Automatic, intelligent commercial SSA sensor scheduling

Richard Stottler, Alan Li
Stottler Henke Associates, Inc.

ABSTRACT

There are close to 20,000 cataloged manmade objects in space, the large majority of which are not active, functioning satellites. These are tracked by phased array and mechanical radars and ground and space-based optical telescopes, collectively known as the Space Surveillance Network (SSN). Meanwhile there are many hundreds of commercially owned SSA sensors scattered across over 100 sites that produce high quality data that should be taken advantage of by government and industry decision-makers to improve their space situational awareness (SSA). This paper discusses scheduling issues and algorithms applied to optimized SSA sensor scheduling, including commercial SSA sensors and presents the surprisingly significant SSA sensing capacity represented by commercial SSA sensors. Included are specific, quantitative results from different scheduling experiments, showing what kind of performance is both possible and should be expected.

1. PROJECT GOALS

Ultimately, by providing automatic intelligent scheduling software, we are striving to ensure that the best Space Situational Awareness (SSA) data from commercial SSA sensors is provided to the relevant decision-makers so that they can take best advantage of the very large number of very capable such sensors. In support of this ultimate goal, we determined which use cases were both possible and beneficial for government use, worked out high-level integration concepts, developed and implemented commercial SSA sensor optimization algorithms for various situations, and determined the capabilities and capacities of SSA sensors. The use cases and timescales for the commercial SSA sensors, and thus for the scheduling algorithm, include a 24-hour schedule, catalog maintenance (maintaining orbital parameters, searching for new objects, and finding newly lost objects), space object identification (SOI) Information, and quick reaction sensing (i.e. tens of seconds to a few minutes).

2. COVARVIANCE AND COMPLEMENTARY OBSERVATIONS

An important function of both the government's SSN sensors and commercial industry's SSA sensors is maintaining accurate orbital track information on all objects orbiting the earth. Of course, all sensor have measurement error, so any specific sensor observation has an associated error volume. Since these measurement uncertainties are often captured in a covariances matrix, they are also referred to as "covariances" or "covariance volume". For the purpose of accurately tracking space objects, not all sensor observations are equivalent. In addition to the obvious fact that different specific sensors will have different resolutions, sensitivities, and measurement errors, the specific geometry and phenomenology will have a large impact. For example, radar systems tend to be very accurate in range but not as accurate in angle. This leads to a "pancake-shaped" error volume (where a line from the radar to the object being sensed is perpendicular to the radius of the "pancake". Meanwhile, optical systems, as singular passive sensors, are not very accurate in range but very accurate in angle. This leads to "cigar-shaped" error volumes where the line from the optical sensor to the object being sensed forms the central axis of the "cigar". These are both illustrated in Fig. 1 below. Additionally, because of orbital dynamics and orbital propagation, immediately after the sensing event, the error volumes become distorted over time. This is because of the nonlinear effects of Earth's gravitational field, such as objects closer to the earth orbiting faster than objects further away. It also means that more accurate tracking tends to occur with orbital diversity, making sensing observations at various different locations of an object's orbit. When using the same sensor phenomenology, this often means having sensor observations very roughly 90 or 270 degrees out of phase (i.e. not at the same place in the orbit or on the near-opposite side of the earth). The measurements do not need to be precisely 90 degrees apart in the orbit. 70% of the benefit is realized at 45 degrees.

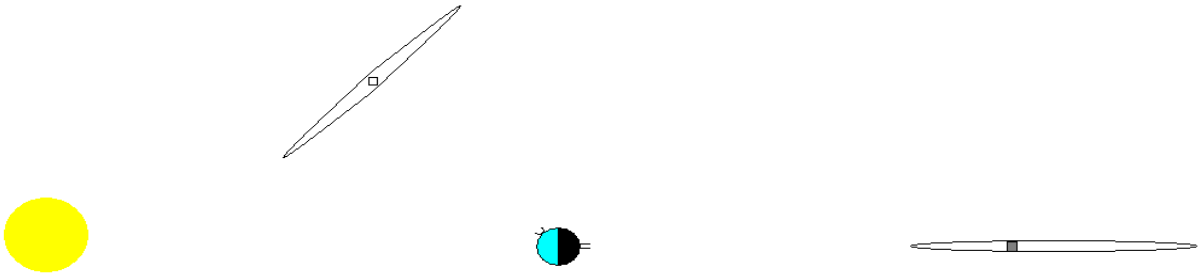

Fig. 1. Radar and optical covariance examples

Given past sensor measurements and the propagation forward of those covariances, each object, at any specific time, will have its own unique error volume shape and orientation. When contemplating which specific sensor and sensing time to use for the next observation, its location, phenomenology, location, and specific sensing geometry (which changes as the object moves through space over time) all have to be considered. This can be done relatively easily for any specific choice of sensor and observation time by calculating the resulting covariance of the observations and intersecting it to the covariance of the object from past observations, propagated to the potential observation's time. As illustrated in Fig. 2 and Fig.3 complementary observations are ones that result in error volumes that are more perpendicular (Fig. 3) than parallel (Fig. 2).

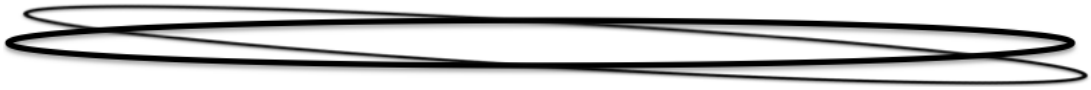

Fig. 2. Combining covariances at a very acute angle.

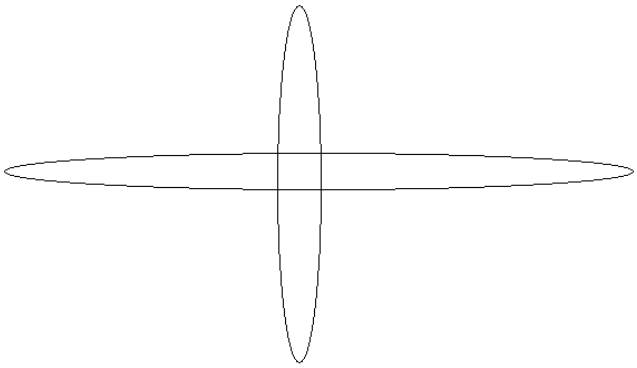

Fig. 3. Combining covariances from orthogonal measurements

3. PAST COMPLEMENTARY OBERVATION EXPERIMENT AND RESULTS

Previously, as described in [1], we looked specifically at the effect of utilizing complementary observations on the resulting covariances, focusing on deep-space objects and compared it against the practices that was current at that time. That space object tracking sensor-scheduling process took as input the current Space Catalog. Based on the characteristics of the object, the specifics of each sensor, and the geometry of each object's orbit with respect to each sensor site, a maximum probability of detection was calculated. Based on each object's priority, number of observations needed, and the maximum detection probabilities, each sensor is tasked to observe each object a specific number of times. A scheduler at each sensor site determines which opportunities and when within each opportunity to make observations (i.e. it schedules), without reference to what other sensors are observing the same object (and when) and which orbit metrics will contain the most and least errors. It does not necessarily schedule the same time or even the same opportunity that corresponds with the maximum probability of detection.

A new scheduling algorithm was developed that took as input the space catalog and the associated covariance matrices and produces a globally optimized schedule for each sensor site as to what objects to observe *and when*. This algorithm was able to schedule more complementary observations, in terms of the precision with which each orbit metric is known, to produce a satellite observation schedule that, when executed, minimized the covariances across the entire space object catalog. The results were increased accuracy of the space catalog with the same set of sensor resources. The algorithm was prototyped and tested in simulation and the resulting orbit metric covariances calculated.

3160 deep space and geosynchronous satellites were considered for the purpose of this study. Nine sensors were scheduled for Deep Space and Geosynchronous object tasking: 5 optical sensors and 4 radars. Sensor data was obtained from [2]. In order to model the Earth orbits of every satellite in our catalog, we used NORAD's SGP4 propagation code (available from www.celestrak.com). Ideally, a sensor track at a given time would consist of several individual observations, which would then undergo differential correction to correct the TLE to reflect the new information provided by the track. For our purposes, since we were not actually updating TLEs, we just needed to calculate the covariance matrix for each measurement, which we did using particle techniques. An observation at time, t, consists of a range, azimuth, and elevation value for radar sensors, or right ascension and declination for optical sensors. The assumed true values of the observation, as calculated by the look angle calculations and SGP4, are perturbed by Gaussian error added in accordance with the published sensor errors from [2]. Once these error-added observations are calculated, the Herrick-Gibbs method of orbit determination is used to produce a set of position and velocity vectors at time, t. The covariance of the sample is found through the standard covariance calculation for each element of position and velocity, producing a 6x6 matrix from our set of 6-D vectors.

3160 deep space objects from the catalog were tasked and scheduled by the prototype scheduler and compared to a simulation of the current method of scheduling for a 10-day period and the resulting covariances calculated in the position-velocity space at the end of the 10-day period. From these 6 x 6 matrices, the 6 variances were extracted and compared as described by Table 1 below.

Table 1. 3160 Deep Space Objects Final 10 Day Position/Velocity Variances.

	Prototype Average	Current Method Average	Current/ Prototype	Prototype Std Deviation	Current Std Dev.	Current/ Prototype
X $(km)^2$	3.36	14.43	4.3	3.26	45.27	13.9
Y	3.66	14.39	3.9	8.04	45.32	5.6
Z	3.11	8.08	2.59	4.46	21.32	4.8
Vx $(km/s)^2$	0.00148	0.00577	3.88	0.00130	0.00662	5.1
Vy	0.00151	0.00577	3.82	0.00131	0.00662	5.1
Vz	0.00141	0.00566	4.01	0.00130	0.00662	5.1

The first three rows refer to the x, y, and z components of the position variances in kilometer- squared units and the last three rows refer to the x, y, and z components of the velocity variances in kilometer per second squared units. The first row is the average of the variances across all 3160 objects for the 10-day schedule of observations output by the prototype scheduler and the second column is the corresponding average for the schedule produced by the current scheduling method. The third column is the ratio of the second column divided by the first and is a measure of how much better the prototype-produced variances were. The last three columns are similar to the first three but refer to the standard deviations of the variances instead of the average.

Clearly, having the scheduler purposefully schedule complementary measurements does significantly reduce the resulting uncertainty of the objects' locations after a ten-day period, producing variances 3 to 4 times better. The ratio of the standard deviations of the variances is even greater, indicating that the current method variances have a far greater spread, which is very undesirable. These high standard deviations for the current method indicate that there were some very bad measurement errors for some objects. One hypothesis to explain this is that the geosynchronous objects, although a very important part of the deep space regime, might have significantly higher measurement variances than the average because current method would tend to sense these objects in the same parts of their orbits each time. To investigate this hypothesis, we ran the same statistics on the 674 GEO objects, resulting in Table 2 below.

Table 2. 674 GEO Final 10 Day Position/Velocity Variances.

	Prototype Average	Current Method Average	Current/ Prototype	Prototype Std Deviation	Current Std Dev.	Current/ Prototype
X $(km)^2$	4.95	29.73	6.0	2.17	72.60	33.5
Y	5.22	27.82	5.3	4.31	67.74	15.7
Z	4.13	9.17	2.22	5.78	16.11	2.8
Vx $(km/s)^2$	0.00191	0.00807	4.23	0.00228	0.00732	3.2
Vy	0.00193	0.00806	4.17	0.00229	0.00735	3.2
Vz	0.00168	0.00783	4.66	0.00231	0.00734	3.2

As hypothesized, the GEO objects had significantly higher average position variances for the current scheduler algorithms, especially in the X and Y dimensions. (The Z dimension is oriented "up" and the GEO objects tend to be near the equator, so large range optical errors from earthbound sensors tend to have a relatively small error in the Z-direction.) Some growth is expected, since angular measurements increase linearly with distance and GEO objects are significantly farther away than the average DS object. This can be seen by the slightly higher mean variances for the prototype. However, the X and Y components of the position variances for the current method double so that the ratio with the prototype is in the 5 to 6 range (or 4 to 5 if the Z component is included.) Even more significantly, the standard deviation of the current method variances is very high, indicating that there are some very large measurement errors, even after 10 days of observations. The prototype's standard deviations are 17 times better for the GEO objects. The GEO belt is an especially important deep space region and covariance-oriented scheduling greatly improves the accuracy of objects' positions and significantly reduces the variability of that accuracy (i.e. greatly reduces the worst-case situations).

These deep space results will not carry over to near-earth objects, where errors are reduced considerably due to their closer proximity and short orbit periods, which tend to give more variability in the position of the orbit where measurements are taken and the different angles between the sensor and object tend to give complementary angles.

4. COMMERCIAL SSA SENSOR CAPABILTIES

Currently there are hundreds of commercially owned SSA sensors scattered across more than a hundred sites, with more being added every month. At the present time, the government does not allow direct use of commercial SSA sensor data in its orbital parameter calculations because of certification issues; each SSN sensor must be certified; for now, the certification process for each sensor is fairly onerous, and the large number of commercial SSA sensors makes this impractical. (There is some movement toward certifying commercial SSA sensor networks as a whole). In spite of these issues, there are several functions that government agencies could use commercial SSA data for now. These include searching for newly lost objects and providing orbital parameters to allow SSN government sensors to re-acquire; searching volumes of space for new objects, and other tipping and cueing; on the fly (short lead-time items) tasking (which could be volume/time based to avoid classification issues); high priority objects (which could be volume/time based, to avoid classification issues); maneuver and propulsion detection; post-launch observations; occasional tracking of classified objects (a small percentage of such requests could not be exploited by adversaries); space object identification (SOI) information such as images, light curves (to derive rotational and other movement frequencies and normal sun orientations), and passive RF signals and their timing; and track maintenance for low priority, unclassified objects such as debris and commercial and university satellites.

While the vast majority of commercial optical SSA sensors are used for GEO object observations, these also have significant capabilities and capacity for LEO objects as well, as shown by our LEO experiments described in Section 8, Experiment Results. Furthermore, there are several commercial SSA sensors specifically designed for LEO objects.

5. COMMERCIAL SENSOR ADVANTAGES

The sheer number and diversity of the hundreds of commercial SSA sensors spread across more than 100 sites ensures a capacity that can provide persistent GEO and LEO coverage, as shown in the Experiment Results in Section 8. Commercial SSA sensors are immediately responsive, able to be re-tasked to a new object in as little as

tens of seconds up to a few minutes. Furthermore, they can provide real-time data, even being able to see what's happening in GEO in real-time a few minutes after tasking. For a satellite operator who has lost their satellite or is experiencing significant anomalies, this can be a real life saver. Commercially owned SSA sensors often have a very low cost per observation (cents!), low cost per angular field of view covered, and are very cost-effective. They often have a subscription business model which provides for continuous improvement in accuracy and information extraction capabilities. Because they were not building to a set of requirements, they did not stop making improvements once these requirements were met and so are continuing to improve over time. After several years of these improvements, they are extracting an extraordinary amount of information (angles/brightness/dim objects) from their sensor data. They currently observe behaviors (including light curves) and can tell if a satellite is 3-D stabilized, spin-stabilized, or tumbling, can observe burns and burn size, see slot changes, view catastrophes, and watch objects being deployed from satellites (or being broken off). They currently help operators locate satellites in response to immediate requests and observe anomalous satellites in response to immediate requests. Satellite operators and other organizations requiring space situational awareness need the space object observation data; they should not want to acquire, own, and maintain SSA sensors. Commercial SSA sensors fill this niche.

6. SPACE APPLICATION SCHEDULING

It is frighteningly easy to build a bad scheduler, and surprisingly hard to build good one. There are two reasons for this. Because the input/output specification is relatively simple (the input is just the tasks and resources that need to be scheduled along with relevant constraints, and the output is just the assignment of resources to tasks for specific time periods), a simple algorithm can fulfill the input/output specification, i.e. assign resources to tasks. However, this simplicity hides an important truth, scheduling problems with meaningful resource assignment are NP-Complete meaning that they require exponential time to guarantee an optimal solution. For example, consider a scheduling problem with 1000 tasks to be scheduled where each task as on average 4 meaningfully distinct choices for resource or time frame. Then the number of different possible combinations is $4 \times 4 \times \ldots 4 = 4^{1000} = 2^{2000} = 10^{200}$ or a '1' with 200 '0's after it. For comparison, this is significantly more than the number of known particles in the known universe (about 10^{80}). Scheduling problems are not just exponential, they are very exponential!

So it is effectively impossible for a scheduling algorithm to guarantee an optimal solution for any scheduling problem of significant size. Every scheduling algorithm is different and produces different answers, some good, some bad, some fast, some slow, with slow algorithms not necessarily producing better schedules than fast ones. Many scheduling algorithms are based on search. These include Genetic Algorithms, Simulated Annealing, A*, Heuristic Search, Iterative Repair, etc. Of course with such a huge number of possible solutions, only an extremely tiny fraction of the possible solutions can be looked at. The field of Operations Research provides techniques such as Convex Optimization, Linear Programming, Branch and Bound, Hill-Climbing, Mixed Integer Programming, etc. To be applicable and tractable, these methods usually must oversimplify the problem. A common very bad algorithm that is very often used is to process the tasks in priority order then greedily pick the resource and time window that is best for that task.

To see why this is a bad algorithm, consider the following very simple example. Assume you have three objects or tracks that need a single observation each in some specific time window, called Track 1, Track 2, and Track 3, where Track 1 has the highest priority and Track 3, the lowest. Assume you have 2 sensors: A (which can sense Track 1 or 3) and B (which can sense Tracks 1 and 2). Also assume that A is slightly better than B for Track 1. A priority-based, greedy allocation is poor because A will be assigned to Track 1, then B to Track 2, and Track 3 gets nothing because A is already allocated to Track 1 (notice the "or"). This is poor because a different algorithm could have allocated sensors to all tracks (B is allocated to Tracks 1 and 2 and A is allocated to Track 3). Another reason priority-based ordering algorithms are bad is because there are other more appropriate ways to ensure that all high-priority tasks get accomplished. For example, as a last step before scheduling is finished, if any high-priority tasks did not get scheduled, lower priority tasks could be swapped out to make room at the end.

What the above example makes clear is that scheduling a flexible, high priority activity on a specific resource at a specific time, when other options are available is a bad idea because the specific resource and time may be when a lower priority activity absolutely needs it. I.e. it tends to be a poor decision to schedule high-priority "easy" tasks before scheduling low-priority "hard" tasks. Much better is to schedule the most-difficult-to-schedule tasks first. Tasks are difficult to schedule if they have highly constrained time windows and if they need resources that are highly contended for by other tasks, especially if they need them during times with the most contention. It is

important, during scheduling, to consider the resource side of the process, not just the task side and to consider the global perspective (schedule difficulty, resource contention) when making local decisions (such as which task to consider next and which resource and at what time to assign it to a task). One way to consider global information throughout the scheduling process is to do calculations during a pre-processing step which pre-calculates which resources are most contended for and at which times and which tasks are going to be the most difficult to schedule. The most-difficult-to-schedule tasks can be considered earliest in the scheduling process. And when assigning resources, try to avoid contended-for resources, especially at the most contended-for times. (Highly contended-for resources at highly contended-for times are often called "bottlenecks") At the end, if necessary, if some high-priority tasks could not be assigned resources, swap out lower-priority tasks to make room.

The above description discusses two important decisions in the scheduling process. One is fairly obviously important, the assignment of specific resources to specific tasks at specific times. The other is just as important but much less obviously so – the order that tasks are processed for scheduling will have a major impact on the optimality of the resulting schedule and runtime efficiency of the scheduling algorithm. This is why priority-ordered algorithms are so bad, they are using essentially random information (from a scheduling difficulty and optimality perspective) for making a very important decision (scheduling order). Reinforcing this point, one expert scheduler once said, "Priorities don't matter". By this she meant that they should not be considered first but only as a last resort. Use of priorities amounts to a kind of surrender. When you use them, you have essentially given up trying to schedule all tasks. This should be a last resort not a first try.

Meanwhile, human expert schedulers often can make use of these global perspectives when performing scheduling, at least on scheduling problems of reasonable size. They can follow a linear or near-linear process to generate near-optimal schedules in a reasonable amount of time. They obviously are not using a search-oriented strategy, OR optimization, or an exponential algorithm. Clearly this implies using near-linear scheduling algorithms that rely on pre-calculated global information.

7. BOTTLENECK AVOIDANCE ALGORITHM APPLIED TO SPACE APPLICATIONS

The motivation behind the Bottleneck Avoidance (BNA) is based on our experience with human expert schedulers who, as mentioned above, are often very successful at building highly optimal schedules. Scheduling is a skill that is very specialized and requires lots of training and experience. Building a schedule manually requires a great deal of time and effort from highly skilled personnel. The opportunity is to apply automated techniques that mimic experts' processes and leverage existing knowledge. We have observed hundreds of expert schedulers over the past three decades in many dozens of different domains. And often, their process is surprisingly similar. First, they look at some representation of the entire schedule in graphical form and see where the areas of highest contention are and typically work on "detangling" these areas first. This is the process that we have algorithmically implemented and described below.

As described in [3], the goal is to schedule the least flexible tasks first; leave room for more flexible tasks to schedule later. The first step is to preprocess all tasks/resources for global information then focus on the most constrained resources at the most constrained times and tasks that most need them; then try to schedule to avoid these bottleneck resources/times. The preprocess step involves partially "allocating" each task's resource requirements across all possibilities then summing all "allocations" for each resource for all times. During scheduling, focus on the peaks of the allocation totals since these represent times when many tasks are requesting the same resources at the same times, i.e. the bottlenecks. At these peaks, first process the biggest contributor to the peaks since this represents the task with the least flexibility. When choosing resources and times, make choices that most reduce these peaks (i.e. choose resources and times away from peaks, specifically preferring "valleys" when such options are valid). After each specific assignment, the profiles have to be adjusted since before the assignment, most tasks could theoretically have been assigned several different resources at several different times (where each has a relatively small probability) but after the assignment, the task has a specific resource at a specific time with probability 1.0 and all other possibilities now have a probability of 0.0. This algorithm has the beneficial property that it automatically adjusts to different scenarios (different bottleneck resources and times), different resource types and numbers, different sequences, etc. The algorithm is described in more detail, with graphical examples, below.

The first step, "probabilistically" allocating the resource requirement for each task across all of its possibilities involves essentially determining how many different possibilities there are. In the case where there is only one

possibility for a resource and time, as shown in Fig. 4, there is a 100% "probability" the task will end up at that resource and time. If there was one specific time, but two different resource possibilities, then each rectangle would have a height of 0.5. Fig. 5, shows the case of when there is exactly 1 resource that can fulfill the task, but a continuum of possibilities in time. In this specific example, the task is 20 minutes in duration and it must be scheduled within a 29-minute window. This means that the start time must be sometime in the first 9 minutes, as shown. This also means that there is a 100% probability that the task will be executing in the middle portion of the 29-minute window. This leads to the characteristic trapezoid probabilistic assignment shape.

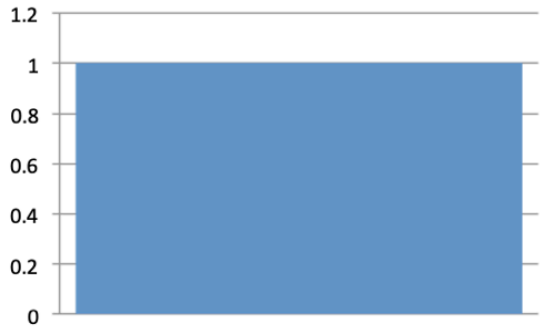

Fig. 4. "Probabilistic" allocation when only 1 choice

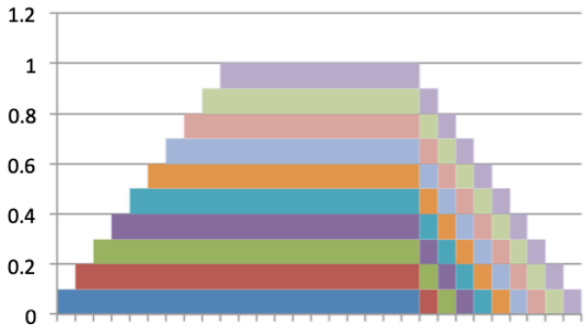

Fig. 5. "Probabilistic" allocation when many choices in time

Very flexible tasks will never reach a 100% probability as shown in Fig. 6. In this example, where a task with a duration of 10 minutes must be scheduled within a time window of 30 minutes, this means that the start time has a range of 20 minutes (or 5% per unit time) which creates a trapezoid with a maximum height of 10 x 5% = 50%. Generally the process is to calculate a trapezoid for each task and divide it across each resource. E.g. with 4 possible resources each resource would get 25% of the trapezoid. In space applications, where different resources may represent different visibilities, this has to be adjusted somewhat, as not all of the trapezoids will be the same for all resources for even the same task, when the visibilities have different durations.

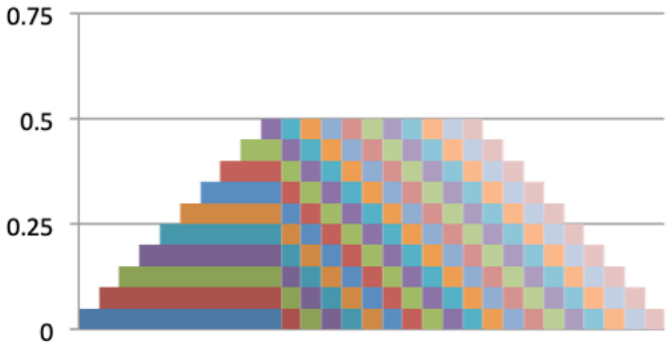

Fig. 6. Task allocation when very flexible in time

After creating the rectangles and trapezoids that represent the "probabilistic" assignments for all tasks, these need to be summed for each resource across all tasks. A simple example is shown in Fig. 7 with two resources at one site and three tasks. The blue one is a LEO object with a very specific time window and the requirement to use only Side 1. The green and red ones have more flexibility in that both can use either side and both have a range of possible times. In the examples below, pre-assignment probability contributions are shown in a lighter color and post-assignment location of the scheduled tasks is shown in a darker color.

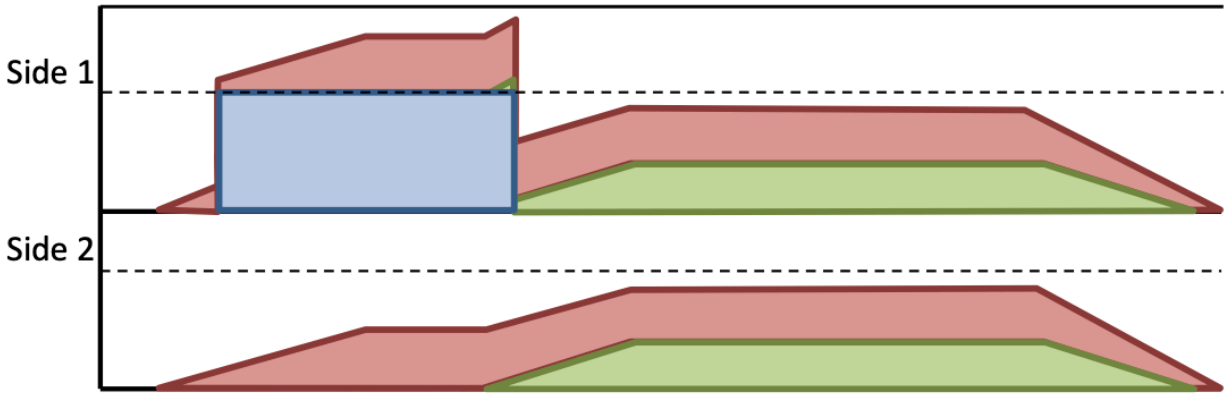

Fig. 7. Summing "Probabilistic" allocations for all tasks across all resources

Schedule Processing Order

BNA is a one-pass algorithm without backtracking. The order in which tasks are processed is *very* important. Avoid waiting to schedule an inflexible task because its required resources may get allocated to a different earlier-processed task. Each step attempts to reduce bottlenecks (peaks of predicted resource contention). To find the next task to schedule:

1. Find the tallest predicted usage peak or bottleneck that has at least one unscheduled task
2. Find the unscheduled task that contributes the most to the peak (the task that is most likely to schedule there)

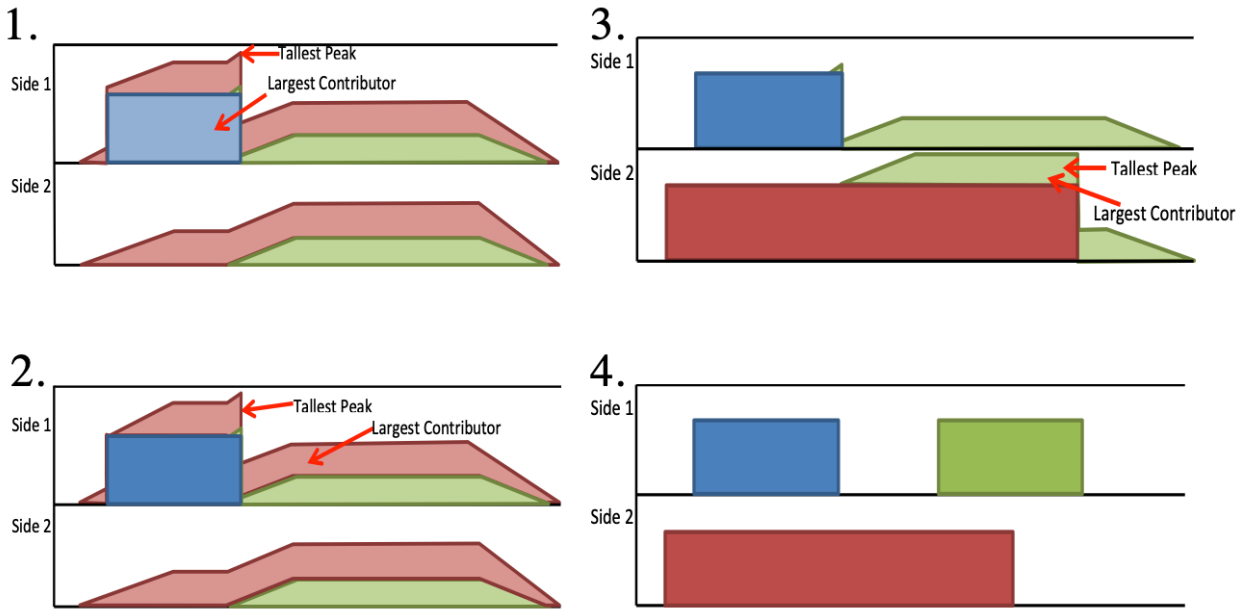

Fig. 8. Processing Order Example

Temporal Allocation and Resource Selection

Once a task is selected for processing, scheduling involves:

1. Finding an open combination of resources that satisfies the task's requirements
2. Allocating the task temporally on those resources

Bottleneck Avoidance attempts to minimize resource contention at each step by scheduling a task *away* from bottlenecks, such that its new allocation minimizes all peaks.

Temporal Allocation and Resource Selection (Procedure)

Remove the task from the bottleneck model (remove all of its "probabilistic" allocations)

Within temporal bounds, find *all* open time spans:

1. Of at least the task's duration
2. Where all resource requirements can be met
   - Keep track of *which* resources are available during each span

For each possible window inside a span:

1. Calculate the maximum bottleneck peak
2. Calculate total area under the curve

Select the possible window with the lowest peak, using area as a tiebreaker
Add the task back to the bottleneck model as an *actual* rather than "*probabilistic"* allocation

8. COMMERCIAL SSA SENSOR SCHEDULING EXPERIMENTS/RESULTS

We implemented the Bottleneck Avoidance algorithm and confirmed that it appeared to produce optimum results. (I.e. humans looking at the resulting schedule could not see any ways to make improvements). As previously mentioned, there are a large number of commercial SSA sensors. These are mostly used for quick reaction, anomaly resolution, and SOI information such as behavioral features, light curves, etc.. But an interesting question is what the theoretical capacity of these commercial sensors really is, compared to the existing set of SSN sensors. This question can only be answered with a high-quality scheduler. (Because a poor scheduler would schedule fewer observations with the same set of sensors than a good scheduler.) Commercial SSA sensor operators say that not only can they see cubesat-sized objects in LEO, they can also see them at GEO. Detection models are outside the scope of our work. So as a first approximation, we took the entire space catalog with over 17,000 space objects, of which about 4,000 are in LEO. Using 528 sensors employed at 93 sites, we calculated all of the visibilities where, for optical sensors, the objects were in sunlight and the optical sensors were in astronomical darkness (sun center was 18 degrees below the horizon). This produced about 1.5M visibilities in a 24-hour period. We were able to successfully schedule, in less than 2.5 minutes on a standard desktop single thread for the Java version, a single observation on every object when the durations were 4 minutes for LEO and 8 minutes for non-LEO objects and the vast majority of objects when the durations were 8 minutes for LEO and 16 minutes for non-LEO, which indicates that, if so-directed, the commercial sensors have a fair amount of LEO and GEO persistence and the capacity to maintain the entire space catalog.

Incidentally, we were able to drop the 2.5 minutes to 1 minute by parallelizing the algorithm and using the multiple cores on the desktop CPU. We also re-implemented the single thread algorithm in C++ which then had a runtime of a little less than 1 minute. We then optimized the single-thread C++ code, dropping the runtime down to 8 seconds. Finally, we parallelized that version of the C++ code for a final runtime of only 4 seconds, which is the version we primarily use.

We also examined a scenario with multiple observations per 24 hour day. All 17,000 objects were given three observation tasks in separate 8 hour windows with the additional constraint that for the same target, the observations had to be separated by at least 4 hours. Observation times were chosen randomly between 2 and 4 minutes for LEO targets and between 4 and 6 minutes for non-LEO targets. The optimized, parallelized C++ scheduler was able to successfully schedule 99% of 52,000 tasks in 4 seconds (while considering 1.5M visibilities). The vast majority of the unscheduled tasks relate to the fact that these experiments were run near the Summer solstice when the North American sites had short nights especially during the 8 hour time window corresponding to the North American day. This represents a worst-case, in terms of time of the year. Longer North American nights would only have increased number of observations that could be scheduled.

Of course one of the main advantages of the commercial SSN is quick reaction. Given a set of 100 high-priority tasks, the scheduler was able to find a valid sensor for all of them in 0.01 seconds (0.1 millisecond for each), while bumping 94 tasks to do so. The average time to that next visibility varied considerably with many being only a few minutes away. (This was highly dependent on the sun geometry.) 31 of the bumped tasks could be rescheduled within 0.023 seconds, thus minimizing the disruption caused by the quick reaction tasks.

9. UNIFIED DATA LIBRARY (UDL) INTEGRATION

The Unified Data Library (UDL) is a central repository of SSA data, jointly funded by AFRL/CAMO and SMC/DPMO in order to increase exposure of commercial space data and enable access to academic, government

and commercially-gathered satellite data sets. The UDL supports a variety of data access methods including batch, query, streaming, and archive. Most commercial SSA providers are represented with data on the location and features of each sensor, observation data, calculated orbital parameters, detected maneuvers, etc. Access to commercial observations is dependent on data purchases or affiliation to an effort that has purchased data. For example, a government funded contractor may receive access to all of the government purchased data, depending on the specifics of the contract terms. There is live data streaming in from commercial SSA sensors all the time. The UDL streamlines data distribution and data integration for end users or applications and is setup to easily allow separate deals to be made between private consumers of SSA data and commercial providers of that data. It is possible, in order to create an integration mechanism among consumers of SSA data, optimized scheduling algorithms, and SSA data providers, to add real-time tasking data to the UDL that could be responded to by commercial owners of SSA sensors (i.e. commercial providers of SSA data) such as specifically assigned tasks in real-time for real-time monitoring/execution by commercial SSA sensor owners. Of course these assignments this will only be as optimal (in terms of getting the most tasking accomplished with the set of sensors available) as the scheduling algorithm used to make the assignments. The UDL also supports a variety of classification levels from Unclassified to TS/SCI. Additionally, in the fall of 2020, SMC's SSA marketplace will come online on the UDL to enable real-time transactions and distribution of SSA data.

10. FUTURE WORK

We are in the process of extending the initial algorithm in a variety of ways. These include more diverse tasking (e.g. searching, SOI, sensor quality), more testing with more diverse and more realistic scenarios, improved deliberate and quick-reaction algorithms, integrating current and forecast weather, including detection probabilities and sensing quality metrics, and integrating with the UDL, both to receive inputs (sensor parameters, data request information (previous successful observations, current orbital parameters and covariances, etc.), etc.) and to export the required task assignments in real-time for the commercial SSA data providers to access and execute. We will also be adding a government agency querying capability. Additionally, we have plans to adapt the schedule for other satellite scheduling domains such as satellite-based sensor collection scheduling and satellite communications scheduling.

11. CONCLUSIONS

Commercial SSA sensors are quite numerous and capable and are constantly improving and becoming more numerous. They offer near-real-time tasking and data/visualization while being extremely cost effective, with observations sometimes costing just pennies each. Commercial SSA sensor data are readily available through the UDL and directly from the individual companies. There are a number of different use cases for government and private industry consumers of SSA data. We have shown in this paper that high-quality space scheduling algorithms exist to take full advantage of SSA sensors.

12. ACKNOWLEDGMENTS

This material is based upon work supported by the United States Air Force under Contract No. FA875119CA057. We also thank Rahul Malappan, Zachary Fine, and Alex Wu, all from Stottler Henke Associates, Inc., for their contributions in the development of the SSN Sensor Scheduler.

13. DISCLAIMERS

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

14. REFERENCES

[1] R Stottler. Improved Space Surveillance Network (SSN) Scheduling using Artificial Intelligence Techniques, *Ica AMOS 2015 Proceedings*, 2015.
[2] *Fundamentals of Astrodynamics and Applications* (Vallado, 2001).

[3] R Stottler, K. Mahan, and R. Jensen. Bottleneck Avoidance Techniques for Automated Satellite Communication Scheduling. Proceedings of the Infotech@Aerospace 2011 Conference, American Institute of Aeronautics and Astronautics. St. Louis, MO.