A Regional Greedy Algorithm for Space Domain Awareness Resource Allocation

Naomi Owens Fahrner
Ball Aerospace

ABSTRACT

This paper presents a new objective function for the problem of Space Domain Awareness resource allocation (SDARA) as well as a novel algorithm to maximize this new objective function. This SDARA problem aims to maximize the total number of targets seen while minimizing resource costs. For this purpose, targets primarily consist of objects in the GEO-belt, while observers consist of GEO, LEO, and ground-based optical sensors. These sensors are assumed to be heterogeneous and have different associated tasking costs.

The novel algorithm, called the "block greedy" algorithm, provides an approximate regional maximum of this objective function in a tractable amount of time. The block greedy algorithm is a hybrid of the weapon-target-assignment and greedy algorithms. This algorithm will be shown to outperform common algorithms used in solving the SDARA problem.

1. INTRODUCTION

Space domain awareness resource allocation (SDARA) has gained much interest over the last few years. Given the rapid growth of resident space objects (RSO), this is no surprise. With over 34,000 RSO greater than 10 cm [10] and only a small fraction of that number of observers, allocating observer resources is imperative.

The resources in this SDARA problem are the space domain awareness (SDA) sensors observing targets. The goal of SDA sensor tasking is to maximize the total number of targets observed while also minimizing resource costs and target revisit time. This is done by constructing an objective function and optimizing this against various constraints.

Current objective function formulations of this SDA problem do not address the different costs associated with tasking heterogeneous sensors [8, 3]. Given the variation in possible observers, it is crucial to also consider the tasking cost of each observer. Depending on the observers, it may be more cost effective to task certain sensors more than others given their cost of use.

This paper presents a new objective function for the SDA sensor tasking problem that considers GEO, LEO, and ground-based optical sensors all at once. For this purpose, targets primarily consist of objects in the GEO-belt. This proposed objective function considers the unique cost of moving the look angle of each observer as well as the cost of storing images in each observer.

Additionally, a novel algorithm to maximize this new objective function is discussed. This will be compared with other common algorithms used to optimize common SDA sensor tasking objective functions. It will be shown that this new algorithm outperforms the other algorithms in maximizing this objective function.

There are several existing objective functions for the SDA sensor tasking problem. Each of these have slight variations from each other. This presented objective function is most like the one presented by Frueh [5]. This presented objective function can be thought of as an extension of Frueh's objective function to include additional observer costs and constraints.

In Little's thesis, multiple algorithms are compared in solving Frueh's objective function formulation [9]. These algorithms include a greedy algorithm, weapon-target-assignment (WTA) algorithm, ant colony optimization algorithm, and a distributed q-learning algorithm. This paper is primarily interested in optimization algorithms as opposed to machine learning algorithms. Thus, of these algorithms, only the greedy and WTA algorithms are considered.

Another common algorithm used to solve the SDA sensor tasking problem is a genetic algorithm [4, 6]. Thus, a greedy, WTA, genetic, and random search algorithm will be compared with the new algorithm to show it outperforms the common algorithms used in solving this SDA sensor tasking problem.

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

2. SCHEDULE GENERATION

Resources are allocated by creating an optimal schedule for observers. A schedule is comprised of tasks for each observer at each time window over some time interval. This time interval is discretized into time windows. A task consists of slew and/or collect commands. The collect command schedules specific targets to be imaged during the given time window.

The general term for the system that schedules tasks will be called a "scheduler" for the remainder of this paper. To create a schedule, the scheduler assigns tasks to each observer for each time window. Algorithms will be implemented in the scheduler in order to create approximate optimal schedules for observers.

To generate a schedule, a search area is first discretized into voxels. For simplicity, a target is assumed to be the center of each voxel. Observers are then tasked to slew and/or collect images of targets. This process continues over all time windows until a schedule is generated.

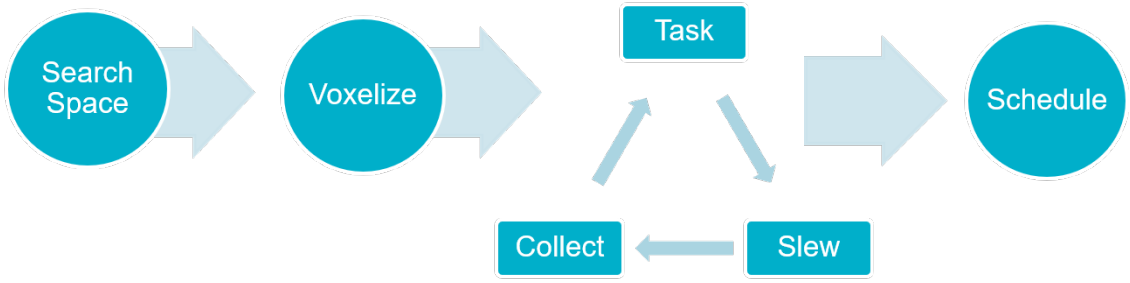


Fig. 1: Schedule Generation

In order to create an optimal schedule, each task is assigned a score. Factors in this score include a target's priority of visitation, a target's probability of detection, and observer costs. Additionally, there are constraints that must be considered for a target to be tasked. These include the following: a target must be within an observer's field of view, a target must lie outside of the Earth's umbra, the Earth must not block a target from an observer, and a target's SNR must be greater than or equal to an observer's SNR threshold. Additional constraints to task an observer include the following: an observer's memory must not be full, it must be able to slew to the given position within the given time window, and the task score must be above the observer's score threshold.

2.1 Priority of Visitation

The priority of visiting a target n, denoted μ_n, is a function of time and can be determined in a variety of ways. For the results to follow, the priority of visiting a target is a linear function of time. In the linear example, the more time that has passed since a target was visited, the higher priority of visitation that target has. Once a target is visited, its priority is reset. Resetting a target's priority of visitation means lowering the priority, unless otherwise specified.

In addition to a linear priority of visitation, the scheduler can set a constant priority of visitation or a user defined priority of visitation. The priority of visitation is a function of the current time window, $w(t)$, and the last time the target was visited, t_n.

2.2 Probability of Detection

Each target has a probability of being detected by each observer. This is modeled by the cumulative distribution function (cdf) of a normal distribution. Each observer has an SNR threshold in order to detect a target. If a target's SNR value is lower than an observer's SNR threshold, it will be undetectable by the observer. Thus, the probability of detecting a target is a function of the target's SNR and the observer's SNR threshold. This is defined below:

$$p_{ns}(r_{ns}, h_s) = 0.5\left(1.0 + erf\left(\frac{r_{ns} - h_s}{\sqrt{2}}\right)\right), \tag{1}$$

where p_{ns} is the probability of observer s detecting target n, r_{ns} is the SNR of target n received by observer s, and h_s is the SNR threshold for observer s.

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

The SNR of a target is a function of time, observer position, and target position. Observer position is computed using an sgp4 propagator [1]. The sgp4 propagator takes inputs of time and observer two-line element sets (TLE). Thus, p_{ns} is dependent on observer SNR threshold, time, observer TLE, and target position.

2.3 Observer Costs

Each observer has a cost associated with tasking it to various objectives. It has a cost of storing images, i_s, and a cost of slewing to a specified position, p_s. As mentioned, these costs vary per observer. This is done to create more realistic tasking scenarios, as real-life observers have different tasking costs. Depending on observer costs, it may be more beneficial to task some observers more often than others.

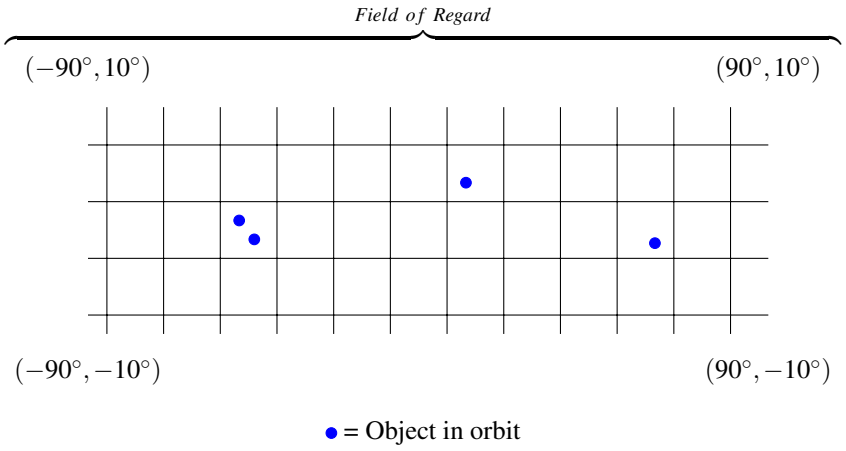
Each observer has a cost of storing an image. The cost of storing a specific image is found by determining how much storage space remains. It should be noted that observer costs are negative and added to the objective function to eliminate ambiguity.

The cost of slewing is called the pointing score. This is a function of the observer's current looking position, the potential looking position, and the cost of minimal gimbal movement. Again, this cost is negative and added to the objective function.

2.4 Constraints

The scheduler has real-life constraints such as in order to schedule a target, the target must be in an observer's field of view, not blocked by the Earth, outside the Earth's umbra, have a high enough SNR, an observer must have available storage space, an observer must be able to slew in the time window, and the task score must be above the score threshold.

Each observer has an associated field of view (fov) and a field of regard. An observer's fov is the volume an observer can see at a fixed orientation. Its field of regard is the total volume an observer is capable of viewing assuming its movement. Below is a 2-dimensional example of this:



In the graph above, each grid represents an observer's possible fov; the spheres represent targets of interest.

Each observer has an SNR threshold, h_s, at which it can see a target. If the received SNR from a target to an observer, r_{ns}, is lower than the observer's threshold, the target will not be seen by that observer at that time. Additionally, if a target lies within the Earth's umbra, $[\vec{u}_{start}, \vec{u}_{end}]$, it will not be seen. Similarly, if the Earth lies between a target and an observer, the observer will not see that target at that time.

The gimbal of each observer has a unique moving speed, and thus a cost to move it. The slew time constraint only allows an observer to slew to a given position if it can move there within the time window. Additionally, each observer has a unique image storing capacity. If an observer's storage is full, it cannot schedule tasks to be collected.

Each observer has a score threshold, d_s, such that a task will not be scheduled if its score is below the score threshold. This is formulated by the following:

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

$$\left[\sum_{n=0}^{N_{ts}-1} p_{ns}\mu_n\right] + p_s + i_s \geq d_s, \tag{2}$$

where N_{ts} is the number of available targets for observer s at time t. Additionally, p_{ns} is the probability that observer s will detect target n, μ_n is the priority of visiting target n, p_s is the cost to point observer s, i_s is the cost of storing an image in observer s, and d_s is the score threshold for observer s. Equation (2) is calculated for each observer s and each time window $w(t)$. A time window $w(t)$ is a time with an index t that determines which time window is being observed.

Ideally, revisit times are minimized as well as observer costs. As the score threshold increases, so does the max revisit time. Additionally, as the score threshold increases, the observer utilization and max score decrease. The max score is calculated as the sum of the scores of all scheduled tasks. With a higher score threshold, fewer tasks are scheduled and hence the total max score decreases. These results are seen in the figure below:

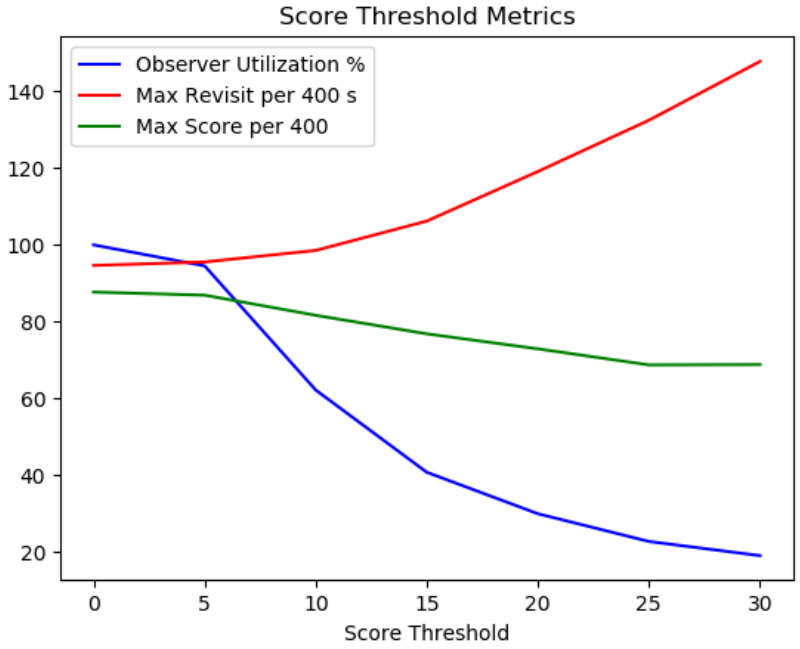


Fig. 2: Score Threshold

3. MATHEMATICAL FORMULATION

First, we will define some definitions. Below is a table of all notations used in this paper:

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

	Definitions
a_s	number of images currently stored for observer s
b_s	buffer size for observer s
d_s	score threshold for observer s
e_{pos}	earth position
F	set of all feasible schedules
fov_s	field of view of observer s
g_s	cost of minimum gimbal movement for observer s
h_s	threshold SNR for a target to be seen by observer s
i_s	cost of storing images for observer s
l_s	slew time
M	number of all possible schedule combinations
M_{st}	number of possible looking directions for observer s at time $w(t)$
N_{ts}	number of available targets for observer s at time $w(t)$
p_s	pointing cost for observer s
p_{ns}	probability of observer s detecting target n
r_{ns}	SNR of target n received by observer s
S	total number of observers
T	total number of time windows
t_n	time target n was last visited
t_{size}	time window size
tle_s	two-line element set for observer s
$\vec{u}_{end}$	end of earth's umbra
$\vec{u}_{start}$	start of earth's umbra
$w(t)$	time window indexed at t
X^*	optimal schedule
X_k	kth schedule
$\vec{x}_n$	position of object n
$\vec{x}_{stk}$	kth schedule looking position of sensor s at time $w(t)$
$\vec{y}_s$	current looking position of observer s
μ_n	priority of visiting target n

The set of feasible solutions is a set consisting of all possible schedule combinations. Each possible schedule combination is one in which all constraints are satisfied, and each observer is tasked for each time window. Note: an observer task does not have to include collecting targets. There may be a case where the negative cost to the observer is more than the positive cost of collecting those targets, resulting in a negative task score.

A schedule is a set of looking directions for each observer for each time. A looking direction for observer s and time t is defined by the vector $\vec{x}_{st}$. This is generalized so that any coordinate system may be used. The looking directions are enumerated for each schedule. The kth schedule, X_k, is defined below:

$$X_k = \{\vec{x}_{stk} : \forall s \in [0, S-1], \forall t \in [0, T-1]\}. \tag{3}$$

The set of all feasible solutions is the set of all possible schedules. This is denoted as F. These are defined mathematically below:

$$F = \{X_0,\ X_1,\ ..,\ X_{M-1}\}, \tag{4}$$

where M is the total number of possible schedules.

An optimal schedule is found by solving the following:

$$X^* = max_{X_k \in F} \sum_{t=0}^{T-1} \sum_{s=0}^{S-1} \left[\sum_{n=0}^{N_{ts}-1} p_{ns}(w(t), h_s, tle_s, \vec{x}_n) \mu_n(w(t)) \right] + p_s(g_s, \vec{x}_{stk}, \vec{y}_s) + i_s(a_s, b_s) \tag{5}$$

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

$$s.t.\ \arccos\left(\frac{\vec{x}_n \cdot \vec{x}_{stk}}{||\vec{x}_n||\,||\vec{x}_{stk}||}\right) \leq fov_s/2 \ \text{(field of view)} \tag{6}$$

$$h_s \leq r_n \ \text{(SNR)} \tag{7}$$

$$\vec{x}_n \notin [\vec{u}_{start}, \vec{u}_{end}] \ \text{(umbra)} \tag{8}$$

$$\vec{e}_{pos} \neq (1-\lambda)\,\vec{s}_{pos} + \lambda \vec{x_n}, \ \ \forall \lambda \in [0,1] \ \text{(Earth blocking)} \tag{9}$$

$$l_s(\vec{x}_{stk}, \vec{y}_s) < t_{size} \ \text{(slew time)} \tag{10}$$

$$a_s < b_s \ \text{(buffer space)} \tag{11}$$

$$\left[\sum_{n=0}^{N-1} p_{ns}\mu_n\right] + p_s + i_s \geq d_s. \ \text{(score threshold)} \tag{12}$$

Let $M_{st} =$ the number of possible looking positions for observer s at time t. The total number of possible schedules, call this M, is calculated by the following:

$$M = \prod_{s=0}^{S-1} \prod_{t=0}^{T-1} M_{st}, \tag{13}$$

since for each observer s there are $\prod_{t=0}^{T-1} M_{st}$ possible schedules. This is multiplied by all observers' possible schedules to get all possible total schedules of all observers.

For demonstration, suppose there are 3 observers, 300 time windows, and 2 possible looking positions for each observer at each time. This results in $(2^{300})^3 \approx 8.452712 \times 10^{270}$ total possible schedules. Even if it only took 0.000001 seconds to generate a single schedule, it would take approximately $2.678503x10^{257}$ years to generate all possible schedules in this simple scenario.

In one of the simplest scenarios represented in the results to follow, there were 3 observers, 1440 time windows, and approximately 3 looking positions for each observer at each time. For this example, M is too large for Python to even compute. Consequently, there is no tractable way to brute force compute the global optimum from equations (5) - (12). Thus, the following algorithms are implemented in order to solve this: greedy, genetic, random search, weapon-target-assignment, and block greedy. In all these algorithms, the constraints remain the same as in equations (6) - (12).

3.1 Random Search

This is the simplest of the algorithms implemented. In the random search algorithm, a random schedule is generated. Its score is then calculated. If this score is greater than or equal to a preset desired score, that schedule is chosen. If the score is lower than the present desired score, a new schedule is generated. This process continues until a schedule is generated with a high enough score. This algorithm is fast but tends to generate the lowest scoring schedule.

3.2 Greedy

The local greedy algorithm schedules the maximum task at each time window for each observer [8, 9, 11]. It solves the following modified version of (5):

$$X^*(s,t) = max_{X_k(s,\,t)\in F}\left[\sum_{n=0}^{N-1} p_{ns}\mu_n\right] + p_s + i_s, \tag{14}$$

for all $s \in [0, S-1], t \in [0, T-1]$, where $X_k(s,t) = \vec{x}_{stk}$.

With this greedy algorithm, X^* is a local optimum. Due to its simplicity, this algorithm runs very quickly. The sacrifice for this quick run time is that this algorithm only guarantees a local optimal solution.

3.3 Weapon-Target-Assignment

The goal of the Weapon-Target-Assignment algorithm (WTA) is to maximize damage by weapons to targets [9]. The dynamic problem determines which targets survived after previous rounds. It then assigns and fires a subset of the remaining weapons with an objective of minimizing the total expected value of surviving targets at end of final stage.

For the SDA tasking problem, the objects to be observed are analogous to the targets and the observers' looking positions available are analogous to the weapons. Only one "weapon" is deployed at a time. Multiple "targets" may be engaged by each "weapon". This algorithm breaks the time interval into groups of time windows. It finds all possible combinations of schedules for each time group and chooses the highest scoring schedule.

This is formulated below:

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

$$X^*(t,c) = max_{X_k(t,c)\in F} \sum_{t=0}^{t+c-1} \sum_{s=0}^{S-1} \left[\sum_{n=0}^{N-1} p_{ns}\mu_n \right] + p_s + i_s, \tag{15}$$

for all $s \in [0, S-1]$, $t \in [0, T-1]$, and $c =$ number of time windows of observation. Here, $X_k(t,\ c) = \{\vec{x}_{st_0k} : \forall\, t_0 \in [t,\ t+c\ -1], \forall s \in [0, S-1]\}$. As c approaches the total number of time windows, X^* approaches the global optimum. If c is less than the total number of time windows, X^* is a regional optimum. Due to its complexity, this is the slowest running algorithm of the five implemented algorithms.

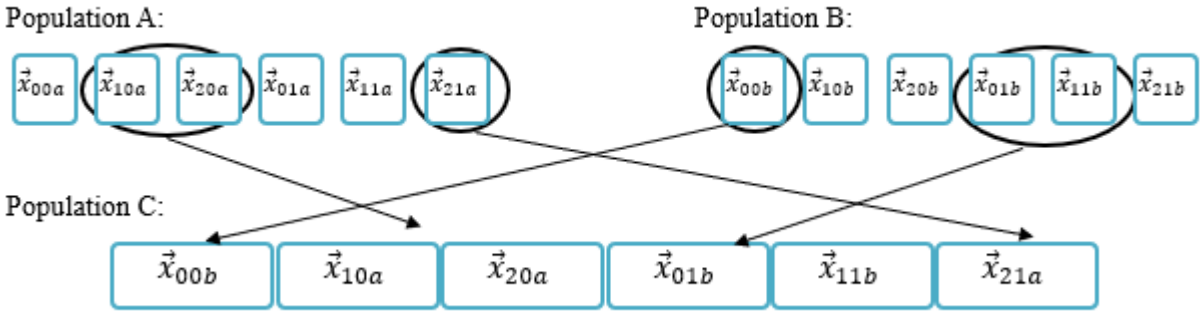
3.4 Genetic

Genetic algorithms are commonly used in high dimensional problems [2]. Because of this, it has been used in the SDA sensor tasking problem [4, 6]. Although genetic algorithms have been shown to be effective for these types of problems, there is not precise way to determine how close to the optimal solution this algorithm gets.

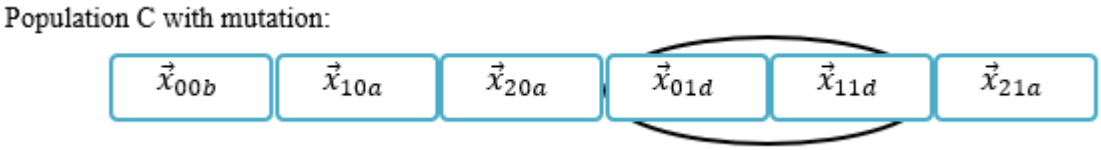
The genetic algorithm uses the following processes: initialize, crossover, and mutate population [7, 2]. A member of a population consists of a possible schedule. In an example with 3 observers and 2 time windows (in the actual simulation there are hundreds of time windows), a schedule consists of the following:

An initial population of size N is randomly created. From the initial population, random member pairs are chosen to crossover. Often these member pairs will be chosen with a weighted factor in order to choose higher scoring members.

The crossover step is completed by randomly selecting pieces of random length of the scheduled pairs. This is done until a new population of size N is created. This process is shown by the following, with random member pairs population A and population B creating new member population C:



After this, the new population schedules are mutated. This is done by choosing a random length segment of the schedule and exchanging it with a random length of another schedule. Mutation is shown by the following: Population C with mutation:



This entire process is repeated for M generations. After these M generations, the highest scoring schedule is chosen as the optimal schedule.

The genetic algorithm is not the slowest or fastest algorithm, nor does it generate the lowest or highest scoring schedules. This algorithm is a common approach in solving the SDA sensor tasking problem. Hence, it is a valuable metric for comparison.

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

3.5 Block Greedy

Finally, the novel block greedy algorithm is a hybrid of the WTA and greedy algorithms, with some additional features. It produces an approximately regional optimal schedule and runs in a tractable amount of time. The block greedy algorithm has not before been proposed or implemented in work on the SDA sensor tasking problem.

Like the WTA algorithm, the block greedy algorithm considers multiple time windows at once. This group of time windows is called a "time block", hence the name "block greedy". Instead of computing the score of every possible schedule combination over this time block as in the WTA, the score for each task at each time window is computed. This is like the local greedy algorithm. However, in the local greedy algorithm, tasks are considered in temporal order. In the block greedy algorithm, tasks are considered without regard to temporal order.

In addition to the mentioned similarities to the greedy and WTA algorithms, the block greedy algorithm has added features. After each task is scheduled, all task scores are reset. This is a key component of the block greedy algorithm. This score reset allows the highest priority tasks to be scheduled first and incrementally schedules lower priority tasks. These lower priority tasks become higher priority tasks as more time passes before they are scheduled. Ultimately, this results in a higher scoring schedule.

To create a schedule using the block greedy algorithm, the score for each task in each time window is calculated for all time windows in each time block. The highest scoring task is scheduled. After this, the priority of visitation is reset for all visited targets. The scores for the tasks across the time block are recalculated and the new highest scoring task is scheduled. This continues until all observers have been tasked for each time window in the time block.

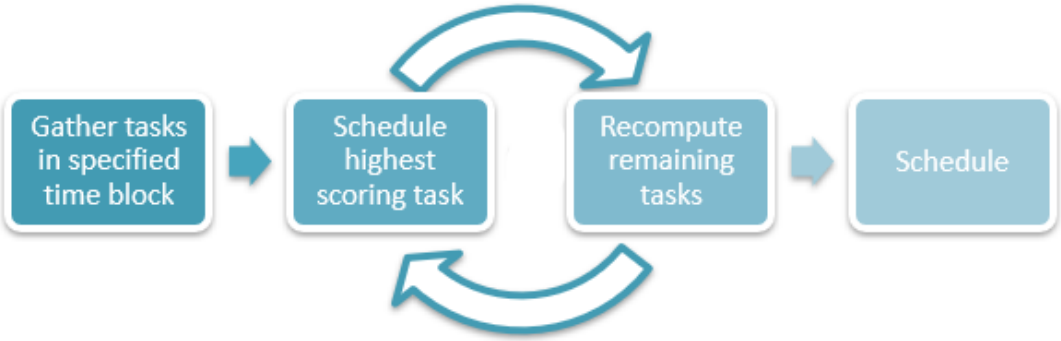


Fig. 3: Block Greedy Algorithm

This is formulated mathematically below:

$$X^*(s,[t,t+c-1]) = max_{X_k(s,[t,t+c-1])\in F} \sum_{n=0}^{N-1} [p_{ns}\mu_n] + p_s + i_s, \tag{16}$$

where $X_k(s,\ [t,\ t+c\ -1]) = \{\vec{x}_{st_0k} : \forall t_0 \in [t,t+c-1], \forall s \in [0,S-1]\}$ and $c =$ number of time windows in time block. The block greedy algorithm is not much slower than the local greedy algorithm and much faster than the WTA. Additionally, the block greedy algorithm generates a higher schedule score than all other implemented algorithms.

4. RESULTS

Recall, equation (5) considers multiple, heterogeneous sensors. Each sensor can have different costs associated with tasking it. Thus, a higher schedule score may be generated by unevenly tasking sensors. Below is a chart of the observer utilization of three heterogeneous sensors in a 24-hour schedule:

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

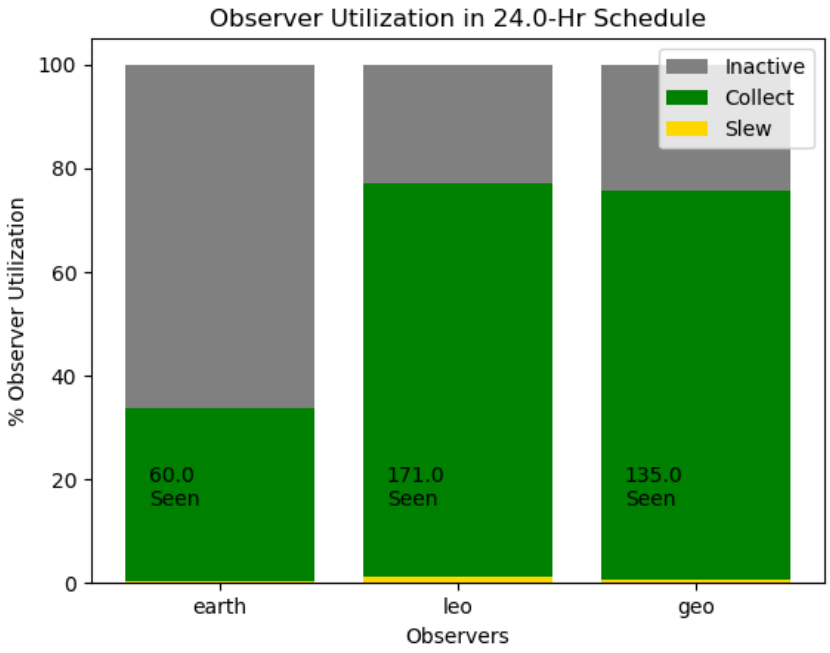


Fig. 4: Observer Utilization Percent

In this scenario, the Earth-based sensor has the lowest reward to cost ratio to task, while the LEO-based sensor has the highest. This means the difference between the positive score and negative resource costs are the most positive in the LEO-based sensor case.

The block greedy algorithm was presented as a new algorithm to solve equations (5) - (12). This is compared with the traditional greedy algorithm, the WTA algorithm, the genetic algorithm, and a random search algorithm. The block greedy algorithm is shown to overall outperform all other compared methods. It runs faster than almost all the algorithms and generates the highest score.

Below is a chart of total schedule scores per algorithm. A schedule score is computed by adding all task scores in a schedule. The figure below was generated with 970 synthesized targets, 3 observers, over 24-hours with 60 second time windows. The block greedy algorithm had 30 time window blocks. Due to the incredibly slow run time of the WTA algorithm, even at 2 time window blocks, this was run with a single time window block. As expected, the block greedy algorithm generates the highest scoring schedule, while the random search algorithm generates the lowest.

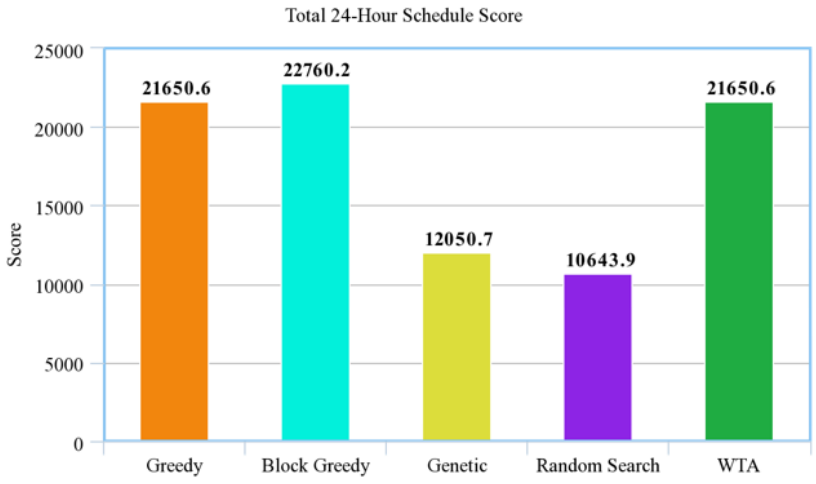


Fig. 5: Total Schedule Score by Algorithm

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

Although this schedule was run for a 24-hour interval, most algorithms were able to see all targets after a much shorter time. Below shows the amount of time each algorithm took to see all objects. Additionally, a lower max revisit time is preferable. This is analyzed below as well. In the following results, the WTA algorithm was run with a 2 time window block and there were 366 synthesized targets.

Algorithm	Time to See All Targets	Max Revisit Time
Greedy	6 hours, 48 min	10 hours, 31 min
Block Greedy	6 hours, 59 min	10 hours, 53 min
Weapon-Target-Assignment	6 hours, 48 min	10 hours, 36 min
Random Search	None	None
Genetic	16 hours, 42 min	None

Fig. 6: Time to See All Targets and Max Revisit Times Per Algorithm

The greedy and WTA algorithms were able to see all targets in the shortest amount of time. They also had the two shortest max revisit times. However, the block greedy algorithm was able to see all objects in just 2.7% more time.

Additionally, the block greedy algorithm had only 3.5% slower max revisit time than the greedy algorithm. The max revisit time was only calculated once all targets were visited twice. Hence, the random search and genetic algorithms were not able to see all targets twice in this time interval.

In terms of run time, the random search algorithm was the fastest. Given the results above, the random search algorithm performed the worst outside of this factor. The next fastest algorithm was the local greedy algorithm. The slowest algorithm was the WTA, as expected. The block greedy runs only 3 times slower than the local greedy algorithm nearly 22 times faster than the WTA algorithm.

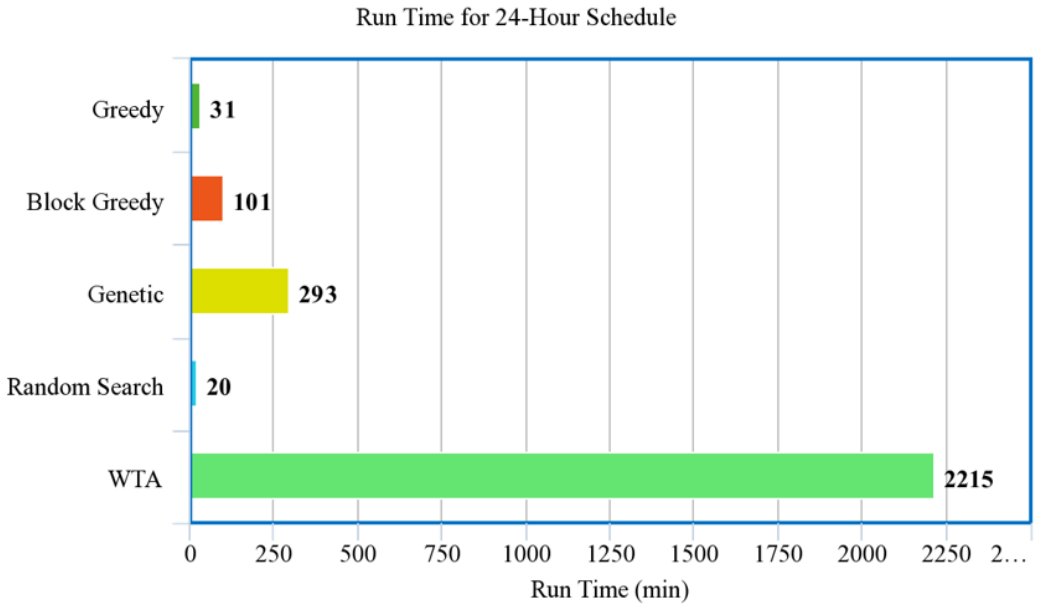


Fig. 7: Run Time Per Algorithm

Altogether, the block greedy has a much higher max score than all other algorithms compared. It successfully reduces resource costs efficiently. It runs only 3 times slower than the local greedy algorithm for 30 time windows in a block. What is sacrificed in run time using the block greedy algorithm is gained in its higher schedule score.

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

5. CONCLUSION

A robust space domain awareness scheduler has been created with realistic sensor tasking scenarios. This scheduler functions with real-life constraints such as only considering targets within observers' field of view, outside of the Earth's umbra, not blocked by the Earth, and with high enough SNR. This scheduler is capable of handling multiple, heterogeneous observers at once and creating an optimal schedule for all observers.

This scheduler considers the unique tasking cost of each observer. It computes the costs of both storing images and slewing for each observer. Not only does this scheduler maximize the total number of targets seen, it also minimizes the total cost of tasking the available observers.

Five algorithms are implemented in the SDA scheduler. Four of these algorithms are common approaches used in solving the SDA observer tasking problem. These algorithms generate a less optimal schedule than the block greedy algorithm. The block greedy algorithm borrows speed from the greedy algorithm and optimality from the WTA. When analyzed, it is found that the block greedy outperforms the other implemented algorithms. It consistently produces a higher scoring schedule and runs in a tractable amount of time.

In summary, this work thus not only introduces a new, more realistic objective function formulation for the SDA resource allocation problem, but also creates a better algorithm for tasking resources. This new SDA resource allocator both optimizes the total number of targets to be visited and successfully minimizes resource costs.

6. REFERENCES

[1] Track earth satellites given tle data, 2020.

[2] Edwin KP Chong and Stanislaw H Zak. *An introduction to optimization*. John Wiley & Sons, 2004.

[3] R Scott Erwin, Paul Albuquerque, Sudharman K Jayaweera, and Islam Hussein. Dynamic sensor tasking for space situational awareness. In *Proceedings of the 2010 American Control Conference*, pages 1153–1158. IEEE, 2010.

[4] Michael S Felten. Optimization of geosynchronous space situational awareness architectures using parallel computation. 2018.

[5] Carolin Frueh, Hauke Fielder, and Johannes Herzog. Heuristic and optimized sensor tasking observation strategies with exemplification for geosynchronous objects. *Journal of Guidance, Control, and Dynamics*, 41(5):1036–1048, 2018.

[6] Andreas Hinze, Hauke Fiedler, and Thomas Schildknecht. Optimal scheduling for geosynchronous space object follow-up observations using a genetic algorithm. In *Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. Maui Economic Development Board Maui, HW, 2016.

[7] Patrick Kim. Genetic algorithm tutorial - how to code a genetic algorithm, Jun 2017.

[8] Bryan D Little and Carolin Frueh. Ssa sensor tasking: Comparison of machine learning with classical optimization methods. In *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference*, pages 1–17, 2018.

[9] Bryan David Little. *Optical Sensor Tasking Optimization for Space Situational Awareness*. PhD thesis, Purdue University Graduate School, 2019.

[10] E. Mazareanu. Number of satellites launched by year 2019, Jun 2020.

[11] Puneet Singla, Manoranjan Majji, Nagavenkat Adurthi, Michael Mercurio, and Kumar Vishwajeet. Optimal sensor tasking for enhanced space situational awareness, 2015.

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com