
Schedule adjustments
Local Search Optimization
Particular Heimdall Algorithm
Adjudication of best FOM

Fig. 3: Heimdall Algorithm Architecture

5. RESULTS

We present the results of some numerical experiments. We compare the uninitialized RL trained scheduler NN to the TSPTW-initialized RL trained NN scheduler.

Figure 4 shows the training reward over 5000 episodes of RL training without any initialization. The averaged results of 5 runs with different seeds are shown. In Fig. 5, the same metrics are shown for the TSPTW-initilaized NN. The TSPTW-initialized NN achieves better performance faster and performs much better. The initialization brings the NN to a level of performance higher than after 5000 episodes of training the uninitialized NN scheduler. Note that the variability in the performance is in large part due to the training process sampling from the distribution over next objects to visit rather than taking the largest probability option; this allows for better exploration of the solution space. In deployment, the latter method is used to exploit rather than explore scheduling performance.

The TSPTW-initialized Scheduler gets to within 5% of the optimal schedule after 20,000 training episodes. Finally, the TSPTW-initialized Scheduler incurs no recurrence violations; training process increases efficiency by reducing slew time.

6. CONCLUDING REMARKS

We established proof of concept for a method of leveraging ML for complicated SDA scheduling problems that takes advantage of previous work. We gain the flexibility and data-driven methodology of ML while employing the model-based approaches developed over the years.

Unlike with custom algorithms, Machine Learning (ML) allows domain experts to tweak algorithms without algorithm experts, especially as problems evolve and new challenges arise. This methodology allows us to still leverage previous custom in-house and external algorithms tailored for similar problems. We use supervised ML to initialize an NN scheduler to mimic an existing algorithm for a similar problem and RL to update and tailor the NN scheduler to the problem under study.

Future work will focus on the identification of pairs of initialized and full problems to which this methodology may be applied. These may include

- Constant slew times $\rightarrow$ Time varying or stochastic slew times
- Constant dwell times $\rightarrow$ Time varying or stochastic dwell times
- Periodic constraints on observation timing $\rightarrow$ Observation timing based on track accuracy constraints

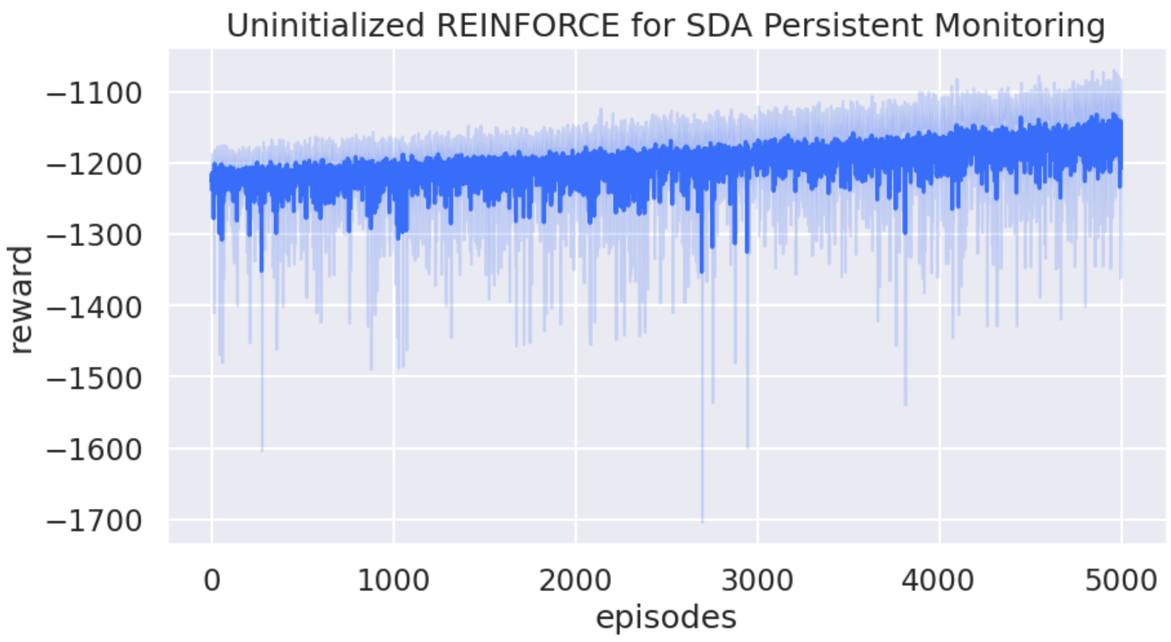


Fig. 4: Performance improvement during RL from uninitialized NN Scheduler.

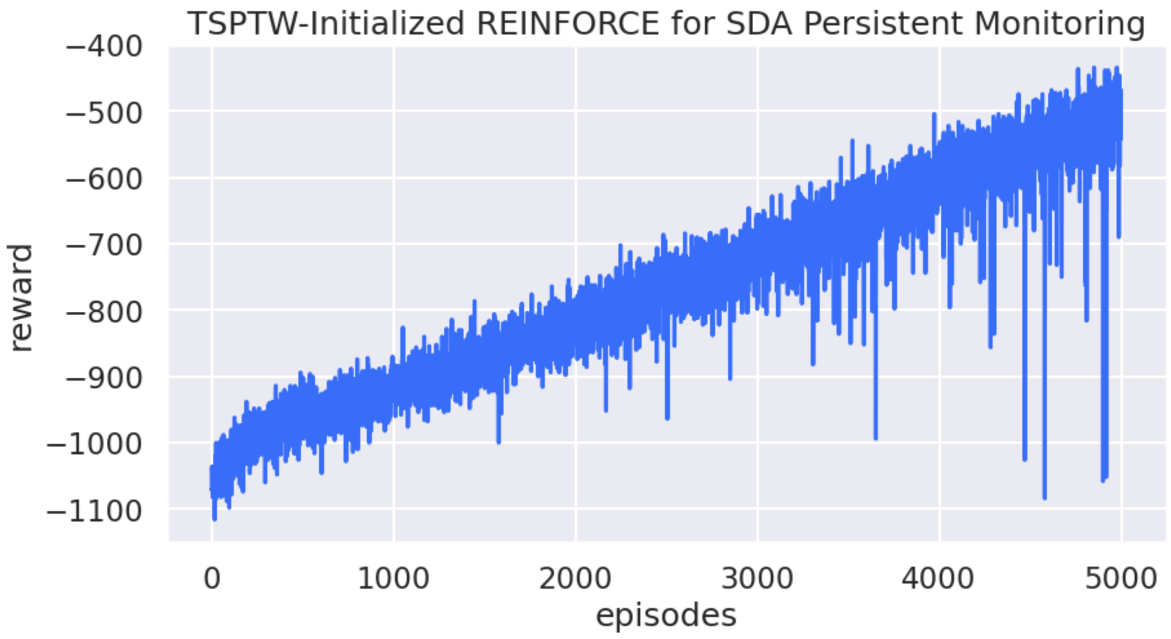


Fig. 5: Performance improvement during RL from TSPTW-intialized NN Scheduler.

- Scheduling to minimally satisfy scheduling requirements $\rightarrow$ Scheduling to employ slack time for additional collections
- Scheduling based on a memoryless slew times $\rightarrow$ Scheduling for slew times with wind up or other state considerations
- Scheduling for fixed dwell times $\rightarrow$ Scheduling for dwell times that depend on energy expended (e.g., radar), environmental factors, rate tracking fidelity/track uncertainty, etc.

REFERENCES

[1] N. Dhingra, C. DeJac, A. Herz, T. Wolf, and B. Jones, "Maximizing the utility of non-traditional sensor network data for SDA," in *Proceedings of the 2021 Advanced Maui Optical and Space Surveillance Technologies Conference*, (Wailea, HI), 2021.

[2] N. K. Dhingra, C. DeJac, J. Neel, A. Herz, T. Wolf, and B. Jones, "The impact of orbit accuracy-based tasking on sensor network efficiency," in *Proceedings of the 2022 Advanced Maui Optical and Space Surveillance Technologies Conference*, (Wailea, HI), 2022.

[3] N. K. Dhingra, C. DeJac, A. Herz, and R. Green, "Space domain awareness sensor scheduling with optimality certificates," in *Proceedings of the 2023 Advanced Maui Optical and Space Surveillance Technologies Conference*, (Wailea, HI), 2023.

[4] R. Stottler and A. Li, "Automatic, intelligent commercial SSA sensor scheduling," in *Proceedings of the 2019 Advanced Maui Optical and Space Surveillance Technologies Conference*, vol. 1, (Wailea, HI), 2019.

[5] R. Stottler, "Improved space surveillance network (ssn) scheduling using artificial intelligence techniques," *Proceedings of the 2015 Advanced Maui Optical and Space Surveillance Technologies Conference*, 2015.

[6] R. Linares and R. Furfaro, "An autonomous sensor tasking approach for large scale space object cataloging," in *Proceedings of the 2017 Advanced Maui Optical and Space Surveillance Technologies Conference*, (Wailea, HI), pp. 1–17, 2017.

[7] P. Hägg, "Optimal sensor planning for SSA using system identification concepts," in *Proceedings of the 2022 Advanced Maui Optical and Space Surveillance Technologies Conference*, (Wailea, HI), 2022.

[8] J. Ferreira, I. Hussein, J. Gerber, and R. Sivilli, "Optimal SSN tasking to enhance real-time space situational awareness," in *Proceedings of the 2016 Advanced Maui Optical and Space Surveillance Technologies Conference*, (Wailea, HI), pp. 20–23, 2016.

[9] R. S. Erwin, P. Albuquerque, S. K. Jayaweera, and I. Hussein, "Dynamic sensor tasking for space situational awareness," in *Proceedings of the 2010 American control conference*, pp. 1153–1158, IEEE, 2010.

[10] A. Herz, B. Jones, E. Herz, D. George, P. Axelrad, and S. Gehly, "Heimdall system for MSSS sensor tasking," in *Proceedings of the 2015 Advanced Maui Optical and Space Surveillance Technologies Conference*, (Wailea, HI), 2015.

[11] A. Herz, E. Herz, K. Center, D. George, P. Axelrad, S. Mutschler, and B. Jones, "Utilizing novel non-traditional sensor tasking approaches to enhance the space situational awareness picture maintained by the space surveillance network," in *Proceedings of the 2016 Advanced Maui Optical and Space Surveillance Technologies Conference*, (Wailea, HI), 2016.

[12] J. Aristoff, N. K. Dhingra, A. Ferris, A. Hariri, J. Horwood, A. Larson, T. Lyons, J. Shaddix, N. Singh, and K. Wilson, "Non-traditional data collection and exploitation for improved GEO SSA via a global network of heterogeneous sensors," in *Proceedings of the 2018 Advanced Maui Optical and Space Surveillance Technologies Conference*, (Wailea, HI), 2018.

[13] Y. Wang, J. Su, I. I. Hussein, and A. Wyglinski, "Price-based information routing in complex satellite networks for space-based situational awareness," in *Proceedings of the 2009 Advanced Maui Optical and Space Surveillance Technologies Conference*, (Wailea, HI), 2009.

[14] C. Frueh, H. Fiedler, and J. Herzog, "Realistic sensor tasking strategies," in *Proceedings of the 2016 Advanced Maui Optical and Space Surveillance Technologies Conference*, (Wailea, HI), 2016.

[15] S. N. Paul, B. D. Little, and C. Frueh, "Detection of unknown space objects based on optimal sensor tasking and hypothesis surfaces using variational equations," *J. Astronaut. Sci.*, vol. 69, no. 4, pp. 1179–1215, 2022.

[16] B. D. Little and C. E. Frueh, "Multiple heterogeneous sensor tasking optimization in the absence of measurement feedback," *J. Astronaut. Sci.*, vol. 67, no. 4, pp. 1678–1707, 2020.

[17] B. D. Little and C. Frueh, "SSA sensor tasking: comparison of machine learning with classical optimization methods," in *Proceedings of the 2018 Advanced Maui Optical and Space Surveillance Technologies Conference*, (Wailea, HI), pp. 1–17, 2018.

[18] R. Linares and R. Furfaro, "Dynamic sensor tasking for space situational awareness via reinforcement learning," in *Proceedings of the 2016 Advanced Maui Optical and Space Surveillance Technologies Conference*, (Wailea, HI), pp. 1–10, 201.

[19] G. Gutin, A. Yeo, and A. Zverovich, "Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP," *Discrete Applied Mathematics*, vol. 117, no. 1-3, pp. 81–86, 2002.

[20] G. Laporte, "Fifty years of vehicle routing," *Transportation science*, vol. 43, no. 4, pp. 408–416, 2009.

[21] B. Kallehauge, J. Larsen, O. B. Madsen, and M. M. Solomon, *Vehicle routing problem with time windows*. Springer, 2005.

[22] D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou, and N. Vathis, "Efficient heuristics for the time dependent team orienteering problem with time windows," in *Applied Algorithms: First International Conference, ICAA 2014, Kolkata, India, January 13-15, 2014. Proceedings 1*, pp. 152–163, Springer, 2014.

[23] J. Ruiz-Meza and J. R. Montoya-Torres, "A systematic literature review for the tourist trip design problem: Extensions, solution techniques and future research lines," *Operations Research Perspectives*, vol. 9, p. 100228, 2022.

[24] A. M. Campbell and J. H. Wilson, "Forty years of periodic vehicle routing," *Networks*, vol. 63, no. 1, pp. 2–15, 2014.

[25] N. Christofides and J. E. Beasley, "The period routing problem," *Networks*, vol. 14, no. 2, pp. 237–256, 1984.

[26] M. L. Fisher and R. Jaikumar, "A generalized assignment heuristic for vehicle routing," *Networks*, vol. 11, no. 2, pp. 109–124, 1981.

[27] M. Gaudioso and G. Paletta, "A heuristic for the periodic vehicle routing problem," *Transportation Science*, vol. 26, no. 2, pp. 86–92, 1992.

[28] D. M. Ryan and B. A. Foster, "An integer programming approach to scheduling," *Computer scheduling of public transport urban passenger vehicle and crew scheduling*, pp. 269–280, 1981.

[29] C. Archetti, E. Fernández, and D. L. Huerta-Muñoz, "The flexible periodic vehicle routing problem," *Computers & Operations Research*, vol. 85, pp. 58–70, 2017.

[30] D. L. Huerta-Muñoz, C. Archetti, E. Fernández, and F. Perea, "The heterogeneous flexible periodic vehicle routing problem: Mathematical formulations and solution algorithms," *Computers & Operations Research*, vol. 141, p. 105662, 2022.

[31] L. Perron and V. Furnon, "Or-tools."

[32] V. Furnon and L. Perron, "Or-tools routing library."

[33] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[34] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[35] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. D. Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, H. Tan, and O. G. Younis, "Gymnasium: A standard interface for reinforcement learning environments," 2024.

[36] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.

[37] Hill, Keric, and Brandon A. Jones, "Turboprop version 4.0, 2009." Colorado Center for Astrodynamics Research.

A. HEIMDALL SOFTWARE ARCHITECTURE

The Heimdall solution is intended to support an operation staff as part of a wider workflow enabling Battle Management Command and Control (BMC2). It specifically occupies the functional role of optimizing sensor tasking across a large number of ground and space sensors to achieve overall SDA-related objectives. Heimdall interacts with other components of a wider architecture using machine-to-machine interfaces utilizing plug-ins that allow the specifics of those interfaces to be easily updated, or even to become compliant with completely different interoperability standards in different systems. This has already been demonstrated via installation of the capabilities in multiple customer systems. The primary interface to Heimdall is a web interface, accessible via standard browsers, through which all of the core administrative and operational features can be accessed.

Fig. 6: Heimdall Logo

One of the core features of the Heimdall solution is the ability to generate coordinated, optimized observation schedules for the full set of available ground and space-based sensors for SDA observations. Heimdall leverages Orbit Logic's STK Scheduler scheduling algorithms, Orbit Logic's Collection Planning and Analysis Workstation (CPAW) scheduling algorithms and tools, and domain-specific Orbit Logic algorithms tailored for SDA.

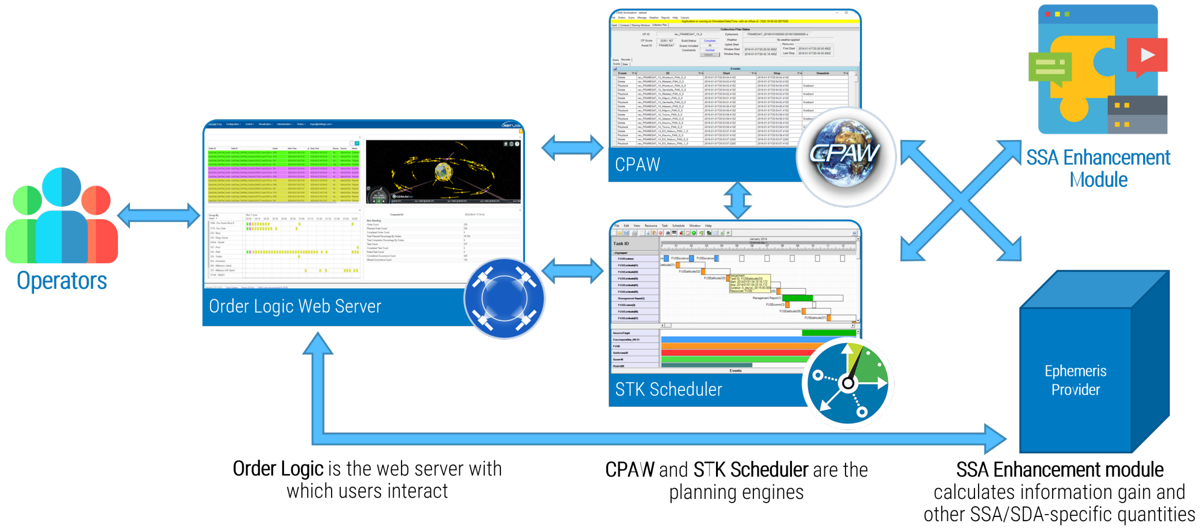

Operators
Order Logic Web Server
CPAW
STK Scheduler
SSA Enhancement Module
Ephemeris Provider
Order Logic is the web server with which users interact
CPAW and STK Scheduler are the planning engines
SSA Enhancement module calculates information gain and other SSA/SDA-specific quantities

Fig. 7: Heimdall System Architecture Diagram: Heimdall builds on and enhances mature Orbit Logic products to create optimized SSA/SDA sensor schedules Scheduling/Tasking Algorithms

STK Scheduler provides multiple scheduling algorithms as well as an algorithm builder tool, to define refined algorithms for specific needs. In the SDA configuration, algorithms are fed the list of SDA FOM-scored observation opportunities and use that list as the basis for generating a high value, valid, deconflicted, coordinated observation schedule. Heimdall calls the STK Scheduler algorithms using an available STK Scheduler STK Connect command via its TCP/IP API with string keyword-value pairs. The specific algorithm may be configured within Heimdall, but an option also exists to call an algorithm-builder-defined custom combination algorithm that computes solutions using multiple algorithms and returns the highest FOM-scoring solution. Earlier versions of the STK Scheduler algorithms were successfully demonstrated to CSpOC personnel as part of the SDA Software Suite from Analytical Graphics for a large scale SSN sensor tasking problem (10,000 objects, 24 hour schedule, 30 sensors), with optimized observation schedule solution time under 2 minutes.

CPAW has a similar set of algorithms for tasking schedule generation. Multiple algorithms are fed the SDA FOM-scored observation opportunities and iterated with high fidelity space sensor models to generate a high value, valid, deconflicted, coordinated observation schedule for all available space-based sensors. The nine available CPAW algorithms may be configured on or off via the Heimdall API, with the algorithm solution from the highest SDA FOM-scoring plan returned. CPAW scheduling algorithms are called via the available CPAW API using command strings delivered via TCP/IP interface. Scheduling results are saved directly to the Heimdall Object Catalog database, associated with applicable objects.

A.1 SDA-specific Figure-Of-Merit

Heimdall makes use of an SDA-specific Figure-of-Merit (FOM). The SDA FOM scores each observation opportunity based on inputs (such as predicted information gain) from the Task Prioritization component and other factors (such as computed object visual magnitude), time since last observation, orbit covariance, anomalous behavior rating, and more.

Each factor has an associated configurable weighting attribute to specify the importance of the FOM factor relative to other FOM factors. Weighting attributes may be set to any value, including 0 (ignored) and negative (penalty) values, allowing for virtually unlimited tuning of the scoring FOM.

Additionally, the FOM is split into object factors and search area factors (as well as common factors that apply to both), and the scores for objects and searches are normalized against each other. Lastly, configurable weighting factors allow for the importance of object observations vs. searches for new objects to be defined.

The SDA FOM is tightly coupled within the SDA versions of STK Scheduler and CPAW. All observation opportunities are automatically scored using the configured SDA FOM as part of the standard processing flow in both software tools.

In a future version of the architecture the SDA-specific FOM will also be made available via web interface for optional use by Tasked and Contributing sensors for their own local schedule optimization.

A.2 Value of Information-Based Tasking

Incorporating measures of information gain into space-object sensor tasking procedures provides a way to quantify the quality of candidate observation opportunities. Heimdall was updated to enable tasking is informed by metrics related to the expected state error covariance of a space-object at a desired epoch time. This feature generates the expected state covariance matrix at that time provided an initial state covariance matrix and a set of candidate measurements. In addition to intelligent tasking, this feature provides elevated operator awareness of the expected catalog state and the tasking algorithm's rationale.

Minimizing the size of the covariance matrix corresponds to maximizing the information gained with a measurement sequence. The user, in Heimdall, will add a "Final Orbit Accuracy" to the order and the planning software will plan to achieve it. This parameter is by default the volume of the covariance matrix, but other metrics can easily be configured. Heimdall provides the initial state and state covariance matrix of a space-object, the observing asset type and location (both ground-based and space-based observing assets are acceptable). A candidate measurement schedule is provided by the user which lists both a sequence of observation times, and the observing asset used per time.

Two renditions of the software were developed. The Extended Kalman filter (EKF) version sequentially updates the space-object's state covariance per measurement in the observation sequence. That is, for each candidate measurement in the sequence, the EKF algorithm evaluates a covariance matrix decrement, which is related to the expected information gained from said measurement. This decrement is then subtracted from the predicted covariance at the time of the measurement. The process is repeated for each measurement in the set. The TurboProp library is used to propagate the space-object state and state covariance between times in the measurement set. After all the measurements are processed, the state and state covariance matrix are propagated to a final epoch of interest, and this final covariance matrix is used to ensure the desired orbit accuracy is met [37]. The second version of the software uses an epoch-state filter formulation to calculate the final state covariance matrix. This formulation calculates a covariance matrix decrement in a batch-like formulation, forgoing the recursive procedure of the standard EKF.

A necessary component of the project was accurately including process noise in space-object dynamics modeling. Process noise is important in quantifying how much information is lost in propagating from measurement-to-measurement – or in other words, how much the covariance matrix grows between measurements. To do this, a new tool was developed in the TurboProp library for propagating a process noise transition matrix between candidate measurement times. This transition matrix was then incorporated into the standard EKF formulation and the epoch-state formulation of the software. The software was tested and validated on both ground-based and space-based sensor tasking scenarios.

A.3 External Plan Ingest

Heimdall was deployed and made available to external users via an Order Logic hosted machine configured for interfacing with leading commercial SSA operators (LeoLabs and Numerica, now Slingshot). Commercial SSA operator observation plans were retrieved and ingested by Heimdall, converted to ISSP format, and then utilized to show how commercial plans can inform Department of Defense (DoD) SSA sensor observation planning to meet DoD operational objectives, including meeting specific orbit accuracy goals.

A.4 Heimdall User Interfaces

Order Logic was developed as a user-facing interface for Orbit Logic's planning software application. The web application has previously been configured as the program-specific front end for both STK Scheduler and CPAW planning applications. In Heimdall, Order Logic is configured to interface with both the STK Scheduler, CPAW, and customized SDA planning engines, and has additionally been enhanced to provide overall workflow and automation control.

Providing an SDA-beneficial software automation framework for a distributed sensor network with worldwide non-traditional sites necessitated a web-enabled solution – one with the ability to monitor the state of space environment from many coordinated consoles and manage data flows in a highly configurable manner. As such, the web-based Graphical User Interfaces (GUIs) comprising the Heimdall solution are key to the overall operations concept.

One of the primary user features exposed through the web interface is visualization of the sensor tasking plans. Heimdall provides multiple ways for an operator to view, explore, and understand planned SDA tasking for ground and space sensors.

A configurable dashboard table view dynamically presents observations in time order, highlighting observations in progress (either in real-time and/or simulated time) and moving through the list of observations as time progresses. The presented list of observations can be filtered based on user preferences. The same Dashboard page provides a more global perspective in a 3D visualization pane. Driven by Cesium, this view is normally configured to run in real-time as a companion to the table view on the Dashboard, showing observations in an accurate graphical view as they occur throughout the collection of available sensors. The user may also select specific observations in the table view, and the Dashboard page Cesium 3D view automatically zooms in on the associated sensor resource and forwards to the time of the selected observation to display a static view of the specific observation geometry.

The Heimdall table and 3D views are driven by the latest object catalog database and associated planned observations saved within the object data there. The screenshot in Figure 2 shows the table view and associated configurable filter, along with the embedded 3D Cesium view and associated metrics.

A.5 Configuration Manager

The Configuration Manager component of Heimdall provides the ability for authorized users (administrators) to define and configure permissions for users, add and configure new SDA sensors, specify sensor downtime, specify optimization goals, review performance metrics, and perform other related setup and configuration functions. Changes made within Heimdall configuration pages are stored to the associated Heimdall database for use internally and/or used to send Application Programming Interface (API) configuration commands to some of Heimdall solution component applications.

A.6 Visibility Computations

At the start of the planning process, constrained access computations are performed for each valid sensor/object combination. Computations consider line-of-site visibility, lighting constraints (when applicable), sensor capabilities, sensor field-of-regard, object attributes, and any applicable object/sensor assignments and preferences and constraints. Because access computations for each object are independent of the access computations for other objects, these computations can be performed in parallel on many cores in order to speed computation time for large object catalogs.