Space Domain Awareness Sensor Scheduling with Optimality Certificates

Neil K. Dhingra, Cameron DeJac, Alex Herz, and Roderick Green
Orbit Logic, a Boecore Company

ABSTRACT

We present our framework for formally characterizing the quality of sensor scheduling algorithms for Space Domain Awareness (SDA) and analyze the optimality gaps achieved by the planning algorithms underpinning our SDA sensor scheduling software, Heimdall. Since characterizing scheduling algorithm quality in full generality is intractable due to the size and complexity of the SDA planning problem, we identify relevant and representative subproblems over which we can formally characterize algorithm quality. This framework uses complex, expensive, and slow methods – which would be impractical in an operational system – to generate high-quality sensor schedule benchmarks against which we compare the quality of sensor schedules planned by lightweight, fast, 'anytime' scheduling algorithms that *do* satisfy operational constraints such as those on runtime. We demonstrate that our Heimdall algorithms identify optimal or near-optimal solutions in several specific SDA scheduling problem instances and improve solution quality relative to baseline approaches while satisfying operational requirements. We identify complicated SDA scheduling problem instances that bely formal analysis and draw connections from them to tractable subproblems which are 'near' to them in the problem space. Finally, we discuss how the analyses enabled by this framework drives algorithm refinement and development. This framework enables the continued development and refinement of the Heimdall SDA sensor scheduling algorithms. Heimdall has been deployed in support of SDA sensor scheduling for several customers and Orbit Logic will provide the scheduling component for 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. In addition to complicating the design of algorithms that can plan feasible sensor schedules, it makes it difficult to formally characterize the quality of those schedules, even when such characterization is crucial to ensure the efficacy of deployed algorithms supporting operational systems. Here, algorithm performance or solution quality is quantified by an objective function or Figure of Merit (FOM) (e.g., total slew time) associated with a given sensor schedule as long as it meets sensing requirements (e.g., plan observations of all space objects of interest); sensor schedules with a lower objective function or FOM score are better. To facilitate analysis and development of deployed SDA scheduling algorithms, we have developed a framework wherein we formally characterize algorithm performance on relevant but tractable subproblems. In other words, we infer overall algorithm performance based on performance in representative special cases.

We have applied this framework to analyze and improve the quality of solutions generated by our Heimdall SDA sensor scheduling software [1, 2]. We demonstrate that our Heimdall algorithms identify optimal or near-optimal solutions in several specific SDA scheduling problem instances and improve solution quality relative to baseline approaches while satisfying operational requirements.

First, we identify several tractable subproblems of the full SDA scheduling problem which are, due to their size and/or complexity, amenable to analysis via integer linear programming techniques, convex relaxation, exhaustive search, and/or other formal methods. These formal methods provide lower bounds on the optimal FOM value or, for small enough problems, the optimal FOM value itself. We characterize solution quality over these subproblems and from that infer solution quality in more complicated problem instances. An analogy is evaluating a student – it is infeasible to characterize their performance in every possible situation, but it is possible to give them tests or coding exercises to evaluate them in a representative sample of scenarios and use that to infer an assessment of their mastery across the entire subject.

To analyze these specific subproblems, we employ complex, expensive, and slow methods which would be impractical in an operational system. We then compare the objective values of sensor schedules planned by Heimdall algorithms with the optimality bounds and sensor schedule objective value scores generated by the more comprehensive methods. Both sets of algorithms are designed around characteristics of the problem itself – revisit rate requirements, moving objects, time-varying slew times between targets, sensor keepout constraints, track accuracy requirements, etc. – but the algorithms used for analysis may be more computationally intensive. The Heimdall algorithms are designed for deployment so must obey operational considerations on runtime, memory, 'anytime' algorithm requirements, limited information sharing across different sites, etc.

We present the results of this analysis which indicates the Heimdall often achieves global optimal or near optimal solutions. When we identify cases in which Heimdall schedules are suboptimal, this analysis drives development to improve those algorithms. In particular, since Heimdall runs several algorithms in parallel and then selects the plan output with the best FOM, novel algorithms can be developed to target especially challenging classes of problems and they can run alongside the existing algorithms. Algorithm development may involve adjusting algorithmic parameters, developing new algorithm approaches, or leveraging specific pieces of the larger more expensive solvers within the deployed Heimdall algorithms.

This framework enables analysis of the value of Heimdall and other sensor scheduling problems for the SDA mission and facilitates the continued improvement of the Heimdall SDA sensor scheduling algorithms. Heimdall has been deployed in support of SDA sensor scheduling for several customers and Orbit Logic will provide the scheduling component for 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 [3, 4, 5, 6, 7, 8]. Of particular interest has been the tradeoff between sensor accuracy and the cost of sensing [6, 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 [4, 9, 10]. Others employ generalized nearest neighbor-based approaches [8, 11] or other meta-heuristics such as those based on pricing [12]. Often, slew time is neglected.

In [13], the authors pose SSA/SDA sensor tasking as an optimization problem and design an approach to solve it optimally. They apply Machine Learning methods to target this underlying problem in [14]. 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]. 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 [17]. 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 [18] can be used to design slew paths for multiple sensors and VRPs with time windows (VRPTWs) [19] 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 [20, 21], but do not typically deal with recurring tasks. Periodic VRPs (PVRPs) can be used to enforce revisit constraints [22, 23, 24, 25, 26] 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 [27, 28]; 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 then discuss the formulation of lower and upper bounds for that problem as well as subproblems on which relaxations for better lower bounds are tractable 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 [28, 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 reindex the assignments and start times per object for convenience.

2.2.3 Problem Statement

The full problem considered in this paper is

$$\begin{aligned}
\text{minimize} \quad & \sum_{i,j} d_i^j + t^j_{a_i, a_{i+1}} && && \text{(1a)}\\
\text{subject to} \quad & \tau^j_{i+1} \geq \tau_i + d_i^j(\tau) + t^j_{a_i,a_{i+1}}(\tau) && \forall i, j && \text{(1b)}\\
& \tau_i^j \in P_i^j && \forall i, j && \text{(1c)}\\
& \tau'_{i,k+1} \leq \tau'_{i,k} + r_i && \forall i, k. && \text{(1d)}
\end{aligned}$$

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.

Often the constraints are not trivial to satisfy. So we may relax them and replace the objective function (1) to form a satisfiability problem; for example,

$$\begin{aligned}
\text{minimize} \quad & \sum_{i,k} \max(0, \tau'_{i,k+1} - \tau'_{i,k} - r_i) && && \text{(2a)}\\
\text{subject to} \quad & \tau^j_{i+1} \geq \tau_i + d^j_{a_i}(\tau) + t^j_{a_i a_{i+1}}(\tau) && \forall i, j && \text{(2b)}\\
& \tau_i^j \in P_i^j && \forall i, j. && \text{(2c)}
\end{aligned}$$

Constraints (2b) and (2c) remain the same as (1b) and (1c). The objective (1a) is replaced by (2a) which is 0 when constraint (1d) is satisfied. Each component of (2a) is 0 if a recurrence constraint is satisfied and equal to the time over the target recurrence interval if the constraint is violated. While slew and dwell time is not explicitly penalized in (2), this problem is often used when the system is overtasked so minimizing slew and dwell time is implicit since it allows for better attempting to satisfy all revisit constraints.

3. RELAXATIONS IN TRACTABLE SUBPROBLEMS

3.1 Exact Solutions – Global Optima

Exhaustive search methods can identify optimal solutions in very small problem instances, but clearly, problems (1) and (2) are large, combinatorial problems. Since the decision variables must encode the allocation of tasks to sensors, the ordering of tasks, and the timing of tasks, the solution space grows combinatorially with the problem dimension. Exhaustively exploring the solution space is not a viable strategy.

Algorithms for Mixed Integer Linear Programming (MILP) offer clever methods for solving combinatorial problems similar to (1) and (2). However, MILP heuristics do not scale gracefully with the problem dimension and the complexity of these problems increase the size of their MILP representations. We note that TDTOPTW problems, which are a simplification of (1), are only approached using MILP methods for very small problems.

Global optimal solutions for these problems can only be expected for 'small enough' scenarios. The threshold for 'small' will depend on the complexity of the constraints that must be considered; a TSP scenario that is 'small enough' to solve optimally in a reasonable amount of time will contain many more cities than a TDTOPTW that is 'small enough' to solve optimally in a similar amount of time.

3.2 Upper Bounds on Solution Quality – Worse than Global Optima

The operator's priority is not to identify the global optimal or best solution, but to identify a valid schedule that is good enough. To this end, many intuitive metaheurstics may be employed, such as greedy algorithms, nearest neighbor or generalized nearest neighbor methods, local search heuristics, and more. More sophisticated methods, such as MILP algorithms, can be used here by imposing problem structure using conservative assumptions. For example, imagine a TSP where the slew/travel times are time dependent. This problem could be simplified by always assuming worst-case slew times in order to plan a valid schedule. There may be more slack in the schedule than necessary, but the schedule itself would be valid.

Any of these heuristics target the creation of a plan that is valid (i.e., it obeys physical constraints such as limits on slew time) and that does its best to accomplish operational requirements (e.g., those on revisit rates). A valid schedule will satisfy the constraints of (1) or (2), but may not optimize the objective function. As such, they represent an *upper bound* on the global optimal value.

Due to the difficulty of the problem, most work on sensor scheduling focuses on heuristics to identify upper bounds. Progress in this area is measured by how much better schedules can be, i.e., how low the upper bound can be made. However, from just valid solutions that represent upper bounds on optimality, it is hard to know how much room for improvement (if any) there is.

3.3 Lower Bounds on Solution Quality – Limits on Global Optima

Complex constraints in the original problem may be relaxed to form a problem that is more easily solved and whose optimal value provides a *lower bound* on the optimal value. Lower bounds on solution quality are often not as well studied because they are not as operationally relevant. The plans associated with them may not be valid and so cannot be used. However, although their utility is limited for operators, these lower bounds are useful for those designing planning and scheduling algorithms because they help identify where operationally relevant heuristic algorithms are doing well and where there may be room for improvement. For example, if an upper bound meets a lower bound, the upper bound schedule is optimal! If the bounds are close, there is little room for improvement and if the bounds are very different, there *may* be room for improvement.

Consider the time-varying slew time TSP example from the prequel. If one instead assumed the *best* possible slew time, the resulting solution would likely be invalid. However, the objective value associated with that solution would be lower than the optimal value for the original problem. It might represent an objective function value that is unachievable, but it would put a bound on what the achievable values for the objective function are. Concretely, if all the desired tasks would not fit in a schedule with the best slew times always assumed, the desired tasks will not fit in any schedule that use real slew times.

Relaxed problems can take several forms. Often, the study of MILPs is concerned with convex relaxations that relax binary or integer valued variables (a source of nonconvexity) into bounded, real-valued variables. Relaxations to other combinatorial problems that are simpler may also be employed.

3.4 Tractable Subproblems

By focusing on particular components of the full SSA planning problem (1), we can relax it to problems that are well-studied or approachable with MILP algorithm solvers.

3.4.1 Vehicle Routing Problem with Time Windows

By focusing on problem (1) without constraints (1d), we obtain a VRPTW. This entails scheduling a network of sensors to observe a set of objects at some point during an observability window. This problem is well-studied and there is a wide set of literature concerning algorithms for efficiently solving it [19, 18].

3.4.2 Periodic Vehicle Routing Problem

By focusing on problem (1) without constraint (1c), we obtain something similar to a PVRP. This problem is also well-studied [22, 18], but the notion of periodicity in that literature – concerning service on disjoint days – do not map exactly to constraints (1d) and the need to consider slewing between 'days'.

To relax this problem, we design a MILP that, to the best of our knowledge, is unique. To simplify exposition, we consider a single sensor, but the framework can easily be extended to a setup with multiple, heterogeneous sensors. We partition the planning period into T_ρ distinct subperiods ρ_k of length $\bar{\rho} \geq \max_i r_i$ and consider a discretized and relaxed version of constraint (1d): in each set of $r_i^\rho := \text{floor}(r_i/\bar{\rho})$ consecutive subperiods, there must be at least one observation on object o_i. This is a similar, but less stringent, constraint than the original.

We define the binary decision variables $z_{i,k}$ which denotes if object o_i is observed in subperiod ρ_k, $x_{ij,k}$ which denotes whether the sensor slews from object o_i to o_j during subperiod ρ_k, $x^b_{ij,k}$ which denotes if the sensor slews from object o_i to o_j between subperiods ρ_k and ρ_{k+1}, where $x_{\cdot i,-1}$ which denotes whether object o_i is the first object observed and $x^b_{i\cdot,T_\rho}$ which denotes if object o_i is the last object observed.

$$\text{minimize} \quad \sum_{i,k} d_i z_{i,k} + \sum_{i,j,k} t_{a_i,a_{i+1}} x_{ij,k} \tag{3a}$$

$$\text{subject to} \quad \sum_{i} d_i z_{i,k} + \sum_{i,j} t_{a_i,a_{i+1}} x_{ij,k} + 0.5 \sum_{i,j} x^b_{ij,k-1} + 0.5 \sum_{i,j} x^b_{ij,k} \leq \bar{\rho} \qquad \forall k \tag{3b}$$

$$\sum_{k \in [K, K+r_i^\rho)} z_{i,k} \geq 1 \qquad \forall i, K \tag{3c}$$

$$\sum_{i,j} x^b_{ij,k} = 1 \qquad \forall k \tag{3d}$$

$$z_{i,k} \leq \sum_j x_{ij,k} + \sum_j x^b_{ij,k} \qquad \forall i,k \tag{3e}$$

$$\sum_j x_{ij,k} = \sum_j x_{ji,k} \qquad \forall i,k. \tag{3f}$$

The objective (3a) is the same as (1). Constraint (3b) ensures that no subperiod is overallocated. Constraint (3c) enforces the relaxation of the original revisit constraint (1d). Constraints (3d), (3e),and (3f) are all TSP constraints on the flow. Finally, subtour elimination constraints are included and solved for with row generation.

Clearly, constraint (3c) is a necessary condition to enforce constraint (1d). However, because actually satisfying the constraint depends on the timing of the tasks *within* the subperiods, it is not a sufficient condition. Moreover, since with this relaxation, the formulation offers no incentive to add multiple tasks on an object to the same subperiod, so we can consider only whether the sensors view an object during a subperiod and not how many times they would view the object during the subperiod. This makes (3) a relaxation of (1).

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. 1.

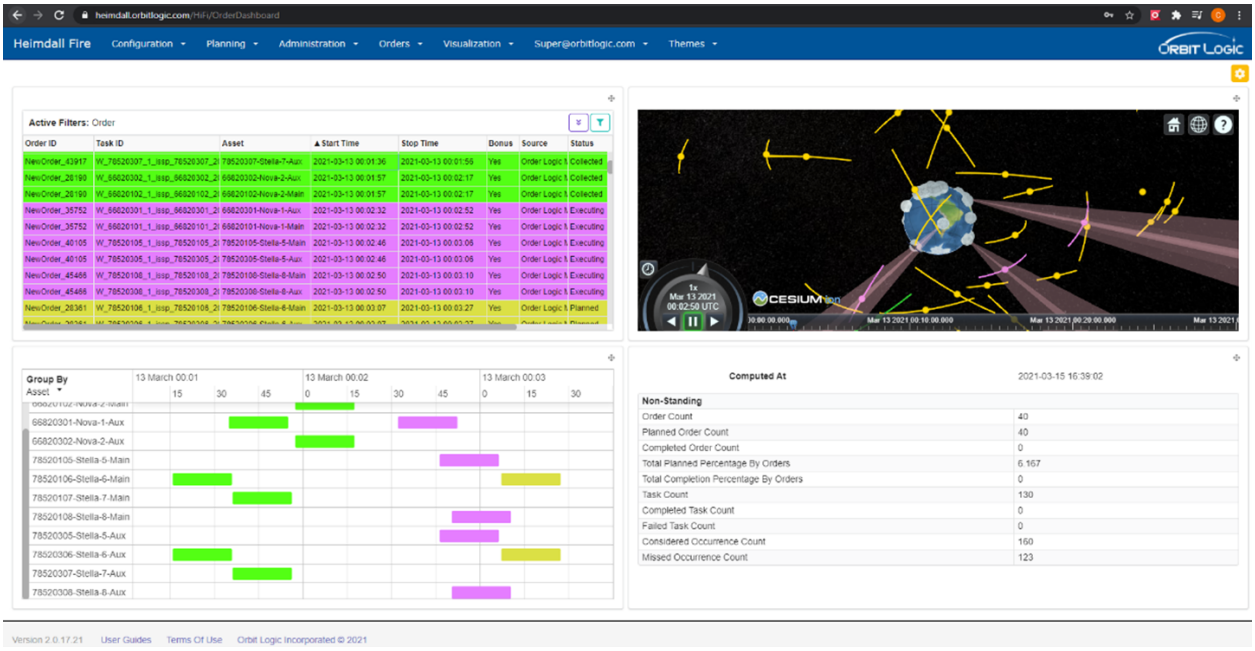


Fig. 1: Heimdall Dashboard Page with the light color theme

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. 2. 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.

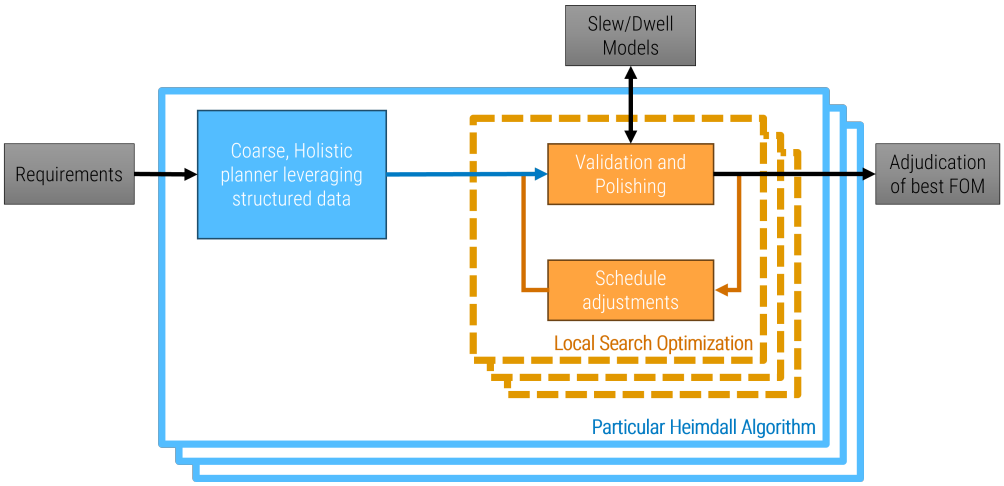


Fig. 2: Heimdall Algorithm Architecture

5. RESULTS

We present the results of some numerical experiments.

5.1 Comparison with Lower Bounds

In scenarios similar to the VRPTW when the problem size is small enough to solve a VRPTW to completion, it may be used as the initial solver in the architecture outlined in Section 4.

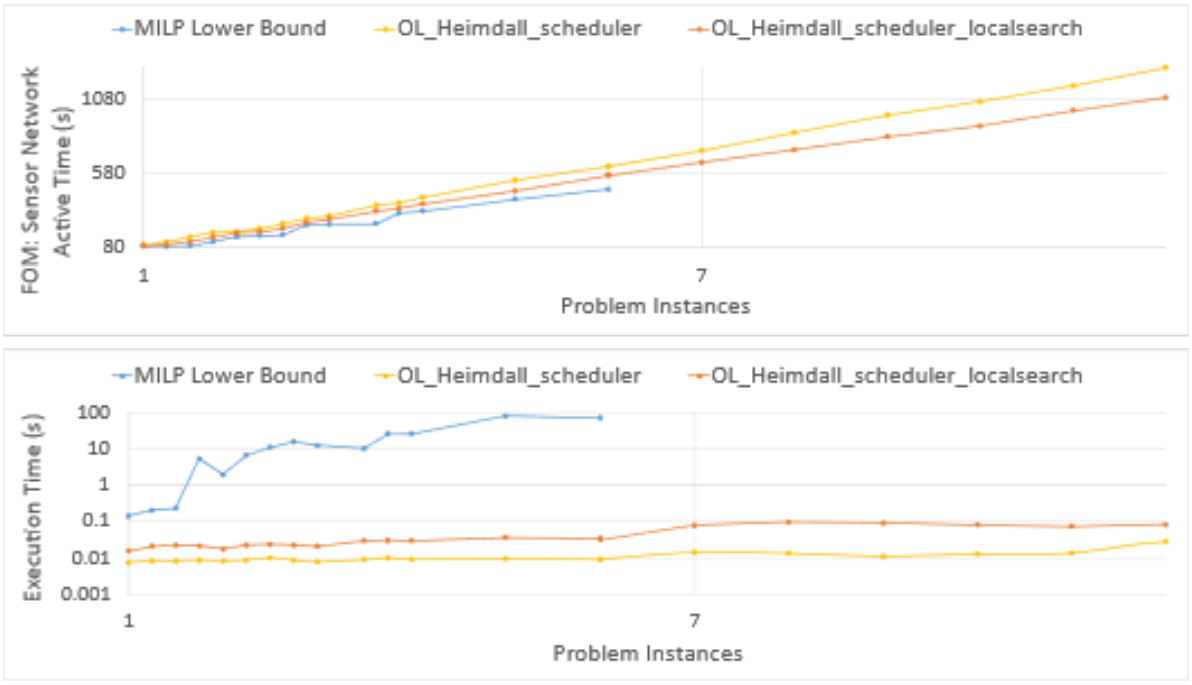


Fig. 3: Performance relative to the lower bound.

We also compared the results of the MILP relaxation in Section 3.4.2. We considered a 24 hour scenario with 10 objects each needing regular revisits at different intervals with different dwell times. The Heimdall scheduler created a valid schedule with the same number of collects as the solution to (3).

We compare Heimdall to the lower bound MILP solution in Fig. 3. The tailored Heimdall algorithm was within 17.6% (33.8%) of the lower bound after (before) polishing. However, Heimdall took $3--4$ orders of magnitude less time to complete than it took to solve the lower bound problem using Google OR-Tools [29]. The difference in computation time is larger for scenarios with larger numbers of objects. We note that for moderate numbers of objects, the MILP solver does not converge after several minutes or hours. Using a commercial MILP solver would certainly improve execution time, but those products are very expensive, and they will also fall victim to the curse of dimensionality eventually since these problems are NP Hard.

From the lower bound analysis, we have a certificate on schedule optimality and from timing, we validate operational relevance. Moreover, we note that on even smaller problems where the lower bound schedules were feasible – thereby certifying that the *global optimal* solution had been obtained, the Heimdall algorithm also matched it and identified the optimal solution.

Finally, we emphasize that the tailored Heimdall algorithm's performance did not start so close to the lower bound. Deriving the lower bound and comparing the schedules that resulted from the lower bound MILP drove the development that drove the improvement of the algorithm and the schedules it produces.

5.2 Larger Scenario

Although the formal lower bound analysis is expensive enough that it can only be applied to small problems, the development it drove yields results for much larger problems as well.

We consider a 4 hour scenario with 90 objects each needing regular revisits at different intervals with different dwell times. The lower bound algorithms could not solve this problem. However, we display Heimdall's performance relative to several other benchmark algorithms.

We consider a greedy generalized nearest neighbor algorithm, which chooses its path based on minimizing a weighted sum of slew time and time until the revisit interval constraint would be violated. We also use a lookahead algorithm that computes a TSP path between the next few objects that may be observed before the revisit interval for *any* object's revisit rate constraint would be violated. Finally, we include an Earliest Deadline First (EDF) algorithm from literature on a similar problem of recurring tasks that do not consider slew time [30].

These algorithms do not target the minimum active time FOM that we consider in this paper. Rather, they aim to fill the schedule with as many tasks as possible. To make the comparison fair, we study the performance of these algorithms in the range where the entire schedule must be filled to satisfy the revisit requirements (1d). We do this by increasing the dwell time required by each task so that more of the schedule must be active.

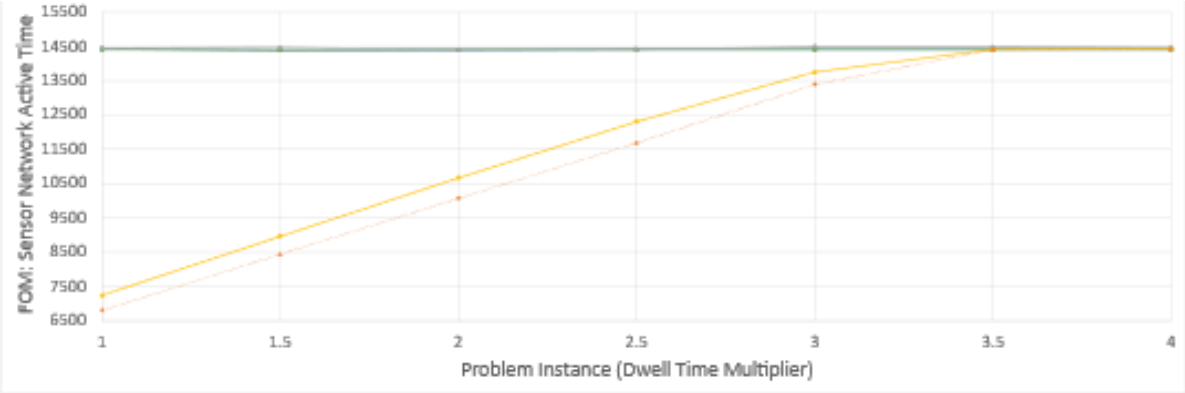


Fig. 4: Performance relative to the lower bound.

Fig. 4 compares the FOM for the schedules generated by the baseline algorithms and that generated by Heimdall. Clearly, Heimdall performs much better. However, this is unsurprising as the baseline algorithms use as much active time as possible.

Fig. 5 compares how often the algorithms' schedules violate the revisit requirement (1d). Since all the algorithms are designed to target this requirement, this is a fair comparison. Again, Heimdall performs much better, incurring many fewer violations as the problem gets larger.

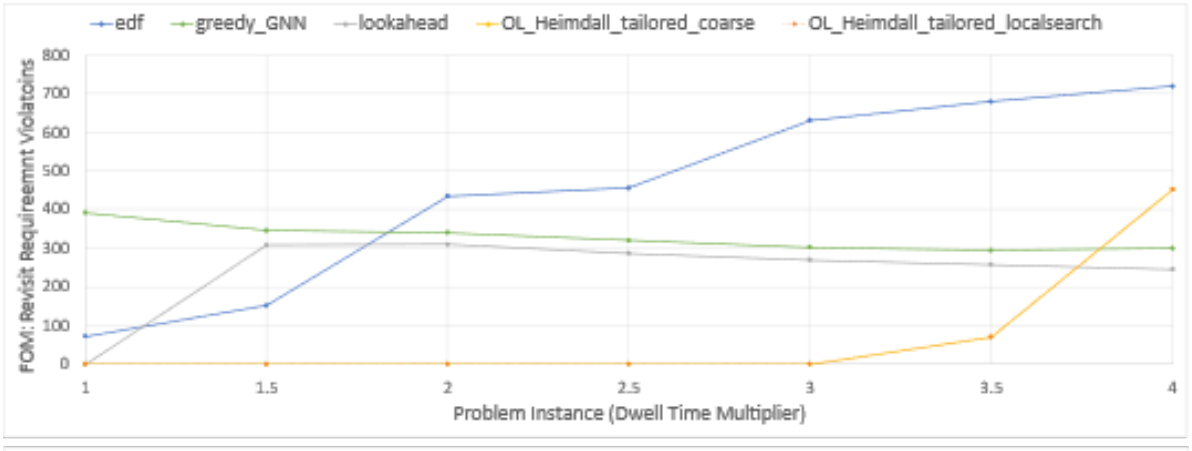
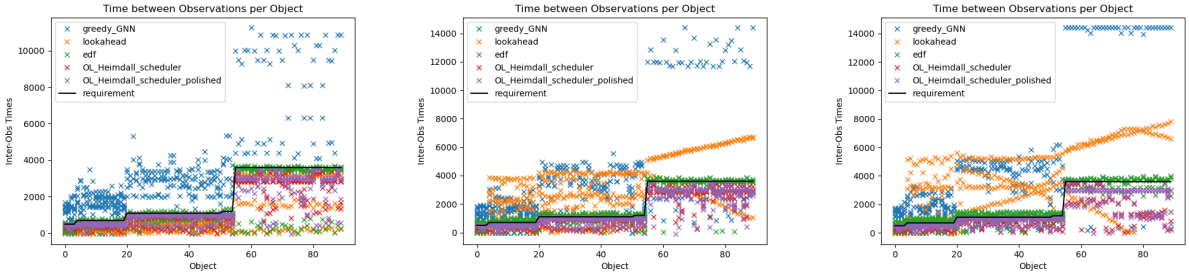


Fig. 5: Performance relative to the lower bound.



(a) Baseline Dwell Times (b) 2x Dwell Times (c) 3x Dwell Times

Fig. 6: Time between collects for several scheduling algorithms as the planning problem becomes more difficult due to longer dwell times.

Fig. 6 shows the time between successive tasks on an object relative to the requirement. Clearly, the Heimdall algorithms perform the best. While some of the other approaches can find a valid schedule in the baseline scenario, they no longer find valid solutions once the problem is made more difficult by inflating the dwell times required.

The increasing number of constraint violations for Heimdall as the dwell time multiplier grows above 3.5 begs the question: are the baseline algorithms better in those problem instances? However, this problem is not simple to answer. While the number of violations is a good metric when schedules nearly satisfy all constraints, it provides insufficient insight when there are many violations. However, consider EDF and the greedy GNN schedules in Fig. 6b; EDF has more violations, but they are all small. Would the operator prefer more small violations (e.g., fifty one-second violations) or fewer large violations (e.g., one fifty-second violation)? This is a question for the operator and the algorithm developers should design a FOM to target their expressed priorities;

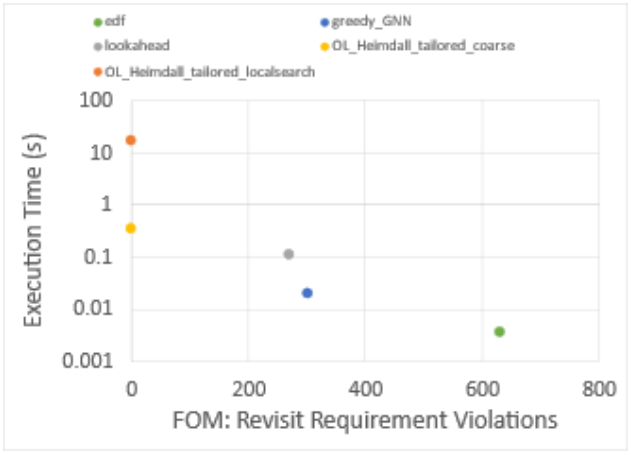


Fig. 7: Algorithms compared in terms of revisit constraint violations and execution time.

however, while this is an important problem
that we do address, it is beyond the scope of this paper.

Finally, we can consider all the algorithms in terms of the two meta-objectives of revisit requirement violations and execution time. All algorithms (besides Heimdall with polishing) took less than half a second to run. Fig. 7 shows this comparison. All the algorithms are Pareto optimal, so it is for the operator to decide which approach to employ. However, we note that Heimdall without polishing – which incurs no violations and requires less than a half-second to run, appears to be a promising option.

Of course, polishing also had a positive effect on the schedule produced by Heimdall. After the initial solver, which took 0.38 seconds to run, all requirements were met with 1274 tasks. After polishing, which took 21.81 seconds total, a compliant schedule was returned with only 1200 tasks; a 5.9% reduction in tasking. Whether this reduction in active time is worth the longer run time is up to the operator.

6. CONCLUDING REMARKS

We established a framework for formal analysis of the complicated SSA/SDA sensor scheduling problem. We identified and formulated key problems that facilitate relaxations of the original problem in order to form lower bounds for the original problem. We showed how this analysis informs the development of our planning and scheduling algorithms and illustrated their efficacy on a representative problem instance.

Future work will focus on the identification of additional tractable subproblems to facilitate further analysis. Of particular interest is the formulation of an orienteering problem whose rewards correspond to a track accuracy objective over time, and the formal analysis of schedule robustness to collect failure or other uncertain events.

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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.
[12] 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.
[13] C. Frueh, H. Fiedler, and J. Herzog, "Realistic sensor tasking strategies," 2016.
[14] 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.
[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," *The Journal of the Astronautical Sciences*, 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," *The Journal of the Astronautical Sciences*, vol. 67, no. 4, pp. 1678–1707, 2020.
[17] 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.
[18] G. Laporte, "Fifty years of vehicle routing," *Transportation science*, vol. 43, no. 4, pp. 408–416, 2009.
[19] B. Kallehauge, J. Larsen, O. B. Madsen, and M. M. Solomon, *Vehicle routing problem with time windows*. Springer, 2005.
[20] 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.
[21] 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.
[22] A. M. Campbell and J. H. Wilson, "Forty years of periodic vehicle routing," *Networks*, vol. 63, no. 1, pp. 2–15, 2014.
[23] N. Christofides and J. E. Beasley, "The period routing problem," *Networks*, vol. 14, no. 2, pp. 237–256, 1984.
[24] M. L. Fisher and R. Jaikumar, "A generalized assignment heuristic for vehicle routing," *Networks*, vol. 11, no. 2, pp. 109–124, 1981.
[25] M. Gaudioso and G. Paletta, "A heuristic for the periodic vehicle routing problem," *Transportation Science*, vol. 26, no. 2, pp. 86–92, 1992.
[26] 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.
[27] C. Archetti, E. Fernández, and D. L. Huerta-Muñoz, "The flexible periodic vehicle routing problem," *Computers & Operations Research*, vol. 85, pp. 58–70, 2017.
[28] 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.
[29] L. Perron and V. Furnon, "Or-tools."
[30] K. Jeffay, D. F. Stanat, and C. U. Martel, "On non-preemptive scheduling of periodic and sporadic tasks," in *IEEE real-time systems symposium*, pp. 129–139, US: IEEE, 1991.
[31] 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. 8: 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.

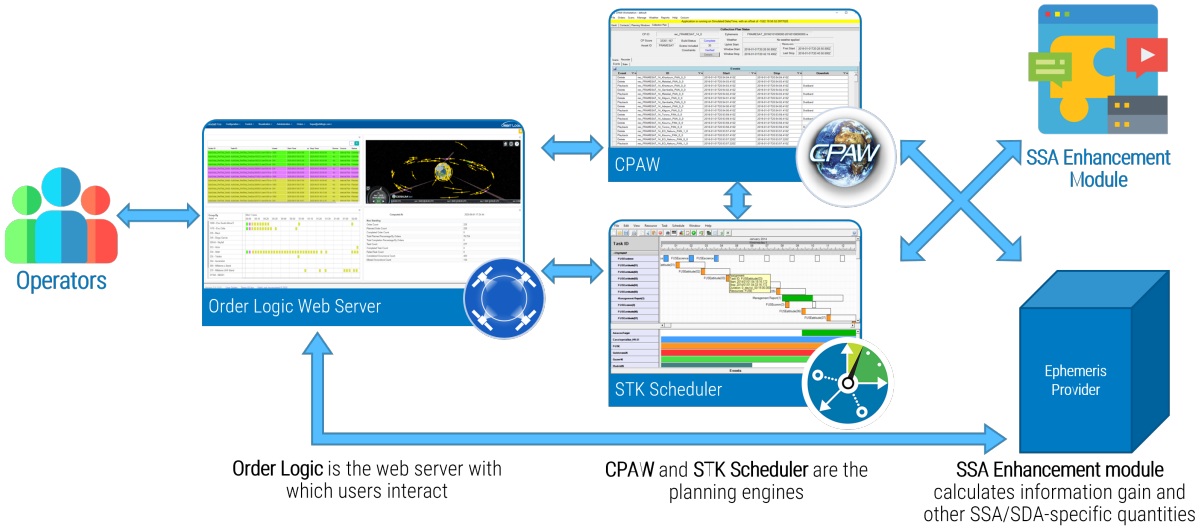


Fig. 9: Heimdall System Architecture Diagram: Heimdall builds on and enhances mature Orbit Logic products to create optimized SSA/SDA sensor schedules Scheduling/Tasking Algorithms

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 [31]. 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 1 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.