2022-06

OPTIMIZING INTELLIGENCE, SURVEILLANCE, AND RECONNAISSANCE INPUTS FOR THE SYNTHETIC THEATER OPERATIONS RESEARCH MODEL

Warner, Steven M.

Monterey, CA; Naval Postgraduate School

https://hdl.handle.net/10945/70780

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

OPTIMIZING INTELLIGENCE, SURVEILLANCE, AND RECONNAISSANCE INPUTS FOR THE SYNTHETIC THEATER OPERATIONS RESEARCH MODEL

by

Steven M. Warner

June 2022

Thesis Advisor:	Johannes O. Royset
Second Reader:	Ruriko Yoshida

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2022	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE OPTIMIZING INTELLIGENCE, SURVEILLANCE, AND RECONNAISSANCE INPUTS FOR THE SYNTHETIC THEATER OPERATIONS RESEARCH MODEL	**5. FUNDING NUMBERS**
6. AUTHOR(S) Steven M. Warner	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

We consider the task of a mission planner for a unit charged with conducting intelligence, surveillance, and reconnaissance (ISR) on an area of operations in a theater-level conflict for enemy combatants using the Synthetic Theater Operations Research Model (STORM). This relies upon effective intelligence planner inputs regarding revisit interval and sensor resolution for enemy combatants. Further complicating the problem is that the entire area of operations (AO), encompassing large geographical expanses, must be searched within the mission's time constraints. Mission planning to coordinate the search assets and cover the search area while meeting intelligence inputs has proven to be difficult, often producing infeasible solutions. We present a mixed-integer linear program that effectively prescribes plans for assets to search optimally in the AO. The program is applicable, implementable, and adaptable to STORM's settings across all campaigns and produces an average of 54.6% and a median of 22.8% improvement in search coverage relative to existing heuristics.

14. SUBJECT TERMS optimal search, STORM, Synthetic Theater Operations Research Model, intelligence, surveillance, reconnaissance, ISR, mixed-integer linear program, sensor allocation, sensor optimization	**15. NUMBER OF PAGES** 81
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

OPTIMIZING INTELLIGENCE, SURVEILLANCE, AND RECONNAISSANCE INPUTS FOR THE SYNTHETIC THEATER OPERATIONS RESEARCH MODEL

Steven M. Warner
Major, United States Marine Corps
BS, Purdue University Global, 2008

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 2022

Approved by: Johannes O. Royset
Advisor

Ruriko Yoshida
Second Reader

W. Matthew Carlyle
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

We consider the task of a mission planner for a unit charged with conducting intelligence, surveillance, and reconnaissance (ISR) on an area of operations in a theater-level conflict for enemy combatants using the Synthetic Theater Operations Research Model (STORM). This relies upon effective intelligence planner inputs regarding revisit interval and sensor resolution for enemy combatants. Further complicating the problem is that the entire area of operations (AO), encompassing large geographical expanses, must be searched within the mission's time constraints. Mission planning to coordinate the search assets and cover the search area while meeting intelligence inputs has proven to be difficult, often producing infeasible solutions. We present a mixed-integer linear program that effectively prescribes plans for assets to search optimally in the AO. The program is applicable, implementable, and adaptable to STORM's settings across all campaigns and produces an average of 54.6% and a median of 22.8% improvement in search coverage relative to existing heuristics.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

List of Acronyms and Abbreviations

AD	air defense
AO	area of operations
AOI	area of interest
CBC	Computational Infrastructure for Operations Research branch and cut solver
CJCS	Chairman Joint Chiefs of Staff
COCOM	Combatant Command
CPLEX	International Business Machines ILOG CPLEX Optimization Studio
CSV	comma separated values
DOD	Department of Defense
EO	Electro-Optical
GMTI	ground moving target indicator
GUI	graphical user interface
IMINT	Imagery Analysis
IR	Infrared
ISR	intelligence, surveillance, and reconnaissance
LHS	left hand side
MCLP	maximal covering location problem
MILP	mixed integer linear program
NIIRS	National Imagery Interpretability Rating Scale

NP	non-deterministic polynomial time
NPS	Naval Postgraduate School
OP	orienteering problem
OPLAN	Operational Plan
SAR	Search and Rescue
SBL	Space Based Laser
SOM	Sensor Optimization Model
SAP	sensor allocation problem
STORM	Synthetic Theater Operations Research Model
USAF	U.S. Air Force
WAD	Warfighting Analysis Division

Executive Summary

We consider the case of using sensors on satellites to distribute looks to a geographic area discretized into grid cells. It is important to look in all grid cells at least once to see the entire area of operations; hence, we would like to initially obtain maximum coverage. Second, we would like to revisit high-priority grid cells with any extra looks. A sensor produces a two-dimensional strip in which it can look for grid cells during each pass over the geographic area which we refer to as a swath. The resolution we use to look at grid cells determines the budget of looks. We can choose to use a high resolution which gives us fewer looks with finer detail, or a low resolution which gives us more looks with coarse detail. This gives us the choice of looking at a few places accurately, or looking at many places inaccurately.

This thesis is motivated and developed in close collaboration with the J8 directorate which is part of the joint staff in the Pentagon. The J8 advises the Chairman Joint Chiefs of Staff (CJCS) on force structure, resources, and assessment. This problem has been distilled down to a general intelligence, surveillance, and reconnaissance (ISR) problem but extends to the real problem the J8 is experiencing in the theater-level campaign model called STORM. STORM uses a heuristic to determine which grid cells receive looks. The STORM heuristic tends to produce undesirable results where large areas of interest are ignored. We would like to improve on the grid cell coverage that the search assets are able to perform.

In this thesis we formulate a novel, large-scale, mixed-integer optimization model for outperforming STORM's heuristic for ISR search. The model, referred to as SOM, penalizes the number of swaths since the last look at each grid cell using a gap index. We want to avoid collecting these penalties which drives us to revisit grid cells. The objective function minimizes this penalty for incurring gaps. We use several constraints to maintain, reset and track the gap counter, a soft constraint for visiting all grid cells, and a constraint to impose a minimum resolution on grid cells. A unique feature of SOM is that it is event driven with swaths passing over the battlespace and is not based on time. SOM uses actual STORM data with over 1,300 lines of code consisting of data collection in R, processing in Python, and model implementation in Pyomo.

We implement the model on the unclassified Punic21 scenario in STORM. There are two

combatants in this scenario: Red and Blue. We can implement SOM from either point of view and each combatant produces their own variables and constraints. To illustrate SOM's large scale, in Punic21 with Red searching Blue grid cells and optimizing over 92 swaths, which correspond to a 48 hour period, we have over 25 million variables and 15 million constraints.

Case studies center both on the computational and operational results. The computational results demonstrate that we can reduce runtime by implementation of different options in International Business Machines ILOG CPLEX Optimization Studio (CPLEX)'s algorithm. The most significant option is providing a warm start using the worst possible solution of no looks occurring. For instance, when we run SOM on a single processor with over 2 million variables and 1 million constraints with default CPLEX options, it takes over 1,400 minutes and does not produce a solution. We identify tailored CPLEX options that reduce runtime and solve this instance in less than 5 minutes. This allows us to increase the size of the problem to over 22 million variables and 11 million constraints and achieve an 11% optimality gap in less than 50 minutes. Operational case study results show that SOM provides an average of 54.6% and a median of 22.8% more coverage compared to STORM. Additional options, native to SOM and not capable in STORM, ensure that SOM will outperform STORM to quickly reach maximum coverage and subsequently focus on allocating looks to the most important grid cells.

We see that the optimization model outperforms STORM's heuristic based on the operational results and allows us to balance the search of all cells, while the heuristic has a tendency to focus on important cells. SOM directs satellites where to look to allow a visit to each grid cell and avoid large revisit gaps compared to STORM's heuristic redundantly searching the same grid cells.

Acknowledgments

Much thanks to my advisor, Dr. Johannes Royset. Your skill in steering me in the right direction, supporting expertise, and guidance has made this a most enjoyable experience expanding my knowledge base and I am inspired to continue my educational journey. I sincerely appreciate the generous amounts of time you devoted to myself and this research. It has truly been an honor to work with you. To my second reader, Dr. Ruriko Yoshida, thank you for helping me start the thesis process on solid ground; your unending optimism is uplifting. To Lieutenant Colonel Matthew Bohman and Mr. Cory Culver, thank you for your time and patience in explaining the problem and answering all my questions as well as numerous Synthetic Theater Operations Research Model (STORM) walkthroughs. Your assistance has been most indispensable. Finally, I would like to thank my wife, Kristen, for continuing to raise our four children during this degree program. Your support as a mother and spouse has afforded me many opportunities and I am genuinely grateful for the hard work you do in nurturing our children.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1: Introduction

Military organizations throughout the globe strive to search large geographic areas for enemy combatants using an efficient and comprehensive search plan that makes best use of search assets. This task has proven challenging when suboptimal search plans are formed, which commonly result in large, unsearched geographic regions and inefficient use of search assets.

1.1 Problem Overview

We consider the task of a mission planner charged with conducting intelligence, surveillance, and reconnaissance (ISR) using satellites on an enemy combatant over a geographic area. The geographic area is divided into grid cells, with each grid cell assigned a value indicative of how often it needs to be searched. The planner has sensors on satellites that orbit the earth in an elliptical manner. During each pass over a geographic area, a sensor can look for objects in a two-dimensional strip, which we refer to as a swath; see Figure 1.1. However, only a portion of a swath can be looked at if the sensor were to produce high-resolution images. A mission planner would need to make a constrained decision about which grid cells within a swath should be looked at and with what resolution. The size of the geographic area and the multitude of sensors and possible resolution levels complicate the problem. The geographic area can include two or more continents and span millions of square km. Multiple satellites orbit the battlespace numerous times over a given time period. Each of these satellites has several sensors. With thousands of grid cells and hundreds of assets, a standard formulation as an optimization problem produces variables and constraints that number in the millions. A mission planner would typically like to search all grid cells at least once and then maximize the detection of enemy combatants by either revisiting grid cells and/or increasing resolution, all within a set period of time. We define this issue as sensor allocation problem (SAP). A heuristic approach to SAP would have difficulties with producing a sensor allocation plan that searches all grid cells. There might be a tendency to over allocate to certain grid cells and under allocate to others, use too high or too low resolution levels, or too long or too short revisit intervals. This heuristic approach can leave large areas unsearched and has detrimental second and third order effects on the battlefield. The suboptimal search in this case has no chance of discovering enemy combatants in unsearched regions and leaves them free to operate unrestricted during a campaign.

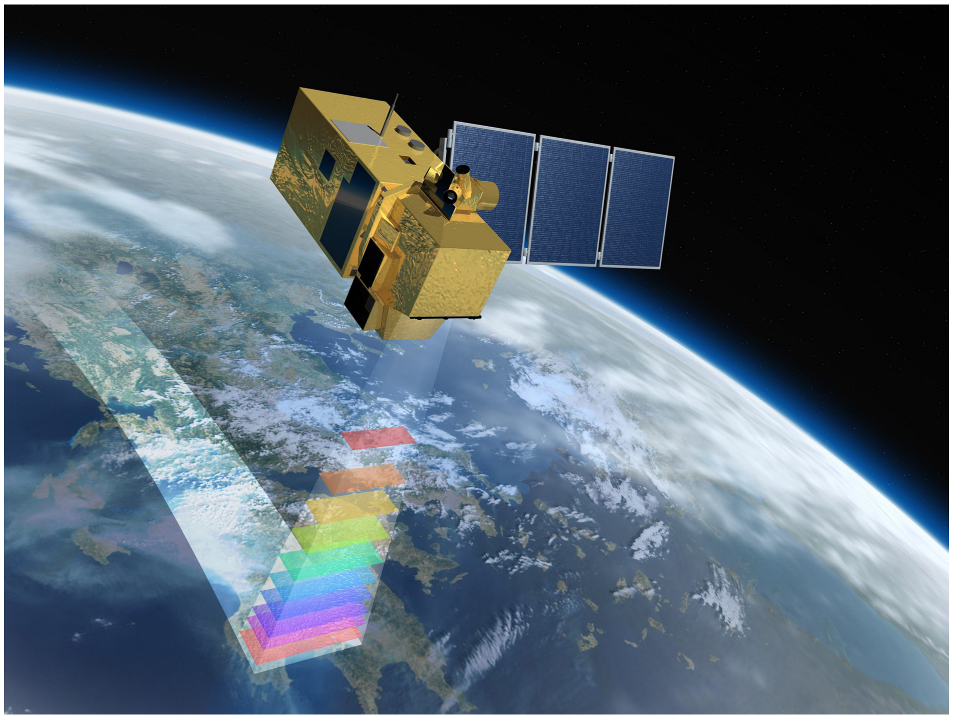


Figure 1.1. Swath depiction. Source: *Earth Imaging Journal* (2015).

We present Sensor Optimization Model (SOM) which prescribes optimal plans for utilizing ISR assets and solving SAP. In particular, SOM ensures that a sensor looks at each grid cell at least once, if feasible, and the important grid cells are revisited as often as possible. Given all grid cells with different increasing priority based upon the last time the cell received a look, SOM directs where the sensors should look on each pass of the battlespace. We implement SOM in the context of the campaign simulation tool Synthetic Theater Operations Research Model (STORM) and demonstrate that the obtained search plans outperform those generated by STORM's heuristic algorithm.

1.2 Literature Review

SAP shares similarities with the orienteering problem (OP) as well as multi-agent search problems, both of which are studied extensively. Tsiligirides (1984) introduces the OP where competitors use land navigation to race to visit checkpoints. Competitors seek to maximize their score by visiting as many checkpoints as possible within the allotted time, often referred to as a score orienteering event (Tsiligirides 1984). In its most basic form, the OP combines node selection with shortest path and relates to both the Knapsack and Travelling Salesman Problems (Gunawan et al. 2016). The OP applies to the fields of logistics for vehicle routing and delivery, tourism for visiting attractions, and military for surveillance activities (Vansteenwegen et al. 2011).

There are numerous variants of the OP. Examples deviate from a team OP approach, which seeks to find multiple paths that maximize a collective score, or incorporating time windows for specific node visitation (Vansteenwegen et al. 2011). The Generalized OP, a variant which uses a nonlinear objective function for computing the overall score for the competitor, has been studied and solved with different methods (Wang et al. 1995; Geem et al. 2005; Wang et al. 2008). Ilhan et al. (2008) studied the OP with stochastic profits where nodes are assigned normally distributed random profits with the goal of exceeding a certain profit threshold. Gendreau et al. (1998) incorporated compulsory vertices into the OP which forced requirements on certain checkpoints to be visited. Fomin and Lingas (2002) considered the time-dependent OP where the travel time between checkpoints depends upon the competitors start time. The capacitated OP was introduced by Archetti et al. (2009) and modeled a limited capacity on the total checkpoints that each competitor could visit. Pietz and Royset (2013) introduced varying quantities of rewards at each node which depended upon resource expenditure to reach a specific node. The multi-profit OP varies profit by time spent at each checkpoint and requires selecting both checkpoint sequence and the time spent at each checkpoint (Kim et al. 2020). SAP most aligns with the categories of a team OP approach, compulsory vertices, limited capacity checkpoints, and varying rewards.

Each of these OP problems is non-deterministic polynomial time (NP)-hard which means that producing an exact solution will most likely be very time consuming. Efforts have focused on reducing runtimes through specialized branch-and-bound algorithms (see, e.g., Pietz and Royset 2013), neural networks (see, e.g., Wang et al. 1995, 2008), heuristics (see, e.g., Kim et al. 2020; Archetti et al. 2009; Gendreau et al. 1998), and various other algorithms to include dynamic programming, approximation, bi-objective genetic, and harmonious (see, e.g., Peng et al. 2020; Fomin and Lingas 2002; Ilhan et al. 2008; Geem et al. 2005).

The OP connects to search problems where specific areas of interest to search are spread about a graph, and search assets are subjected to budget constraints. Searches in this manner are modeled as an OP if using a single search asset or a team OP if using multiple search assets (Ding 2018). An example showcasing this relatability to SAP, and tying in with optimal search theory, is when Peng et al. (2020) applies the OP to the scheduling of satellites for target looks. In Peng et al. (2020), each target has an associated profit and time window for a look; however, satellites focus on manipulation of look angle for image quality and the algorithm is based on a single orbit.

Optimal search theory focuses on searching for a stationary or moving target. For a stationary search, the problem associates a prior distribution to the location of the target. It uses a detection function that relates search time with probability of detection for a target given it is in that specific cell (Stone et al. 2016). Stationary search then focuses on allocating search assets amongst cells to maximize the probability of target detection. Much of the work focused on search of a stationary object with one search asset occurred several decades ago from 1946 to 1982 and is listed in Ding (2018). Song and Teneketzis (2004) expanded stationary search to include multiple search assets but did not accommodate for search assets that are only able to search a portion of the geographic area. Searching for a moving target requires modeling a target's movement through potential paths using a probability distribution. The last three decades of work in search theory focus on moving targets with one or multiple search assets using algorithms based on branch-and-bound (Dell et al. 1996; Washburn 1998; Lau et al. 2008; Bourque 2019; Sato and Royset 2010), cutting plane (Royset and Sato 2010), open loop network flow (Berger et al. 2021), and heuristics (Dell et al. 1996; Abi-Zeid et al. 2019).

Various military applications in search theory abound; for example, Yoash et al. (2018) focuses on finding an optimal search pattern over a small scale search area covered by a single aviation asset to obtain full coverage in search of a single adversary. Kress et al. (2012) examine search and interdiction with multiple moving adversaries and two types of search assets. They pose a greedy heuristic due to their primary model being computationally intractable regarding realistic application.

Both stationary and moving search problems are conducted over a set time period and are limited by the effort available for the search. SAP shares similarities since we are searching over a set time period and have a budget associated with the search effort; however, we are focused on covering and monitoring cells in a suitable manner and not detecting one or more specific targets.

The maximal covering location problem (MCLP) is a thoroughly studied NP-hard problem in which we are given a number of sets that have some elements in common and we seek to choose a limited number of these sets which maximize the total coverage of elements (Church and ReVelle 1974). Hoogendoorn (2017) gives an overview of the different algorithmic approaches applied to MCLP within the past several decades. Efforts within the past decade have proposed branch and cut through benders decomposition (see, e.g., Cordeau et al. 2019; Vatsa and Jayaswal 2021), greedy heuristics (see, e.g., Sekar Amarilies et al. 2020; Seyhan et al. 2018), metaheursitics (see, e.g., Pereira et al. 2015), minmax regret model (see, e.g., Baldomero-Naranjo et al. 2021), and simulated annealing (see, e.g., Zarandi et al. 2013). Price et al. (2022) completed recent work regarding military application of MCLP over the Western Hemisphere on counterdrug efforts. They extend the generic MCLP to three different mixed integer linear program (MILP)s in the context of counterdrug trafficking. Their approach satisfies both multiple drug cartels and multiple counterdrug assets while identifying the best interdiction locations. However, their formulations only model a small subset of operationally relevant constraints.

The combination of an OP, stationary search, and MCLP, with specific constraints for integration with a large scale campaign model such as STORM, differentiates SAP from previously studied problems. Additionally, the specific nature of a military campaign scenario covering an expansive geographic area with thousands of cells and hundreds of swaths equating to millions of variables compounds the complexity of SAP. We present a novel MILP which solves SAP by optimally distributing looks from sensors to grid cells. Results show we can capitalize on different International Business Machines ILOG CPLEX Optimization Studio (CPLEX) algorithm options to significantly reduce runtime as well as increase coverage compared to existing heuristics.

1.3 Thesis Organization

In Chapter 2 we discuss SAP through the lens of the campaign analysis tool called STORM.

In Chapter 3 we formulate SOM, a novel MILP which solves SAP. SOM directs sensors on satellites to look at specific cells in a swath. Rigorous data collection is a requisite for STORM integration and we discuss this in further detail.

In Chapter 4 we test SOM on data produced through STORM. We analyze how SOM compares to STORM's heuristic grid cell search and explore several excursions of sponsor related questions in further detail.

Chapter 5 provides a summary of our work and recommends topics for future research.

CHAPTER 2: STORM Context

To conduct campaign analysis in STORM, sensor allocation is required and we thus provide a brief overview of STORM's purpose, key terms, and pertinent literature. We provide further background and definitions as well as refine and scope SAP through our sponsor, the J8 Directorate in the Pentagon, which gives the necessary context to understand how SAP relates to SOM.

2.1 STORM Overview and Literature

STORM began as a U.S. Air Force (USAF) project in 1996 to simulate both aviation and space power across the joint military (Group W 2019a). STORM has since experienced several enhancements and updates to incorporate multiple domains including the maritime environment for naval combat and expeditionary warfare (Group W 2019a). Current STORM users range widely, consisting of military branches, numerous high level staffs such the Joint Staff and Combatant Command (COCOM)s and different international users (Group W 2019a). STORM is designed to simulate theater level conflict while representing all warfighting domains across the range of military operations. STORM has functional area tools which are separated into three categories: input, execution, and output. Input is primarily composed of over one hundred text data files which provide definitions and background, permit user changes, and are used to run the simulation. To generate the primary output, the user subsequently calls the execution tool to run the input data through STORM which creates a dbase.out file. This file populates a data warehouse which can then be processed by the user to generate analytical results through different mapping, graphing, and reporting components. These main functional area tools are accessed through STORM's graphical user interface (GUI). STORM's stochastic nature is demonstrated when the campaign level simulation scenario input data is replicated and executed multiple times, giving the user satisfactory confidence in the results. On the unclassified side, STORM has two modern, prebuilt scenarios titled Punic21 and Wona. This thesis uses the Punic21 scenario for primary analysis. The Punic21 scenario has two combatants, Blue and Red, and takes place in the geographic regions of Europe and North Africa. STORM has three manuals directed at informing and educating the different individual users: the Programmer's Manual, the User's Manual and the Analyst's Manual (Group W 2019a,d,c,b). The Programmer's Manual is tailored for development, maintenance, and modification of STORM's base code (Group W 2019a). The User's Manual focuses on navigating the graphical user interface (GUI) and the Analyst's Manual concentrates on simulating campaign-level combat to generate and analyze results that aid in decision making (Group W 2019a). STORM also has an associated Appendices document which serve as a single reference for all types of users (Group W 2019b).

2.2 STORM Background and Definitions

To fully understand SAP's relation to STORM, a background of STORM's ISR, battlespace, satellites, sensors, and resolution is necessary.

2.2.1 ISR

ISR in STORM consists of a fixed number of satellites and aircraft prior to campaign execution that look at both geographic regions and individual targets. Combatants use their assigned ISR assets to discover and further develop targets to strike, as well as conduct post-strike analysis through follow on assessment and monitoring (Group W 2019a).

2.2.2 Battlespace

In STORM, the battlespace encompasses all areas of interest for a campaign. Each combatant in a campaign will have different area of interest (AOI)s that they likely occupy while operating. The opposing forces COCOM planner assigns types and establishes boundaries for these AOIs in order to search them for intelligence collection and targets. The type of AOI determines the priority of search for that area. For instance, a "Naval Search Area" covering large expanses of ocean may have a lower priority compared to an "Air Defense" area covering detailed land. STORM permits AOIs to change type based on pre-determined conditions as the scenario progresses. In the Punic21 scenario, the "Blue East Med AOI" is initially assigned as a Naval Search Area but changes to a ground moving target indicator (GMTI) Search Area once the number of ships in the East Mediterranean Sea is less than five. In STORM's terminology, AOIs are named after the combatant that will be searching them but owned by the combatant that is likely to be found inside that AOI. For instance, the Blue combatant would use Blue ISR assets to search for Red targets in the AOI owned by the Red combatant which is titled “Blue Italy AOI.”

The entire battlespace in STORM is divided into grid cells to save on efficiencies in computing (Group W 2019a). The default setting for grid cell size is 50 km wide by 50 km long which equates to 2,500 square km. The default setting for number of grid cells is a width and depth of 60 by 60 which equates to 3,600 total cells. However, for the Punic21 scenario, AOIs for both sides expand beyond the 60 by 60 square grid. To ensure entire scenario coverage, grid cell expansion to 120 by 120 was necessary which resulted in 14,400 total grid cells covering 36 million square km. We henceforth refer to targets as grid cells since that is the primary focus. Figure 2.1 depicts the required expansion by showing a 60 by 60 grid cell overlay on top of Red AOIs.

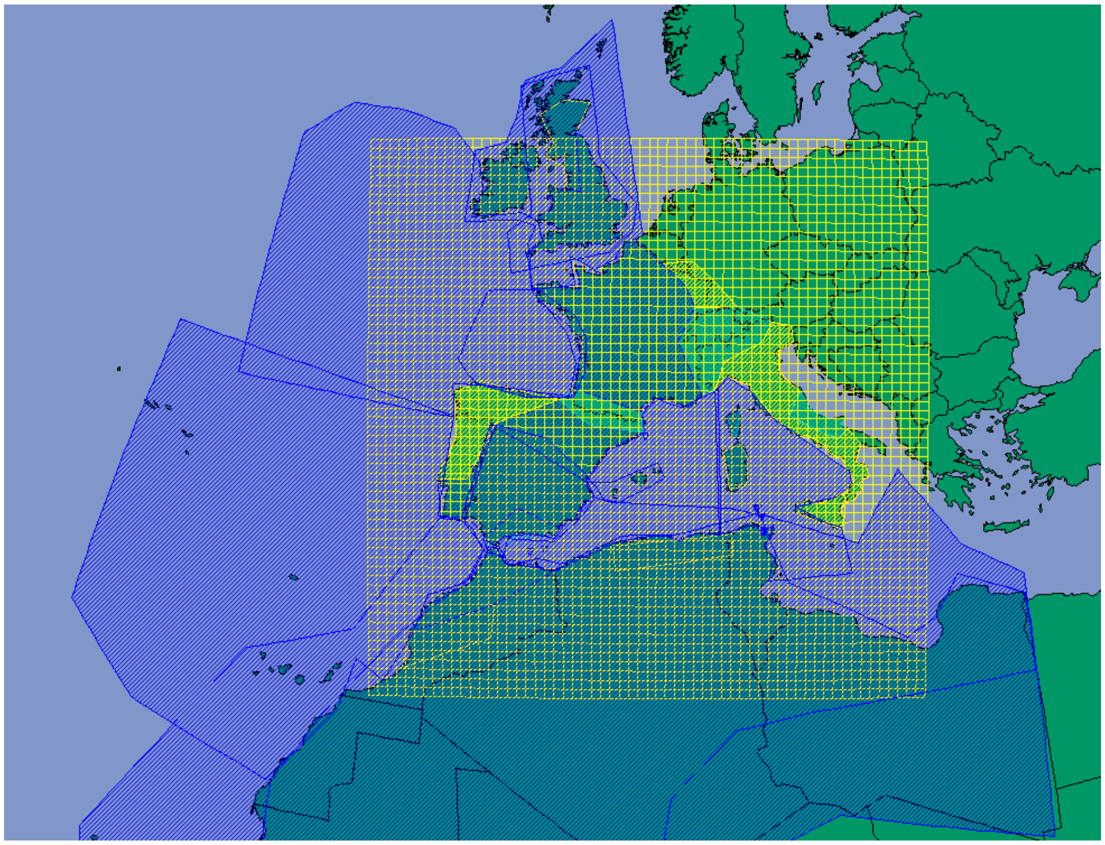


Figure 2.1. Screen capture from the Punic21 scenario in STORM with two overlays: a 60x60 yellow grid and Red AOIs which are depicted in blue. We notice the grid cells do not cover all the AOIs and must be expanded. Source: *Group W* (2005).

2.2.3 Satellites

Different satellite types perform different functions for each combatant such as reconnaissance, spectrum surveillance, or GPS and are positioned in low, medium, or high earth orbits. These satellites follow a dual body Kepler orbit based upon a spherical earth (Group W 2019a). The orbit distance from earth combined with the Keplerian motion determines how often a satellite passes over the battlespace. Prior to simulation execution, STORM creates a script of orbital asset interactions with the battlespace using the Orbital Tool (Group W 2019a). The Orbital Tool produces an OrbitalEvent file that lists satellite entry and exit latitude and longitude as well as an instantaneous time the satellite passed over the battlespace. To focus on a broad area search, this thesis has been scoped down to specifically examine satellites searching geographic regions in the ISR representation.

2.2.4 Sensors

Each satellite has a specific number of sensors on it which determines the function the satellite performs for the combatant. We are primarily concerned with reconnaissance sensors since they are designed to look at geographic regions. The two types of satellites that have sensors that perform reconnaissance are the Reconnaissance and the Space Based Laser (SBL) satellites. These two types of satellites make approximately 4 to 6 passes over the battlespace every 24 hour period. In Punic21, both combatants have four Reconnaissance satellites and one SBL satellite. The Reconnaissance satellites have four sensors on them compared to SBL which has three sensors. These sensors are named after the type of functions they perform which are Electro-Optical (EO) Imagery Analysis (IMINT), Infrared (IR) Day IMINT, IR Night IMINT, and Search and Rescue (SAR) IMINT. Each time a satellite passes over the battlespace, individual sensors produce a swath (Section 1.1; Figures 2.2 and 2.3). Since sensors provide the lowest echelon search asset, we will henceforth group the various levels of search assets into sensors to simplify terminology.

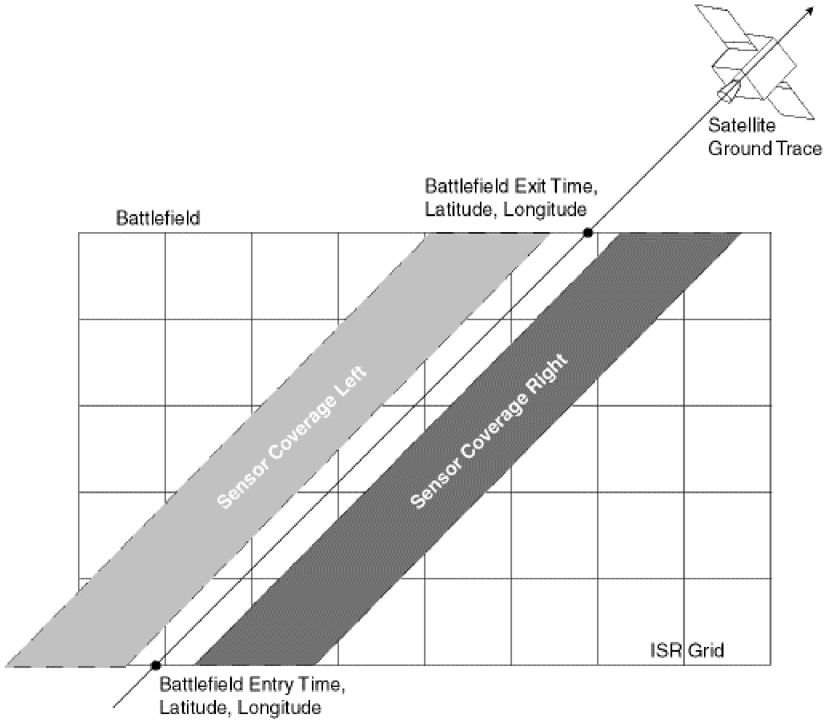


Figure 2.2. Orbital asset swath representation. Source: Group W (2019a).

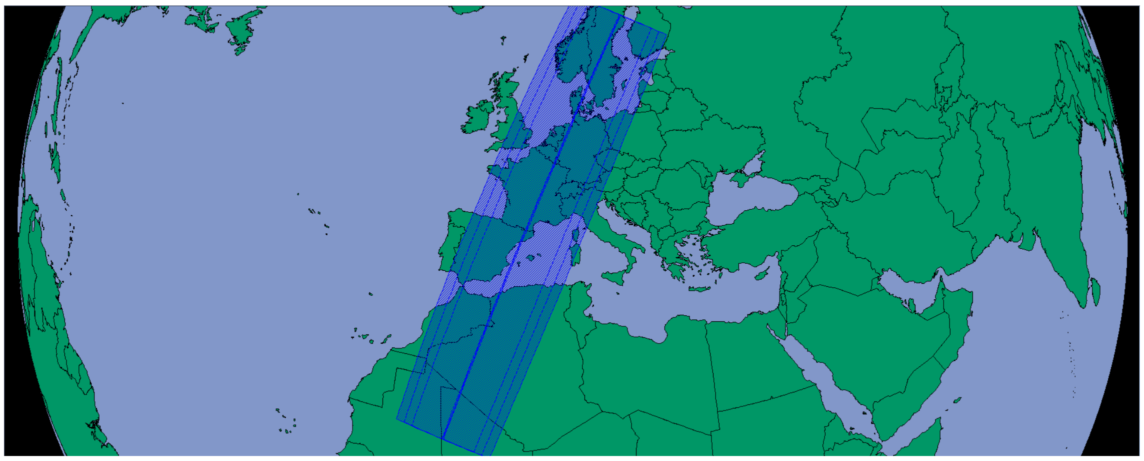
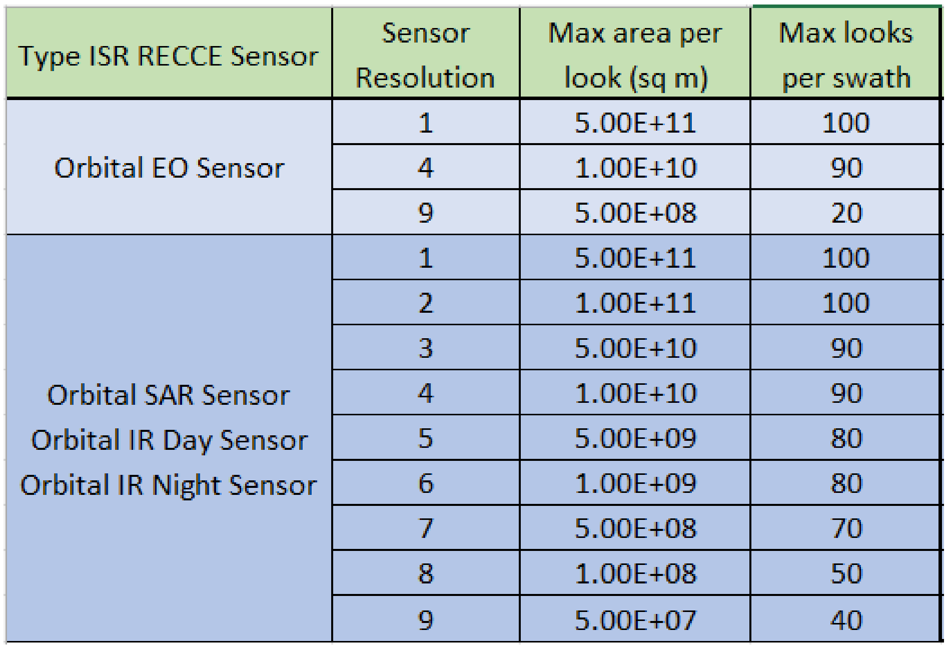


Figure 2.3. A screen capture from the Punic21 scenario in STORM of a single satellite passing over the battlespace producing various swaths from its sensors. Source: *Group W* (2005).

2.2.5 Resolution

Each sensor has a limited number of looks it can allocate to grid cells per swath which is determined by the adopted resolution levels. These resolutions follow the National Imagery Interpretability Rating Scale (NIIRS) and have at most nine discrete levels. A higher NIIRS level equates to a more detailed look and an increased likelihood of detecting something in that grid cell. However, setting a high NIIRS level serves as a trade-off with less looks to distribute per swath compared to setting a low NIIRS level. The limit on the number of looks for a swath is determined by two separate calculations, one involving the square area that is viewed in the look and the other involving the total look count (Figure 2.4). We refer to these as the square area resource budget and the look count budget.



Type ISR RECCE Sensor	Sensor Resolution	Max area per look (sq m)	Max looks per swath
Orbital EO Sensor	1	5.00E+11	100
	4	1.00E+10	90
	9	5.00E+08	20
Orbital SAR Sensor Orbital IR Day Sensor Orbital IR Night Sensor	1	5.00E+11	100
	2	1.00E+11	100
	3	5.00E+10	90
	4	1.00E+10	90
	5	5.00E+09	80
	6	1.00E+09	80
	7	5.00E+08	70
	8	1.00E+08	50
	9	5.00E+07	40

Figure 2.4. STORM's max area per look and max looks per swath for the listed sensors NIIRS level resolutions. Source: *Group W* (2005).

We explain these budgets in detail which are defined in *Group W* (2005). Since a sensor can use different resolutions for separate grid cell looks, both the square area resource budget and the look count budget are normalized to an initial value of one. When a sensor passes over the battlespace and produces a swath, each grid cell it looks at encompasses a square area. The default STORM grid cell square area is 2,500 square km or $2.5 \cdot 10^9$ square meters.

We update the square area resource budget by subtracting

$$\frac{\text{square area}}{\text{max area per look}} \tag{2.1}$$

for each look during a swath. For instance, if we use the first look on a grid cell with the Orbital EO Sensor using NIIRS 4, we update the square area resource budget by subtracting

$$1 - \frac{2.5 \cdot 10^{9}}{1.0 \cdot 10^{10}} \tag{2.2}$$

to obtain the updated sensor square area resource budget of $\frac{3}{4}$. The numerator in this case is the grid cell area in square meters and the denominator is obtained from Figure 2.4. We discount each subsequent look during the swath in the same manner from the updated square area resource budget. Additionally, each

$$\frac{1.0}{\text{max looks per swath}} \tag{2.3}$$

counts against the look count budget for each swath. This budget starts at one and is discounted by each look during a swath. For example, for the first look during a swath we compute the updated look count budget by subtracting

$$1 - \frac{1.0}{90} \tag{2.4}$$

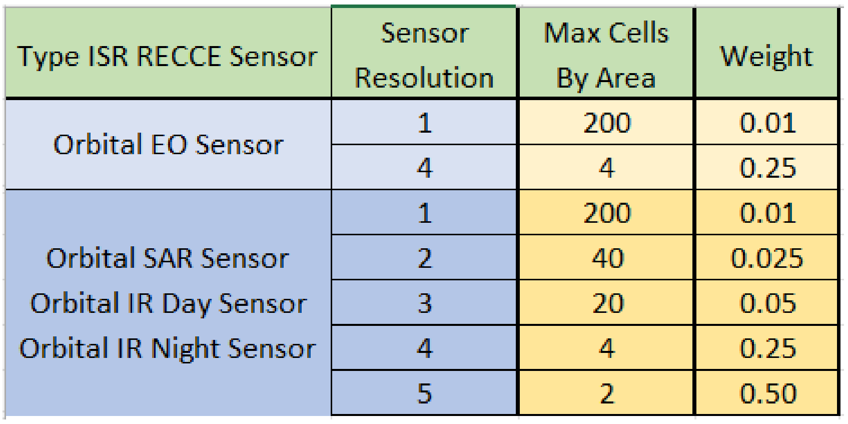
which gives us a remaining budget of $\frac{89}{90}$. Since each grid cell has the same area, we compute how many grid cells we can look at during a swath with a particular resolution by dividing

$$\frac{\text{grid cell square area}}{\text{max area per look}} \tag{2.5}$$

for each value in column three of Figure 2.4. This value is input into column three of Figure 2.5 to create the max cells by area. We assign a weight associated with each look using a specific resolution level which is

$$\frac{1}{\min\left(\text{max looks per swath, max cells by area}\right)} \tag{2.6}$$

and depicted in column four of Figure 2.5. We eliminate NIIRS nine for the orbital EO Sensor and NIIRS six through NIIRS nine for all other sensors since only a fraction of a grid cell is capable of being looked at with those resolutions.



Type ISR RECCE Sensor	Sensor Resolution	Max Cells By Area	Weight
Orbital EO Sensor	1	200	0.01
	4	4	0.25
Orbital SAR Sensor Orbital IR Day Sensor Orbital IR Night Sensor	1	200	0.01
	2	40	0.025
	3	20	0.05
	4	4	0.25
	5	2	0.50

Figure 2.5. STORM's resolution level budget for the listed sensors with assigned weights. Source: *Group W* (2005).

We thus produce an updated budget chart depicted in Figure 2.5. From this updated budget chart, we produce a bar graph which shows the number of looks per swath the available resolutions are capable of producing (Figure 2.6).

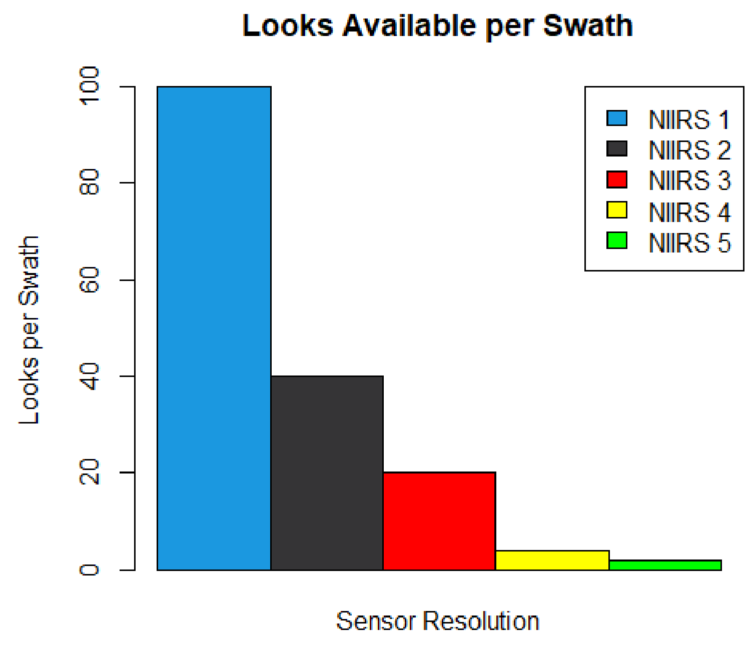


Figure 2.6. STORM's number of looks available per resolution level for a swath. Source: *Group W* (2005).

2.3 J8 Directorate Problem Set

The J8 Directorate of the Joint Staff at the Pentagon supports the Chairman Joint Chiefs of Staff (CJCS) in development and evaluation of force structure requirements (Joint Chiefs of Staff 2022). To accomplish their mission, the J8 conducts studies, analysis, and assessments and evaluates plans, programs, and strategies. Within the J8 is the Studies and Analysis branch which is further subdivided into the Warfighting Analysis Division (WAD). The WAD provides campaign analysis for both Joint staff and COCOMs on Operational Plan (OPLAN)s of record in STORM. The J8 is currently experiencing issues with STORM's ISR representation.

The ISR cycle is used to discover, develop, strike, assess, and monitor grid cells during a campaign. Thus, accurate ISR representation is vital to correctly model realistic outcomes.

A key element to the effectiveness of an ISR asset such as a sensor looking at a grid cell is how grid cells are prioritized and the timeliness of their looks. These aspects of a look are captured in the collection plan supplied by the COCOM planners. The collection plan is designed to optimize the process of allocating looks; however, it may not prove optimal, realistic, or feasible while being modeled in STORM for ISR coverage. One major issue is that STORM determines ISR grid cell priorities using a heuristic collection approach (Group W 2019a). STORM takes 13 different COCOM inputs supplied by the COCOM planner to compute a collection score and assign it to each cell in a specific AOI (Group W 2019a). As the time between a grid cell look increases, the collection score increases until it reaches a peak priority input. Figure 2.7 depicts three grid cells, each with a different collection score.

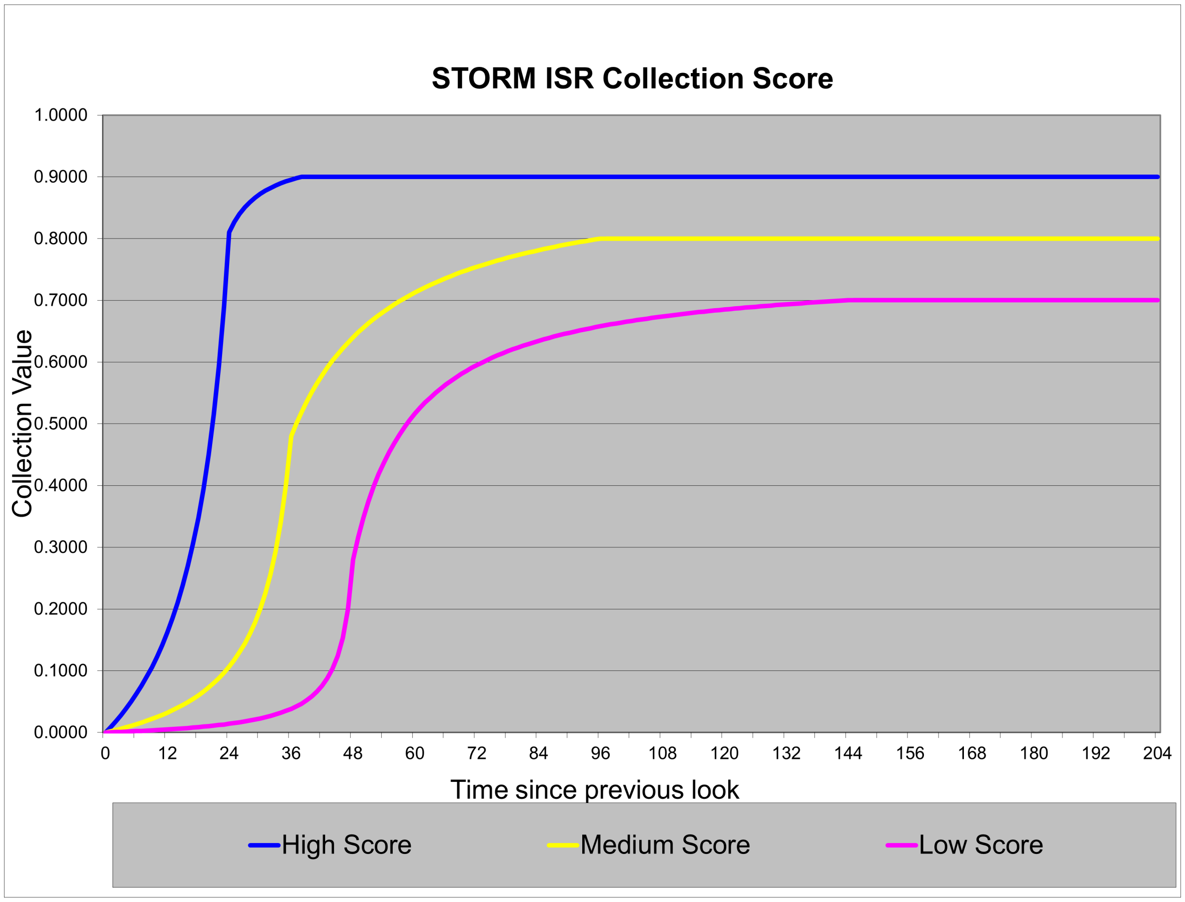


Figure 2.7. Each color line depicts a collection score for a different type of grid cell. Time is on the x-axis and the associated collection score is on the y-axis.

STORM bases the heuristic collection approach upon the highest collection score. STORM's search heuristic examines all cells in a particular swath and decides which cell to look at based upon the highest collection score, proceeding in a top to bottom, left to right manner to break collection score ties (Figure 2.8).

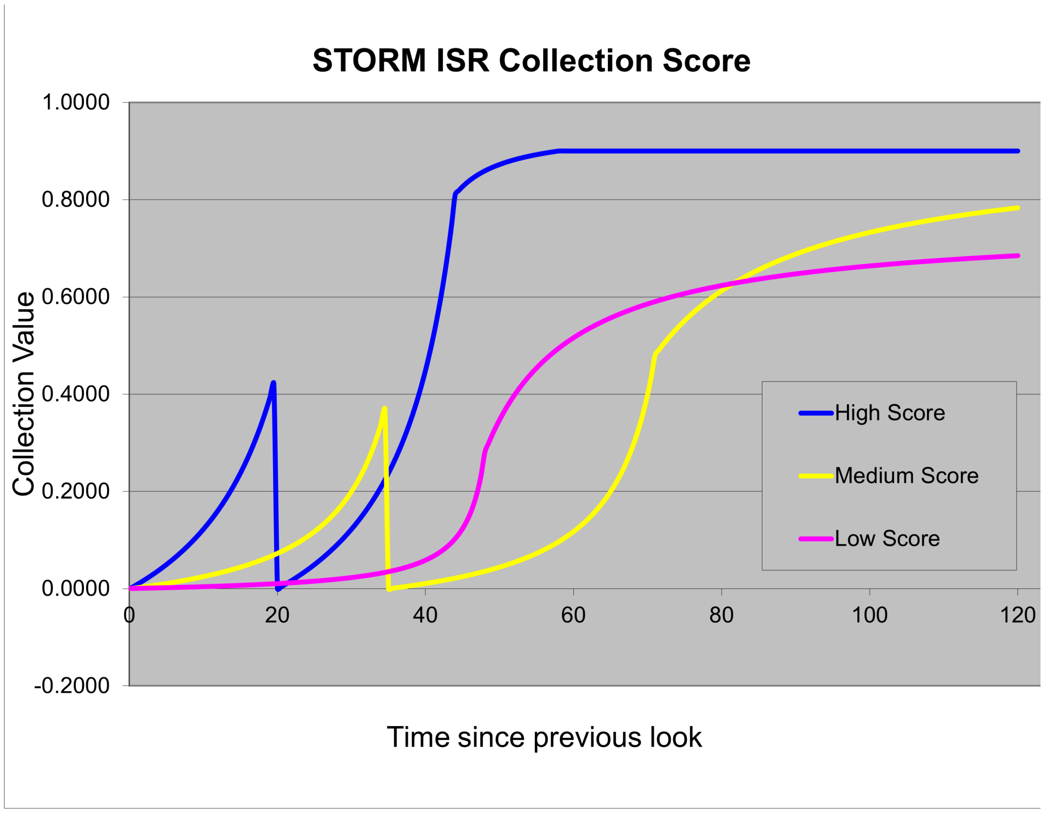


Figure 2.8. Each color line depicts a collection score for a different type of grid cell. Time is on the x-axis and the associated collection score is on the y-axis. Two looks were available at times 20 and 35 respectively. The look went to the highest score at the time; hence the blue grid cell received a look and the yellow grid cell received a look.

While STORM uses the terminology of collection score, we henceforth refer to this as the collection penalty since the word score is associated with maximization but we would like to minimize this number.

This manner of allocating looks to grid cells does not optimally balance looks over a given time period and creates extreme difficulty in ensuring all grid cells receive at least one look. There is no direct indication to the mission planner of efficient use of orbital assets in a scenario. This heuristic approach is commonly referred to as a "greedy" algorithm because it selects the locally optimal cells that are immediately available during a swath without regard to future consequences (Rardin 2016). STORM's greedy heuristic fails to balance optimal looks over an entire scenario, resulting in skewed campaign outcomes if large swaths of search area go unobserved. We see an example of STORM's greedy algorithm leaving unsearched areas in Figure 2.9.

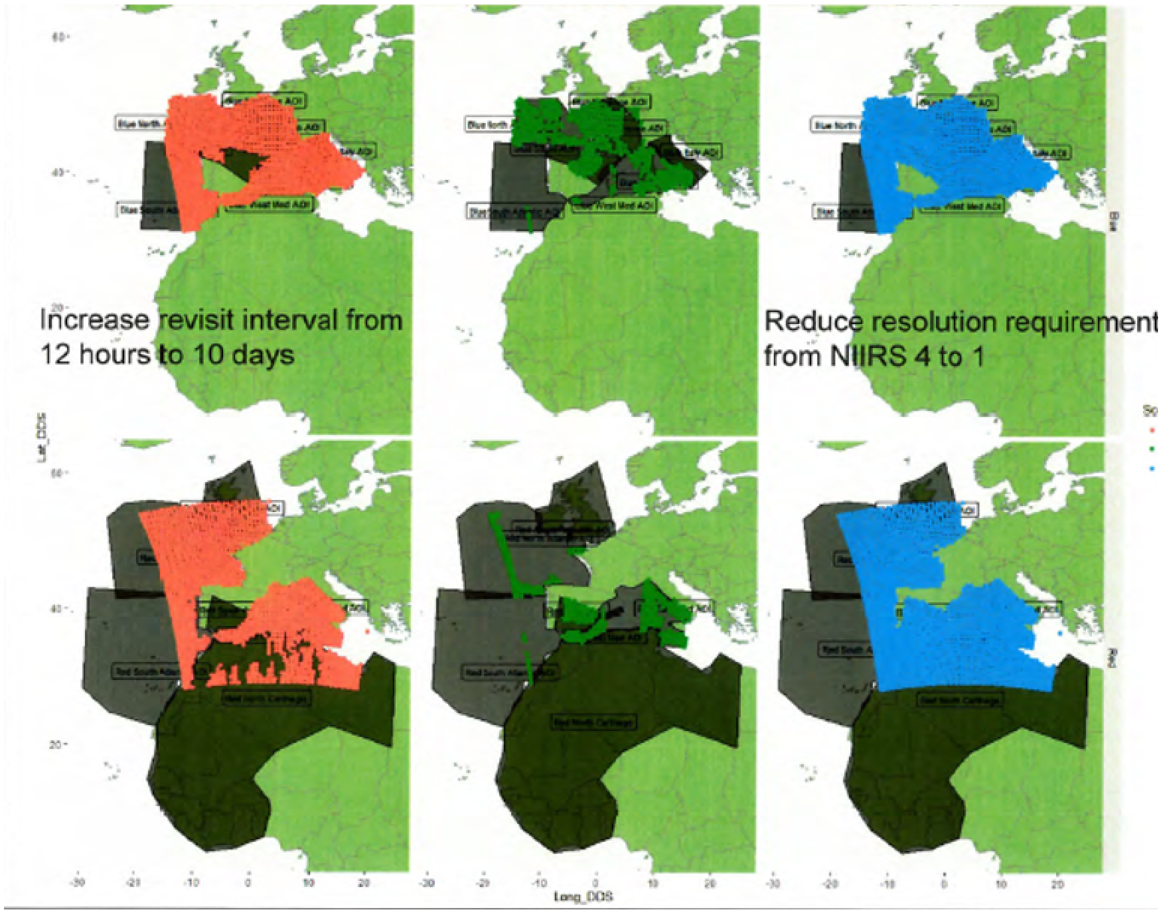

Increase revisit interval from 12 hours to 10 days
Reduce resolution requirement from NIIRS 4 to 1
Lat_DDS
Long_DDS

Figure 2.9. The top row represents Blue searching for Red grid cells and the bottom row represents Red searching for Blue grid cells in the Punic21 scenario. Searched cells are overlayed over the black AOIs in red, green, and blue colors respectively. Column 2 is the baseline case in STORM and shows a significant amount of unsearched area. Column 1 shows a scenario where the revisit interval for grid cells has been increased from 12 hours to 10 days and depicts an increase in the number of grid cell looks. Column 3 shows a scenario where the minimum grid cell resolution was decreased from NIIRS 4 to NIIRS 1 and depicts an increase in the number of grid cell looks. Some of the unsearched black cells in both column 1 and column 3 are due to the grid being limited to 60x60 grid cells.

CHAPTER 3:
Model

In this section, we address SAP by presenting SOM and discuss data collection. We provide model justification through an explanation of indices, data, decision variables, objective function, and constraints. We explain the data collection process of parsing STORM outputs through a series of steps to obtain usable input for SOM.

3.1 SOM Formulation

Since we formulate a MILP, we make assumptions on both time and the geographic region. We assume that time can be discretized into instantaneous sensor passes. Since each sensor produces a swath as it passes over the battlespace, we are able to base the model on swaths. Thus, a unique aspect of this model is that it is event driven, reliant on swaths passing over the battlespace and not based on time. In STORM, each swath has an instantaneous time that it passes over the battlespace which can be referenced via a lookup table. Since swaths have a pre-built accounting of time, a specific number of swaths has an associated time period. For example, over a 24 hour period Blue satellites produce 43 swaths and Red satellites produce 46 swaths that contain grid cells the particular combatant is interested in (Table 3.1).

Table 3.1. Hours in the Punic21 scenario and number of swaths produced.

Scenario Hours	Blue Swaths	Red Swaths
6	10	12
12	34	36
24	43	46
48	86	92
72	121	138

We additionally make the assumption that the geographic region can be discretized into a number of grid cells. For indices, we avoid indexing over AOIs by specifying individual grid cells which assists in limiting the number of variables.

The model, denoted as SOM, is a unique mixture of an OP, stationary search, and MCLP. SOM takes the following form.

Indices

c	grid cell $\{1, ..., C\}$
r	resolution $\{1, ..., R\}$
s	swath $\{0, 1, ..., S\}$
g	gap $\{0, 1, ..., G\}$

Data

$pen_{c,s,g}$	penalty in grid cell c for swath s at a gap of g
$rmin_c$	minimum resolution level for a look in grid cell c
$wei_{s,r}$	weight for swath s using resolution r
δ	max number of looks permitted with resolution level less than $rmin_c$
m	large constant for grid cell non-looks
n	large constant for significant swath gaps between looks

Decision Variables

$$X_{c,s,r} = \begin{cases} 1 \text{ if grid cell } c \text{ receives a look with resolution } r \text{ from swath } s \\ 0 \text{ otherwise} \end{cases}$$

$$Y_{c,s,g} = \begin{cases} 1 \text{ if, at swath } s\text{, there is } g \text{ swaths since last look in grid cell } c \\ 0 \text{ otherwise} \end{cases}$$

$J_{c,s}$ = at swath s, number of swaths since last look for grid cell c

Z_c = amount by which the at-least-once constraint is violated

Objective Function

$$\text{minimize} \sum_{c,s,g} pen_{c,s,g} Y_{c,s,g} + m \sum_{c} Z_c \qquad (3.1)$$

Subject to

$$\sum_g gY_{c,s,g} = J_{c,s} \qquad \forall c, s \tag{3.2}$$

$$\sum_g Y_{c,s,g} = 1 \qquad \forall c, s \tag{3.3}$$

$$1 + J_{c,s-1} \leq J_{c,s} + n \sum_r X_{c,s,r} \qquad \forall c, s > 0 \tag{3.4}$$

$$\sum_{s,r} X_{c,s,r} \geq 1 - Z_c \qquad \forall c \tag{3.5}$$

$$\sum_{c,r} wei_{s,r}\ X_{c,s,r} \leq 1 \qquad \forall s \tag{3.6}$$

$$\sum_{s,r<rmin_c} X_{c,s,r} \leq \delta \qquad \forall c \tag{3.7}$$

$$X_{c,0,r} = 0 \qquad \forall c, r \tag{3.8}$$

$$Y_{c,s,g} = 0 \qquad \forall c, s, g > s \tag{3.9}$$

$$0 \leq J_{c,s} \qquad \forall c, s \tag{3.10}$$

$$J_{c,0} = 0 \qquad \forall c \tag{3.11}$$

$$0 \leq Z_c \leq 1 \qquad \forall c \tag{3.12}$$

The objective function in (3.1) treats the collection penalty as a coefficient and seeks to minimize this penalty multiplied by the number of grid cells that received no looks. Any unobserved cells are tracked through the Z variable and penalized with the large m coefficient which we also seek to minimize. We choose m to be sufficiently large in proportion to the penalty computed in the first term. The value of m ensures enforcement of constraint (3.5) to search all grid cells.

The gap index is tied to the grid cells that receive no looks and hence tied to the objective function. As time continues without a look, the gap index continues to increase, and, in turn, the count of $Y_{c,s,g}$ variables increase. We can associate gaps with the "Time since previous look" in Figure 2.7 for simplicity's sake but we use a lookup table since the time between swaths steps do not occur at uniform times. We can think of the blue, yellow, and pink lines as individual grid cells in Figure 2.8. All three grid cells have a gap of 19 when the "Time since previous look" equals 19. The first look is given at time 20 to the largest collection penalty, the blue grid cell. The blue grid cell now has a gap of 0 at time 20. The yellow and pink grid cells continue with gaps of 20 each since they did not receive a look. The next available look occurs at time 35 which is given to the yellow grid cell. At this time the blue grid cell has a gap of 15, the yellow has a gap of 0, and the pink has a gap of 35. We seek to minimize the collection penalty multiplied by the number of grid cells that receive no looks which consequently incorporate the gaps for these cells.

We use constraint (3.2) to maintain the value of the gap to track the total gaps between looks. Constraint (3.3) keeps the current $Y_{c,s,g}$ variable at 1 and turns previous $Y_{c,s,g}$ variables to 0. Hence, as time steps forward, old $Y_{c,s,g}$ variables turn to 0. This ensures old gaps aside from the current gap are not consecutively summed over in constraint (3.2). For example, when $s = 3$, $Y_{c,s,g}$=0 for $s = \{0, 1, 2\}$. When $s = 4$, $Y_{c,s,g}$=0 for $s = \{0, 1, 2, 3\}$. Constraint (3.4) forces the gap counter to reset when a look occurs. We choose n as small as possible to satisfy the left hand side (LHS) of constraint (3.4) when $J_{c,s} = 0$ which correlates directly with $n = S$. Constraint (3.5) can be changed to require 2 or more looks per grid cell and sets up tracking of unobserved cells through the Z_c variable. Constraint (3.6) uses the weight derived from each resolution to maintain the budget of looks for each sensor. Constraint (3.7) allows us to require a minimum resolution for looks greater than or equal to δ. If $\delta = 1$, then one look is permitted to happen with a resolution less than the minimum resolution specified for that cell; all other looks shall use a higher resolution. We can adjust this constraint to require a minimum resolution on all looks by setting $\delta = 0$. The remaining constraints (3.8) through (3.12) set various variables that are non-existent to zero, establish non-negativity, and establish upper and lower bounds for variables and data.

3.2 Data Collection

We seek to instantiate SOM with data from STORM. This requires gathering data from STORM and converting it into usable input into SOM. We break data collection down into three steps which entail STORM output, initial processing, and model input. We accomplish STORM output through the use of a virtual machine to run STORM, examine data files, and retrieve data with pre-built functions created by the J8 staff in R, a programming language designed for statistical analysis (R Core Team 2016). STORM output data is further processed in R, outside of the virtual machine environment, to obtain usable input for Pyomo. Pyomo is a language designed to model optimization and is based in Python (Bynum et al. 2021; Hart et al. 2011; Van Rossum and Drake 2009). The last step in data collection involves the transfer of data from R to Pyomo with follow on processing before we run the model.

3.2.1 STORM Output

Regarding STORM output, we examine and transfer various text files generated in STORM to usable input in R. STORM runs on the Linux operating system and its base language is C++. Therefore, we use a virtual machine which provides software access to pre-built packages such as Stormtracker, a library that allows us to parse STORM output data into usable form (Schrad 2022). We obtain two specific files: AOI region and orbital path. These files provide us with the AOI type, AOI name, sensor type, grid coordinates for AOI boundaries, entry and exit sensor grid coordinates, and sensor battlespace passage simulation time (Section 2.2.3). We produce both of these files prior to starting STORM's simulation, which provides significant reductions in run time and eliminates stochastic outputs of different runs in STORM.

Additionally, we obtain grid cell information and collection penalty inputs through various STORM data files. Battlespace information includes the center grid, width, depth, and cell size, which we obtain from the battlespace.dat file. We manually input this information into R to facilitate adjustment in expanding cells for total AOI coverage. We also manually input the 13 inputs relating to each AOIs collection penalty into a comma separated values (CSV) file.

3.2.2 Initial Data Processing

We read the input files into a separate instance of R, outside of the virtual machine, where we divide cells into individual grid points, determine which grid points are in which AOIs, and determine which grid points are in which swaths. We assume that if the center point of each grid cell lies in a particular AOI, then it is assigned to that AOI. We use this center point grid cell assumption not only with AOIs but also in the case of swaths. STORM manuals and data files provide little information on the exact details of how an individual grid cell is divided up or assigned so these assumptions are necessary to move forward with SOM. To create grid points, we subdivide a rectangular polygon into a coordinates matrix of individual latitude and longitude pairs using the Geosphere and Simple Features libraries in R (Pebesma 2018).

We determine which grid points are in which AOIs through creation of polygons and the contains function in the Simple Features library which checks for points within a shape boundary. Producing the AOI cells in this manner assumes that the AOIs do not overlap and no AOIs contain grid points that are in another AOI. Hence, no grid cells are split between multiple AOIs.

We build swath objects initially with Geosphere for great circle distance and border dimensions. We then turn the swath into a multi-polygon object to add left and right swaths through Simple Features. We use the multi-polygon object to check which grid points it contains and store it in a list. Using this information, we generate simple statistics such as how often a cell appears in all swaths, which cells do not appear in any swaths, and the average number of cells per swath.

From this data we determine that the Blue combatant is required to search 1,415 cells and the Red combatant is required to search 2,663 cells. Within the first 12 hours (34 swaths), the Blue combatant swaths cover all 1,415 cells whereas the Red combatant swaths leave 5.1% or 137 grid cells uncovered (Tables 3.2 and 3.3).

Table 3.2. Blue combatant time and swaths it takes to cover all 1,415 Red grid cells.

Scenario Hours	Swaths	Cells Not Covered	Uncovered Percentage
6	10	120	8.5%
8	12	98	6.9%
12	34	0	0%

Table 3.3. Red combatant time and swath's it takes to cover all 2,663 Blue grid cells.

Scenario Hours	Swaths	Cells Not Covered	Uncovered Percentage
6	12	331	12.4%
8	12	331	12.4%
12	36	137	5.1%
18	39	114	4.3%
24	46	114	4.3%
72 (3 days)	138	95	3.6%
144 (6 days)	284	71	2.7%
288 (12 days)	565	55	2.1%
480 (20 days)	942	39	1.5%

By graphically depicting Tables 3.2 and 3.3 over the 20 day period, we observe that the Blue combatant is never able to search all cells (Figure 3.1). The best cell coverage Blue is able to reach still leaves 1.5% (39 cells) not covered by any swath. When analyzing cell coordinate locations, we conclude that the gradual reduction in uncovered cells is a result of the continuous axial turn of the Earth combined with the daily repeat of satellite orbits slightly shifting satellites on each pass.

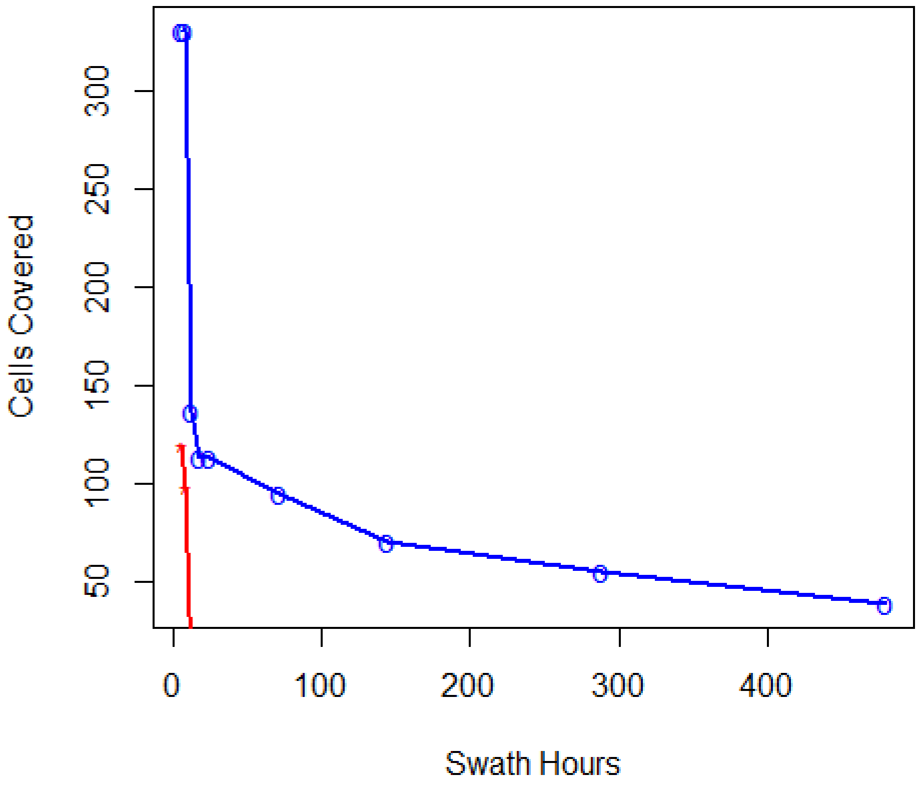


Figure 3.1. Swath hours in the Punic21 scenario and number of grid cells covered per combatant. The blue line represents the Blue grid cells and the red line represents the Red grid cells. The lines show the remaining grid cells that have not been covered compared to swath hours. We see Red's remaining grid cells reach and remain at 0 after 12 swath hours whereas Blue grid cells never reach 0.

We output three CSV files titled “AOI Consolidated,” “Swath Type,” and “CS List” to conclude the initial data processing. The “AOI Consolidated” file gives us the cell index position and type of AOI for each combatant. The “Swath Type” file gives us the swath indices, sensor type, and sim time for each combatant according to a user specified length of time to truncate the scenario. The “CS List” file gives us the grid cell index position for grid cells in each swath for each combatant.

3.2.3 Model Input Data Processing

For the next step in data processing, we operate in the Pyomo environment and read in the appropriate CSV files produced from R. We build a function to generate the collection penalty. This function takes the time since the previous look and 13 collection penalty inputs involving priority, revisit interval, worth, bias, and urgency and uses a rational bias function to determine 3 overall values labeled value, urgency, and priority (Section 2.3). The numerical values for value, urgency, and priority range between $[0, 1]$ and are then multiplied together to obtain the collection penalty which also ranges between $[0, 1]$.

AOIs that change search area type as the scenario progresses pose challenges for modelling (Section 2.2.2). This change in AOI type necessitates a change in collection penalty inputs and the type of sensor that is capable of looking in that geographic region. It additionally requires running the STORM scenario prior to running the model to find the time that these pre-determined conditions are met. Thus, the model assumes AOI types remain constant during the duration of the scenario. We examine two different cases of AOI types based on preconditions that are met at scenario start time that change the AOI type. For the cases we examine, the Blue combatant searching Red cells show all AOI types remaining constant. However, the Red combatant searching Blue cells have two AOIs, the "Red Anglo Republic" and the "Red North Carthage," that meet certain preconditions and both seamlessly shift from air defense (AD) to GMTI search areas. If the two AOIs do not change, we see an increase in cells from 2,663 to 5,565 since the AD AOI type is required to be searched by the same specific types of sensors.

A second function is used to inflate the collection penalty when it levels off. This function checks the previous collection penalty against the current collection penalty and, if equal, adds a minimal amount not to exceed 1. This assumption of increasing collection penalties over time is required in the model to appropriately track the gaps in constraints (3.2) and (3.3) and is thus a divergence from the STORM representation.

We use the "AOI Consolidated" file to separate grid cells into lists for each type of AOI. We build a dataframe with the appropriate swath number, time the swath occurred, and time difference for each gap. Any gap greater than the swath number is set to zero since those gaps do not exist. We calculate the collection penalty for each indexed swath and gap using the collection penalty function. We adjust these collection penalties using the inflation function and then add in grid cells for each specific dataframe of collection penalties when we transfer the dataframe over to a dictionary. The dictionary keys are cell, swath, and gap and the value is the collection penalty adjusted for inflation.

To calculate weight, we assign specific sensor resolution types the appropriate pre-calculated weight. We build a weight dictionary when we iterate through the "Swath Type" file, where swath and resolution represent the keys and the pre-calculated weight represents the values. We transfer the "CS List" into a two-dimensional array and assign a minimum resolution through a dictionary to each cell which concludes the data processing.

CHAPTER 4: Results

We are interested in both the computational and operational results that SOM produces. We first address the size of SOM since computational efficiency can often depend upon the number of variables and constraints for an optimization model. We conduct experiments that compare the performance of Computational Infrastructure for Operations Research branch and cut solver (CBC) and CPLEX algorithms and vary numbers of grid cells and swaths to see their impact on SOM runtime. We examine different options in CPLEX that reduce runtime and propose the best combination of options. Operational results focus on scenario grid cell coverage while adjusting the number of swaths, minimum resolution, and revisit intervals.

4.1 Computational Results

Because thousands of cells and hundreds of sensors produces millions of variables and constraints, it is prudent to have a firm grasp of SOMs size before discussing algorithms and computational efficiency. We consider the case with two different types of AOI grid cells and hence two different types of collection penalties. We set $C = 1,415$, $R = 5$, $S = 34$, $G = 34$, $\delta = 0$, $m = 100,000$, and $n = S$, with a minimum resolution of 4 for all grid cells. We use STORM's pre-loaded collection penalty parameters for the AOIs. These settings are representative of the initial Blue combatant scenario over 12 hours and we consider this the benchmark case. The experiments consist of this case unless otherwise annotated. We see that the variables and constraints for the benchmark case are approximately 2.03 million and 1.36 million respectively (Figure 4.1). The variables and constraints for the Blue combatant increase to 3.11 and 2.09 million for $S = 43$ and $G = 43$ and 11.45 million and 7.64 million for $S = 86$ and $G = 86$ (Figures 4.1 and 3.1). A comparison of the Blue and Red combatants shows that the variables and constraints more than double in size mainly due to the increase in grid cells from $C = 1,415$ to $C = 2,663$ (Figure 4.1).

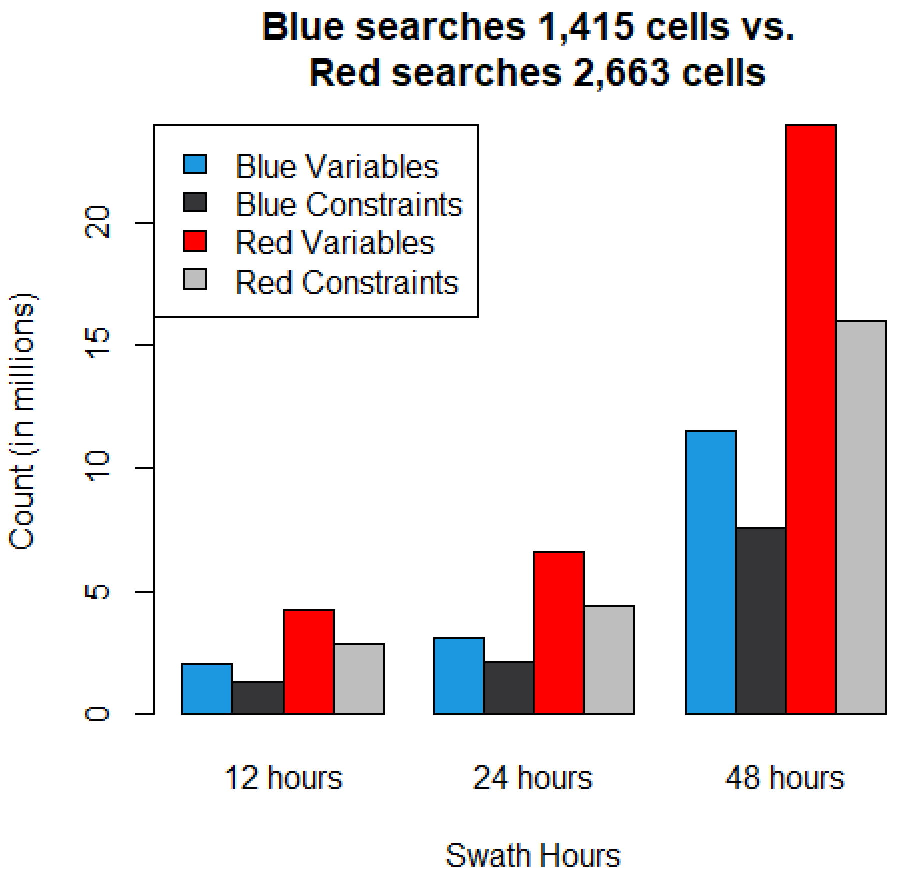


Figure 4.1. Blue and Red Combatant Variables and Constraints.

A direct comparison of the two different Red combatant scenarios with $C = 2,663$ versus $C = 5,565$ again shows the variables and constraints more than double in size (Figure 4.2). We initially combat the model size by setting as many specific variables to zero using constraints (3.8), (3.9), and (3.11).

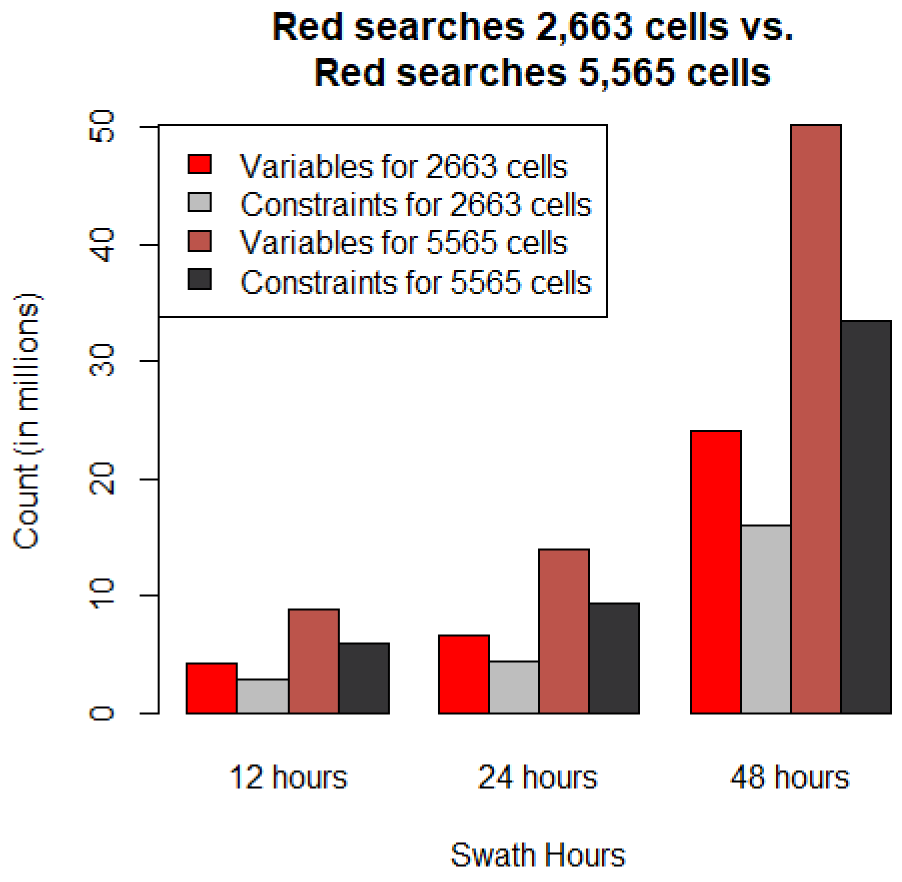


Figure 4.2. Red combatant variables and constraints for grid cells of 2,663 and grid cells of 5,565.

For the first set of experiments, we compare the performance of two algorithm's: CBC and CPLEX version 20.1.0.0 (Forrest and Lougee-Heimer 2005; CPLEX 2017). We solve on a laptop with 16 gigabytes (GB) random access memory (RAM) and a 2.69 gigahertz (GHz) processor running Windows 10 Pro. Table 4.1 shows that CPLEX outperforms CBC by a substantial amount in terms of runtime.

Table 4.1. Minutes of runtime for CBC and CPLEX with varying grid cells C using 10 swaths, and 10 gaps with minimum resolutions of 4 and 1 for higher and lower penalty grid cells respectively. We solve each scenario to optimality. If optimality could not be achieved, we report the runtime and, in brackets, when an optimality gap ≤ 5 was achieved.

C	CBC	CPLEX
25	0.1101	0.0502
50	0.1573	0.0624
75	0.9313	0.0770
100	4.1916	0.0868
125	2.3942 [0.012]	0.1070
150	0.4457 [0.0007]	0.1279
200	12.1968 [0.082]	0.6446

We see from Table 4.1 that CBC struggles to achieve optimality with $C \geq 125$. However, we notice that CBC achieves 0.012%, 0.0007%, and a 0.082% optimal gaps in a short amount of time for 125, 150, and 200 grid cells respectively. CBC plateaus at this optimality gap as each of these three experiments continue to run for more than 90 minutes. We manually end each of these experiments at 90 minutes. Regardless, in every instance CPLEX outperforms CBC. We conclude it is prudent to focus efforts on CPLEX and hence all remaining experiments use this algorithm. We fix grid cells to 100 and run experiments where we vary the number of swaths (Table 4.2).

Table 4.2. Minutes of runtime and optimality gap achieved for CPLEX by varying swaths S which in turn varies gaps G using 100 grid cells. We solve to a 5% optimality gap, and run all experiments on the 16 GB of RAM laptop.

S	Variables	Constraints	Runtime	Optimality Gap
16	39,201	24,638	0.2592	0%
18	47,601	29,885	0.5726	0%
42	210,801	117,899	57.4594	0.04%
46	249,201	139,303	218.1108	0.96%
65	475,301	259,547	190.3238	1.79%

We see a significant increase in runtime with $S \geq 42$ and compare these runtimes with experiments conducted on a desktop computer with 128 GB of RAM and a 2.10 GHz processor running Windows 10 Enterprise (Table 4.3).

Table 4.3. Minutes of runtime and optimality gap achieved for CPLEX by varying swaths S which in turn varies gaps G using 100 grid cells. We solve to a 5% optimality gap, and run all experiments on the 128 GB of RAM desktop.

S	Variables	Constraints	Runtime	Optimality Gap	GB
42	210,801	117,899	38.9580	0.51%	128
46	249,201	139,303	63.5254	2.0%	128
65	475,301	259,547	184.6071	3.7%	128

We see a decrease in runtime and optimality gap from the shift of 16 GB to 128 GB of RAM in rows 1 through 3 of Table 4.3 compared to rows 3 through 5 of Table 4.2. Since the runtime increases dramatically with $S \geq 42$, we examine this further with various combinations of grid cells and swaths in Table 4.4. We notice a trend of lack of progress in finding the best integer solution correlated with increasing runtimes when we examine individual CPLEX logs.

Table 4.4. Minutes of runtime and optimality gap achieved for CPLEX with varying grid cells and swaths. We solve all experiments to optimality on the 16 GB of RAM computer. Failed experiments did not produce any feasible solution within the runtime period and ran out of memory.

C	S	Variables	Constraints	Runtime	Optimality Gap
200	50	581,601	323,797	43.9361	0.01%
1,415	18	673,541	421,160	4.3940	0%
1,415	21	873,056	539,778	6.3790	0%
500	52	1,508,501	839,178	532.6857	failed
1,415	51	4,269,056	2,373,438	49.0897	0%

The lengthy runtimes in rows three, four, and five of Table 4.2 and one, four, and five of Table 4.4 have CPLEX logs that continue to increase in nodes without producing a feasible solution. For instance, row 1 of Table 4.4 and row 3 of Table 4.2 found initial best integer solutions at node 3,565 and 18,983 respectively. After producing a best integer solution, CPLEX is able to quickly work towards the optimal solution. To reduce runtime, we experiment with three different approaches as described in CPLEX (2017). We first see decreases in runtime by changing the node selection strategy from best bound search to best estimate search. The best estimate search selects the next node through a ratio of the previous node's progress toward a feasible integer solution compared with the objective function. Best bound search selects the next node in terms of the best objective function. Changing this parameter from best estimate to best bound search can be helpful in producing feasible solutions.

The second approach implements rounding of the collection penalty to the sixth decimal place. Since we are inflating each collection penalty by a minimal amount, each penalty has the potential to be 16 decimal places in length. The sixth decimal place is sufficient to permit a constantly increasing collection penalty and we expect this implementation to save on memory and potential computing time.

The third approach supplies CPLEX with the worst possible feasible solution to implement a warm start. We assign $Z_c = 1$, $J_{c,s} = s$, $X_{c,s,r} = 0$ for all c, s, and r. Additionally, we assign $Y_{c,s,g} = 1$ for all $s = g$ and 0 otherwise. The provided solution states that no looks occur over the entire time period. Providing a starting point via warm start supplies CPLEX with a 100% optimality gap to work from and may permit both quicker development in cutoff values for the best integer solution as well as enhance probing to fix variables (Klotz and Newman 2013). We label each of these options as CPLEX one, two, and three respectively and report the results in Table 4.5.

Table 4.5. Minutes of runtime and percentage optimality gap achieved in brackets with 10 swaths which in turn equate to 10 gaps with minimum resolutions of 4 and 1 for higher and lower penalty grid cells respectively. We solve to a 5% optimality gap using the 128 GB of RAM desktop.

S	Variables	Constraints	CPLEX	CPLEX 1	CPLEX 2	CPLEX 3	CPLEX 1, 2, 3
10	266,021	184,824	1.8810 [1.11]	1.8479 [0.03]	0.9781 [0.016]	0.6281 [0.01]	0.7065 [3.54]
22	945,221	585,761	96.1515 [1.60]	123.785 [0.18]	101.2721 [1.69]	1.8739 [0.03]	1.9362 [0.6]
34	2,031,941	1,183,678	1,466.37 [failed]	806.964 [2.91]	676.629 [0.55]	4.045 [1.52]	4.1544 [1.04]

We see that CPLEX option three, the warm start, provides a significant reduction in runtime whereas CPLEX options one and two improve upon the CPLEX default options with $S = 34$ (Table 4.5). Column eight shows that all three options combined do not significantly detract from CPLEX option three. We explore the impact of the other options combined with option three in Table 4.6.

Table 4.6. Minutes of runtime and percentage optimality gap achieved in brackets with varying swaths which in turn vary gaps. We use a minimum resolutions of 4 and 1 for higher and lower penalty grid cells respectively. We solve to a 5% optimality gap using the 128 GB of RAM desktop. All experiments in row three ran out of memory and were thus unable to achieve the 5% optimality gap.

S	Variables	Constraints	CPLEX 3	CPLEX 1 & 3	CPLEX 2 & 3	CPLEX 1, 2, 3
86	11,450,181	6,150,850	23.1269 [2.90]	23.3050 [2.90]	24.4981 [3.70]	24.6840 [3.70]
111	18,702,056	9,890,060	40.5063 [4.08]	40.8749 [4.08]	42.2248 [0.70]	42.1449 [4.08]
121	22,098,056	11,643,435	39.5528 [9.55]	39.0537 [9.55]	44.2908 [11.06]	49.6045 [11.06]

We notice a slight increase in runtime when we combine CPLEX option three with option two or options one and two. However, all instances in Table 4.6 were able to solve relatively quickly given the variable and constraint size and no CPLEX logs showed lengthy increases in nodes without a change in the optimality gap. We thus continue to explore these options and vary other parameters.

We set $rmin_c = 1$ for all C by removing constraint (3.7). When comparing row 1 with all other rows in Table 4.7, we see that removing this constraint increases the complexity of this problem significantly from an average of 4.2 minutes 966.2 minutes to solve to a 5% optimality gap.

Table 4.7. Minutes of runtime and percentage optimality gap achieved in brackets with varying minimum resolution $rmin_c$. We require all grid cells to use the specified minimum resolution except in $rmin_c = 5$. In that case, we set the higher collection penalty grid cells to 5 and the lower collection penalty grid cells to 4 to allow use of sensors that do not have NIIRS 5. We solve to a 5% optimality gap using the 128 GB of RAM desktop.

$rmin_c$	CPLEX 3	CPLEX 1 & 3	CPLEX 2 & 3	CPLEX 1, 2, 3
1	1,186.40 [0.92]	1,209.12 [0.7]	690.7936 [1.08]	778.67 [1.1]
2	4.1042 [0.01]	4.0752 [0.01]	4.4161 [0.01]	4.5694 [0.01]
3	4.2928 [0.88]	4.2189 [0.88]	4.2725 [0.88]	4.4734 [0.88]
4	3.9862 [0.08]	3.8098 [0.08]	4.0863 [0.08]	4.1340 [0.08]
5	4.1350 [0.08]	4.1969 [0.08]	4.1560 [0.08]	4.1546 [0.08]

When we compare the CPLEX logs from row one of Table 4.7, we notice similar progress across all columns. All 4 CPLEX logs show an initial drastic decrease in the optimality gap within the first 110 minutes to less than 29%. Significant trouble with lack of progress in reducing the optimality gap for the next 250-350 minutes follows until a repeat and restart of the solver occurs. The optimality gap slowly decreases to 6-7% within the next 300 minutes. Row 1, columns 2 and 3 both take over 400 minutes to achieve the less than 5% optimality gap threshold whereas row 1, columns 4 and 5 are able to achieve the threshold in significantly less time (Table 4.8).

Table 4.8. CPLEX log comparison for row 1 of Table 4.7. We report milestones (MS) which we consider significant decreases in the optimality gap. MS 2 is when the restart and repeat of the presolver occurred for all CPLEX options. We list the time in minutes and the optimality gap percentage in brackets.

CPLEX options	MS 1	MS 2	MS 3	MS 4
CPLEX 3	100.69 [26.62]	442.92 [26.62]	723.57 [6.02]	1,170.33 [1.23]
CPLEX 1 & 3	102.48 [26.62]	444.13 [26.62]	734.39 [6.02]	1,204.29 [0.7]
CPLEX 2 & 3	108.89 [28.51]	351.98 [25.89]	623.95 [7.81]	679.52 [1.08]
CPLEX 1, 2, 3	110.33 [28.51]	355.40 [25.89]	642.71 [6.70]	756.72 [1.1]

We suspect that by permitting more resolutions to be used for all grid cells causes this increase in computational complexity. Similarly, we can permit a single look for each grid cell to use any resolution by changing δ from 0 to 1 in constraint (3.7) instead of removing it entirely (Table 4.9).

Table 4.9. Minutes of runtime and percentage optimality gap achieved in brackets with varying number of sensors and using $\delta = 1$. We require all grid cells to use the specified minimum resolution $rmin_c = 2$ for all looks except 1. We solve to a 5% optimality gap using the 16 GB RAM laptop for row 2 and the 128GB RAM desktop for rows 1 and 3.

S	$rmin_c$	CPLEX 3	CPLEX 1 & 3	CPLEX 2 & 3	CPLEX 1, 2, 3
22	2	2.002 [0.91]	1.981 [0.91]	2.079 [0.91]	2.107 [0.91]
34	2	841.292 [4.56]	950.58 [1.1]	906.28 [1.54]	879.13 [0.89]
86	2	1,562.26 [50.21]	1,581.477 [50.21]	1,562.92 [49.13]	1,506.02 [49.13]

Table 4.9 increases our confidence in the complexity assumption due to increases in resolution. We see that as we increase from 22 to 34 swaths, the runtime increases sharply. We observe that CPLEX option one and three tends to slightly increase runtime; however, when

we bundle all three options together we see the lowest runtime. We believe incorporating CPLEX option one improves our chances to achieve both a low optimality gap and runtime when the problem is computationally expensive. We conclude that it is best to use all three options together to solve SAP.

4.2 Operational Results

We have several choices when it comes to parameters for setting up a scenario. We focus on two specific parameters, the time period (number of swaths) and the minimum resolution and explain both of them in the context of the benchmark scenario.

When we compare the results to STORM, we adjust appropriate STORM settings to prohibit attacks on and jamming of sensors. This eliminates some of the stochastic nature of STORM and allows us to appropriately compare the number of looks from SOM compared to STORM. We run the benchmark scenario but vary the number of swaths according to Figure 3.1. We immediately see increases in coverage for all swath hour intervals (Table 4.10 and Figure 4.3).

Table 4.10. Benchmark scenario coverage of SOM compared to STORM with varying the swath hours according to Table 3.1.

Swath hours	SOM looks	STORM looks	SOM percentage increase
12 hours	135	119	13.5%
24 hours	171	146	17.1%
48 hours	331	175	89.1%

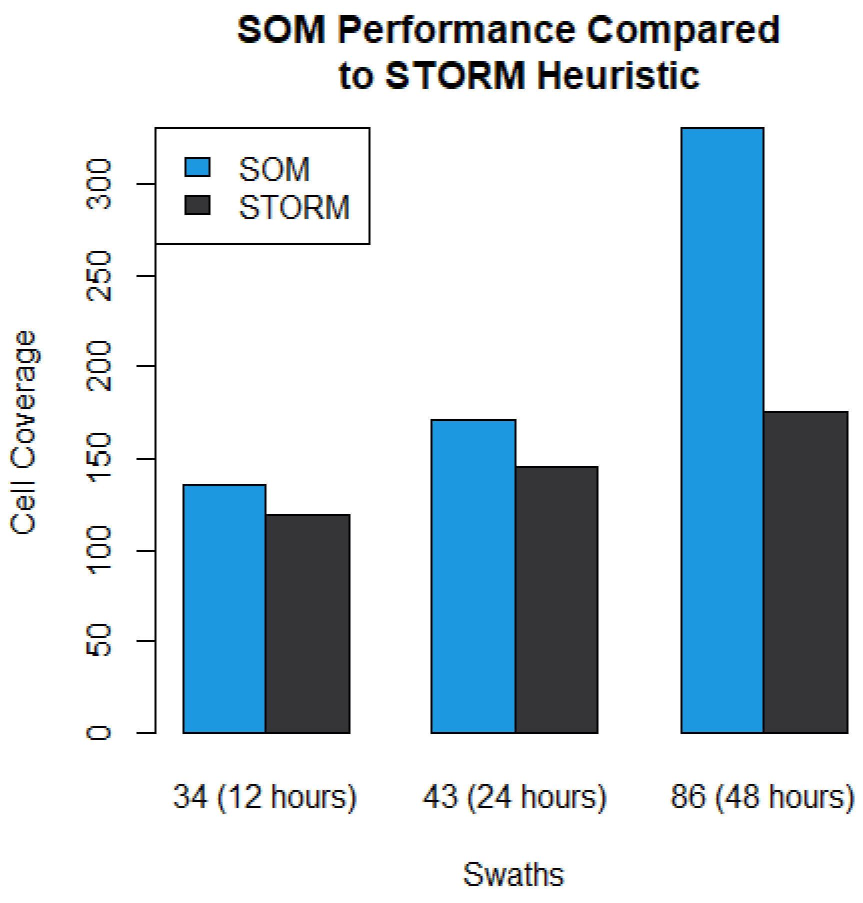


Figure 4.3. Benchmark SOM performance compared to STORM with varied swaths.

When we closely examine the impact of varying the number of swaths, we see that an initial low number of swaths results in similar coverage between SOM and STORM. SOM steadily increases coverage as swath's increase, more than doubling the coverage at 111 swaths compared to STORM (Table 4.11).

Table 4.11. SOM compared to STORM scenario coverage by AOI type with varied swaths. The total number of grid cells searched, if different than the unique number, are in brackets and the percentage of cells covered are listed in the coverage column.

	SOM			STORM		
s	AD	Naval	Coverage	AD	Naval	Coverage
10	24	16	2.83%	26	14	2.83%
22	0	83 [87]	5.87%	46 [64]	28	5.23%
34	0	135	9.54%	55 [76]	64	8.41%
43	65	106 [107]	12.08%	60 [85]	86 [87]	10.32%
86	100	231 [244]	23.39%	70 [164]	105 [171]	12.37%
111	0	409 [416]	28.90%	77 [228]	116 [227]	13.64%

When we relax the resolution for the larger grid cell Naval search AOI we see that SOM quickly outperforms STORM in coverage due to the increase in available budget. SOM shows 36% and 56% increases for 10 and 22 swaths respectively when compared to STORM (Table 4.12). STORM decreases the gap in coverage at 34 swaths to 30% and we notice STORM progressively reduce this deficit but never fully catch SOM by the 111 swath mark. Additionally, we see that SOM fails to search all cells as the budget increases. We suspect this may be due to solving to a 5% optimality gap.

Table 4.12. SOM compared to STORM coverage by AOI type with varied swaths and minimum resolution of 1 for the Naval search area. The total number of grid cells searched, if different than the unique number, are in brackets and the percentage of cells covered are listed in the coverage column.

	SOM			STORM		
s	AD	Naval	Coverage	AD	Naval	Coverage
10	2	802 [816]	56.82%	26	260 [300]	20.21%
22	18	1156 [1287]	82.97%	46 [64]	323 [386]	26.08%
34	48	1158 [1685]	85.23%	56 [78]	724 [1173]	55.12%
43	62	1158 [1914]	86.22%	62 [88]	848 [1549]	64.31%
86	135 [136]	1158 [3560]	91.38%	68 [160]	1102 [3630]	82.68%
111	192	1158 [4370]	95.41%	82 [228]	1114 [4802]	84.53%

Tables 4.11 and 4.12 show the difference in changing the minimum resolution compared with different swaths. However, we have not maximized STORM's heuristic performance for searching as many grid cells as possible. Two specific parameters, the required revisit interval and the desired revisit interval, play a significant role in when STORM chooses to revisit a grid cell. The desired revisit interval refers to the best possible timeliness for a look to occur whereas the required revisit interval refers to the time a look becomes useless. The required revisit interval is often associated with the maximum acceptable latency. Both revisit intervals are inputs into the penalty collection and have various effects on the penalty collection curve (Figures 2.7 and 2.8). The required revisit interval changes where the penalty collection curve flattens out to a slope of zero. For instance, the blue high score curve has a required revisit interval of 38 in Figure 2.7. The desired revisit interval changes the slope of the curve as it approaches the required revisit interval. A low desired revisit interval results in a high slope whereas a high desired revisit interval results in a low slope for the penalty collection curve line segment prior to the required revisit interval value.

Increasing the required revisit interval in STORM increases the chances of searching more grid cells. Setting the required revisit interval to the length of the swath period we are optimizing over helps to ensure that STORM will rotate looks when allocating looks to a specific type of AOI. However, setting a high required revisit interval comes at a cost and must be tempered with realism. If we increase all required revisit intervals on all grid cells to the optimization period time, we will see a decrease in the priority of revisiting the higher priority grid cells. Grid cells that incur a higher collection penalty have a higher priority and should be revisited more often.

We experiment with increasing the low priority Naval search area AOI required and desired revisit intervals to the optimization period time denoted in Table 3.1. We additionally increase the high priority AD search area AOI to $\frac{1}{6}$(Naval Search Area) to maintain the revisit priorities (Table 4.13).

Table 4.13. SOM compared to STORM coverage by AOI type with varied swaths and adjusted revisit rates. Required and desired revisit rates for the Naval search area match the swath hours interval in Table 3.1 and for the AD Search area are $\frac{1}{6}$(Naval Search Area). We report the total number of grid cells searched, if different than the unique number, in brackets and we report the percentage of cells covered in the coverage column.

	SOM			STORM		
S	AD	Naval	Coverage	AD	Naval	Coverage
10	2	802 [816]	56.82%	26	260 [300]	20.21%
22	17	1157 [1314]	82.97%	46 [64]	323 [386]	26.08%
34	48	1158 [1685]	85.23%	56 [78]	724 [1173]	55.12%
43	62	1157 [1908]	86.15%	62 [88]	993 [1522]	74.56%
86	136	1158 [3559]	91.45%	71 [164]	1209 [3470]	87.31%
111	194	1158 [4083]	95.54%	87 [222]	1215 [4897]	88.45%

We see that SOM is quite robust to revisit rate changes over all experimented swaths whereas STORM shows significant increases in coverage in the Naval search area AOI with $S \geq 43$. However, SOM still outperforms STORM's coverage in every instance. The robustness in revisit rate changes displayed in SOM meets our expectations since SOM is constrained to look at all grid cells first.

Of note, rows 7 and 8 of column 7 in Table 4.13 have more unique grid cells in the Naval search area than the 1,158. We speculate this is due to STORM determining grid cells in AOIs and swaths by a more complicated area method than the center point method we use in section 3.2.2. For coverage in these cases, we add the difference to the total grid cells to compute STORM coverage in a conservative manner.

Another option we adjust is the minimum resolution setting. We set blanket minimum resolutions for all grid cells and compare the coverage in Table 4.14.

Table 4.14. SOM compared to STORM coverage by AOI type with varied minimum resolution imposed on both search areas. For minimum resolution of 5, the AD search area is assigned 5 and the Naval search area is assigned 4 to prevent loss of sensor usage that do not possess NIIRS 5. We report the total number of grid cells searched, if different than the unique number, in brackets and we report the percentage of cells covered in the coverage column.

	SOM			STORM		
$rmin_c$	AD	Naval	Coverage	AD	Naval	Coverage
1	257 [753]	1153 [2130]	99.64%	262 [697]	1171 [2136]	100%
2	257	680 [725]	66.22%	227 [364]	583 [616]	57.24%
3	11	495 [508]	35.76%	157 [241]	278 [279]	30.74%
4	0	135	9.54%	55 [76]	64	8.41%
5	0	136	9.61%	21 [30]	80	7.14%

We see once again the coverage for STORM in row one, columns five and six exceed the expected grid cell count from SOM. We again attribute this to the center point method. STORM outperforms SOM for a resolution setting of NIIRS 1 by 0.36% and we suspect this is due to solving to a 5% optimality gap. The remaining minimum resolutions all outperform STORM in coverage. As expected in row five, SOM chooses all Naval search area grid cells using NIIRS four in order to maximize the budget and achieve the most amount of looks.

Since we have examined blanket performance regarding SOM compared to STORM, we transition to a more nuanced approach examining individual scenarios. For this approach, we first determine the number of cells in each type of AOI. We can calculate the required number of swaths to allocate one look to each cell by multiplying the grid cell count by the resolution weight (Table 4.15). Since we want to use a higher resolution on the AD search area, we do not bother creating combinations with the AD search area having a resolution less than the Naval search area.

Table 4.15. The number of cells is multiplied by the resolution weight from Figure 2.5 to obtain the required number of swaths.

AD Search Area (257 Grid Cells)	Naval Search Area (1158 Grid Cells)	Required Swaths
RMIN 1	RMIN 1	14.15
RMIN 3	RMIN 1	24.43
RMIN 3	RMIN 2	41.8
RMIN 4	RMIN 1	75.83
RMIN 4	RMIN 2	93.2

We then determine the percentage of cells covered after specific numbers of swaths (Tables 3.2 and 3.3). This lets us choose a point in time when we obtain a satisfactory amount of coverage. Next, we pick the time to optimize over based on the percentage of cells covered, desired number of swaths, and the maximum acceptable time for the required revisit interval. This chosen time will be used to set the maximum acceptable latency for the lowest priority AOI. Last, we pick minimum resolutions that fit within the budget for the total number of swaths based on Table 4.15.

For example, in the Blue combatant searching Red grid cells scenario, we see that minimum resolution levels of NIIRS 3 and NIIRS 1 will require 24.43 swaths to cover 257 and 1,158 grid cells respectively (Table 4.15). We achieve full grid cell coverage at 12 scenario hours with 34 swaths (Table 3.2). We can thus keep the resolution of NIIRS 3 and NIIRS 1 when choosing an optimization period of 12 hours. These resolutions will keep us under the 34 swath budget while providing the ability for full grid cell coverage. We set these parameters and compare them to STORM in Table 4.16.

Table 4.16. Benchmark scenario of SOM compared to STORM coverage with minimum resolution of 3 for AD search area grid cells and 1 for Naval search area grid cells. We report the total number of grid cells searched, if different than the unique number, in brackets and the percentage of cells covered in the coverage column.

Model	Search Area	NIIRS 1	NIIRS 2	NIIRS 3	NIIRS 4	Total	Coverage
STORM	AD	0	0	142 [218]	15 [20]	157 [238]	61.1%
STORM	Naval	934 [1460]	0	0	0	934 [1460]	80.6%
STORM	Total	934 [1460]	0	142 [218]	15 [20]	1091 [1698]	77.1%
SOM	AD	0	0	239	4	243	94.6%
SOM	Naval	1157 [1624]	1 [8]	0 [1]	0	1158 [1633]	100%
SOM	Total	1157 [1624]	1 [8]	239 [240]	4	1401 [1876]	99.0%

We observe with STORM that we achieve only 77.1% grid cell coverage but with SOM we achieve 99.0% total grid cell coverage and 100% coverage in the Naval Search Area. Since STORM uses NIIRS four for a portion of coverage for the AD Search Area, we suspect the greedy algorithm fails to balance looks over the entire time interval when sensors with NIIRS three are unavailable. SOM clearly outperforms STORM in this instance by achieving 21.9% better grid cell coverage.

A potential way to increase coverage in STORM using the greedy heuristic would be to set all maximum latency periods to the same time interval that we optimize over. Setting equal time intervals across all types of grid cells assists us in keeping higher priority grid cells from overtaking lower priority grid cells in collection penalty after the higher priority grid cell receives a look. The tradeoff to making this setting change is that STORM will tend to rotate looks and not allocate more extra looks towards higher priority collection penalty grid cells. Using this setting, STORM will continue to alternate looks. Thus, once all grid cells receive a look, the higher priority grid cells will not be prioritized. Additionally, this is an unrealistic setting from a COCOM planner's point of view since higher priority grid cells will need to be revisited more often than lower priority grid cells. While this scenario is not ideal in STORM from a planner's perspective, we set these parameters in both SOM and STORM and examine the comparison in Table 4.17.

Table 4.17. SOM compared to STORM coverage with adjusted revisit intervals all set to 12 hours and minimum resolutions of 3 and 1 for the AD and Naval search areas respectively. We report the total number of grid cells searched, if different than the unique number, in brackets and the percentage of cells covered in the coverage column.

Model	Search Area	NIIRS 1	NIIRS 2	NIIRS 3	NIIRS 4	Total	Coverage
STORM	AD	0	0	182 [196]	19 [20]	201 [216]	78.2%
STORM	Naval	1010 [1614]	0	0	0	1010 [1614]	87.2%
STORM	Total	1010 [1614]	0	182 [196]	19 [20]	1211 [1830]	85.6%
SOM	AD	0	0	239	3	242	94.2%
SOM	Naval	1156 [1640]	1 [8]	0 [1]	1	1158 [1650]	100%
SOM	Total	1156 [1640]	1 [8]	239 [240]	4	1400 [1892]	98.9%

We see an increase in STORM coverage by 8.5% but we still see SOM outperform STORM by 13.3%. Additionally, we see a loss in a single look from Table 4.16 to Table 4.17 and we suspect it is due to solving to a 12.4% optimality gap due to experiment time constraints. Thus, a change in revisit interval does not appear to have a significant impact on SOM performance for the initial set of unique looks which aligns with the assumptions from Table 4.13.

We create the next operational scenario reverting back to realistic revisit intervals that span the entire optimization time period. We see from Tables 4.15 and 3.1 that 43 swaths over a 24 hour period will provide us a large enough look budget using minimum resolutions of NIIRS 3 and NIIRS 2 for AD and Naval search areas respectively and we display the results in Table 4.18.

Table 4.18. SOM compared to STORM coverage with adjusted revisit intervals of 4 and 24 hours and minimum resolutions of NIIRS 3 and NIIRS 2 for the AD and Naval search areas respectively. We report the total number of grid cells searched, if different than the unique number, in brackets and the percentage of cells covered in the coverage column.

Model	Search Area	NIIRS 2	NIIRS 3	NIIRS 4	Total	Coverage
STORM	AD	0	140 [213]	14 [19]	154 [232]	59.92%
STORM	Naval	504 [523]	0	13 [17]	517 [540]	44.64%
STORM	Total	504 [523]	140 [213]	27 [36]	671 [772]	47.42%
SOM	AD	0	35	20	55	21.40%
SOM	Naval	996 [1077]	0	20 [24]	1016 [1101]	87.74%
SOM	Total	996 [1077]	35	40 [44]	1071 [1156]	75.69%

We see in this case that SOM continues to outperform STORM in coverage but SOM is 24.31% away from entire scenario grid cell coverage. We suspect this is due to sensors that are able to only use certain resolutions. Because 25% of the total swath sensors do not possess the capability of using NIIRS 2 or 3 to look at grid cells, both SOM and STORM are forced to use NIIRS 4 which reduces the budget of looks significantly.

We run an experiment on another instance from Table 4.15 using resolution levels NIIRS 4 and NIIRS 1 which require 76 swaths and display the results in Table 4.19.

Table 4.19. SOM compared to STORM coverage with adjusted revisit intervals set to 8 and 48 hours and minimum resolutions of 4 and 1 for the AD and Naval search areas respectively. We report the total number of grid cells searched, if different than the unique number, in brackets and the percentage of cells covered in the coverage column.

Model	Search Area	NIIRS 1	NIIRS 4	Total	Coverage
STORM	AD	0	56 [78]	56[78]	21.79%
STORM	Naval	858 [1163]	0	868 [1163]	74.09%
STORM	Total	858 [1163]	56 [78]	914 [1241]	64.59%
SOM	AD	0	140	140	54.47%
SOM	Naval	1158 [3486]	0	1158 [3486]	100%
SOM	Total	1158 [3486]	140	1298 [3626]	91.73%

We again see SOM outperform STORM but SOM fails to achieve 100% coverage with all extra looks allocated to the Naval search area using NIIRS 1. We suspect this is because SOM reduces the objective function value with many looks allocated to a lower priority search area than few looks allocated to a higher priority search area.

These direct comparisons between SOM and STORM aid us in evaluating performance. However, the major benefits of SOM are the options it provides compared to STORM. One option is the ability to view grid cells with a low resolution for a single look and then require a higher resolution for subsequent looks. We see that this option quickly gives us 100% coverage with a coarse resolution and then concentrate subsequent looks on higher resolutions (Table 4.20).

Table 4.20. SOM compared to STORM coverage with minimum resolution of 2 for all grid cells and SOM setting $\delta = 1$. We report the total number of grid cells searched, if different than the unique number, in brackets and the percentage of cells covered in the coverage column.

Model	Search Area	NIIRS 1	NIIRS 2	NIIRS 3	NIIRS 4	NIIRS 5	Total	Coverage
STORM	AD	0	211 [344]	0	16 [20]	0	227 [364]	88.32%
STORM	Naval	0	571 [600]	0	12 [16]	0	583 [616]	50.35%
STORM	Total	0	782 [944]	0	28 [36]	0	810 [980]	57.24%
SOM	AD	31 [257]	226 [324]	0	0	0	257 [581]	100%
SOM	Naval	1067 [1158]	88 [308]	3 [11]	0 [5]	0 [1]	1158 [1483]	100%
SOM	Total	1098 [1415]	314 [632]	3 [11]	0 [5]	0 [1]	1415 [2064]	100%

When we compare this option with STORM we use the case where the minimum resolutions match. This is the most closely aligned case and we see that SOM easily outperforms STORM in terms of coverage by over 42%. STORM does receive several more high quality looks compared to SOM but we can easily help compensate for this by imposing a higher minimum resolution and keeping $\delta = 1$.

When we compare the 28 operational case studies conducted, we see that SOM provides an average of 54.6% and a median of 22.8% more coverage compared to STORM (Figure 4.4).

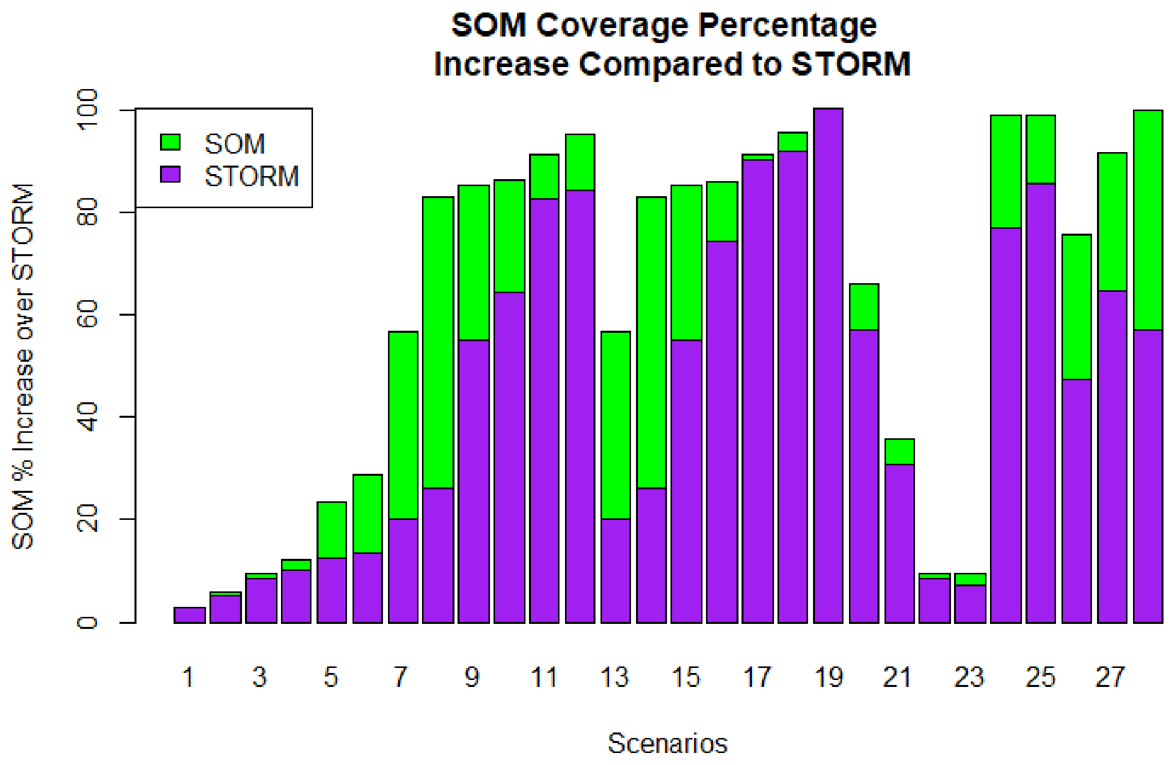


Figure 4.4. Summary of each operational case study illustrating SOM's increase in coverage compared to STORM.

We conclude that SOM provides the best options for searching all grid cells and provides extra, tailorable options that STORM does not possess.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5: Conclusion

In this thesis, we develop SOM, which is a MILP that seeks to distribute a limited number of looks from sensors to grid cells in the best possible way. We refer to this problem as SAP which directly applies to a specific problem encountered in STORM, a campaign analysis tool used by the J8 directorate in the Pentagon. Since STORM uses a heuristic to distribute grid cell looks, we additionally seek to outperform this heuristic. SOM accomplishes both of these objectives by directing where sensors look to ensure all grid cells receive a look and avoid large revisit gaps. Results show that by introducing several CPLEX options including a warm start, we can reduce runtime from over 1,400 minutes with no feasible solution to less than 5 minutes with a feasible solution on a benchmark campaign scenario. The additional CPLEX options allow us to increase the size of the problem to over 22 million variables and 11 million constraints and achieve an 11% optimality gap in less than 50 minutes. Operational case study results show that SOM provides an average of 54.6% and a median of 22.8% more coverage compared to STORM. Additional options present in SOM and absent in STORM ensure that SOM will outperform STORM to quickly reach maximum coverage and subsequently focus on allocating looks to the most important grid cells.

Drawbacks to SOM are that slight changes in δ result in potentially lengthy runtimes. This change in δ also results in the inability to reach an acceptable optimality gap when variables and constraints number above 11 million and 6 million respectively. Another computational drawback is having to accept an optimality gap between 5 and 12% to avoid out of memory errors on a single processor as variables and constraints increase above 22 million and 11 million respectively. Operational drawbacks are that the center point grid cell method does not completely match STORM's area method for determining the number of grid cells in AOIs and swaths.

Further computational research can be directed towards runtime reduction and expanding SOM's options. Implementation of a rolling horizon approach as well as constraint programming may prove useful with further testing for reductions in both runtime and required memory. Incorporation of more sophisticated options into SOM such as a prioritized, progressive, contingent based AOI search, increased penalties commensurate with required higher resolutions, or integration of STORM aircraft sensors and targets are all worthy of further exploration but require additional constraints, data collection, and model programming.

Despite drawbacks, SOM outperforms STORM's heuristic in all cases studied except one. The additional option of allowing no minimum resolution for a user defined number of looks ensures that SOM will outperform STORM to quickly reach maximum coverage and subsequently focus on allocating looks to the most important grid cells.

List of References

Abi-Zeid I, Morin M, Nilo O (2019) Decision support for planning maritime search and rescue operations in canada. *ICEIS (1)*, 328–339.

Archetti C, Feillet D, Hertz A, Speranza MG (2009) The capacitated team orienteering and profitable tour problems. *The Journal of the Operational Research Society* 60(6):831–842.

Baldomero-Naranjo M, Kalcsics J, Rodriguez-Chia AM (2021) Minmax regret maximal covering location problems with edge demands. *Computers Operations Research* 130:105–181.

Berger J, Barkaoui M, Lo N (2021) Near-optimal search-and-rescue path planning for a moving target. *Journal of the Operational Research Society* 72(3):688–700, http://dx.doi.org/10.1080/01605682.2019.1685362.

Bourque FA (2019) Solving the moving target search problem using indistinguishable searchers. *European Journal of Operational Research* 275(1):45–52, http://dx.doi.org/https://doi.org/10.1016/j.ejor.2018.11.006.

Bynum ML, Hackebeil GA, Hart WE, Laird CD, Nicholson BL, Siirola JD, Watson JP, Woodruff DL (2021) *Pyomo–Optimization Modeling in Python*, volume 67 (Springer Science & Business Media), third edition.

Church R, ReVelle C (1974) The maximal covering location problem. *Papers in Regional Science* 32(1):101–118.

Cordeau JF, Furini F, Ljubic I (2019) Benders decomposition for very large scale partial set covering and maximal covering location problems. *European Journal of Operational Research* 275(3):882–896.

CPLEX II (2017) V12.8: Ibm ilog cplex optimization studio cplex user's manual. *International Business Machines Corporation* 596.

Dell RF, Eagle JN, Alves Martins GH, Garnier Santos A (1996) Using multiple searchers in constrained-path, moving-target search problems. *Naval Research Logistics* 43(4):463–480.

Ding H (2018) Models and algorithms for multi-agent search problems. Https://search.ebscohost.com/login.aspx?direct=truedb=dduAN=BA8F7E4C1D1C53D8site=ehost-live.

Earth Imaging Journal (2015) Esa's sentinel-2a readies for launch (March 17).

Fomin FV, Lingas A (2002) Approximation algorithms for time-dependent orienteering. *Information Processing Letters* 83(2):57–62.

Forrest J, Lougee-Heimer R (2005) *CBC User Guide*. *INFORMS* 257.

Geem ZW, Tseng CL, Park Y (2005) Harmony search for generalized orienteering problem: Best touring in china. *Advances in Natural Computation*, 741–750, Lecture Notes in Computer Science (Berlin, Heidelberg: Springer Berlin Heidelberg).

Gendreau M, Laporte G, Semet F (1998) A tabu search heuristic for the undirected selective travelling salesman problem. *European Journal of Operational Research* 106(2):539–545.

Group W (2019a) *STORM Analyst's Manual Version 2.9*. Vienna, VA 22180.

Group W (2019b) *STORM Appendices Version 2.9*. Vienna, VA 22180.

Group W (2019c) *STORM Programmer's Manual Version 2.9*. Vienna, VA 22180.

Group W (2019d) *STORM User's Manual Version 2.9*. Vienna, VA 22180.

Group W (2005) Synthetic theater operations research model, version 2.10.0.3. Accessed November, 2021, Vienna, VA 22180.

Gunawan A, Lau HC, Vansteenwegen P (2016) Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research* 255(2):315–332.

Hart WE, Watson JP, Woodruff DL (2011) Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation* 3(3):219–260.

Hoogendoorn YN (2017) The maximum coverage problem.

Ilhan T, Iravani SMR, Daskin MS (2008) The orienteering problem with stochastic profits. *IIE Transactions* 40(4):406–421.

Joint Chiefs of Staff (2022) J8 force structure, resources, and assessment directorate. Accessed April 17, 2022, https://www.jcs.mil/Directorates/J8-Force-Structure-Resources-Assessment/.

Kim H, Kim BI, Noh D (2020) The multi-profit orienteering problem. *Computers & Industrial Engineering* 149:106808, http://dx.doi.org/https://doi.org/10.1016/j.cie.2020.106808.

Klotz E, Newman AM (2013) Practical guidelines for solving difficult mixed integer linear programs. *Surveys in Operations Research and Management Science* 18(1):18–32.

Kress M, Royset J, Rozen N (2012) The eye and the fist: Optimizing search and interdiction. *European Journal of Operational Research* 220(2):550–558.

Lau H, Huang S, Dissanayake G (2008) Discounted mean bound for the optimal searcher path problem with non-uniform travel times. *European Journal of Operational Research* 190(2):383–397.

Pebesma E (2018) Simple Features for R: Standardized Support for Spatial Vector Data. *The R Journal* 10(1):439–446, http://dx.doi.org/10.32614/RJ-2018-009.

Peng G, Song G, Xing L, Gunawan A, Vansteenwegen P (2020) An exact algorithm for agile earth observation satellite scheduling with time-dependent profits. *Computers Operations Research* 120:104946–.

Pereira MA, Coelho LC, Lorena LA, de Souza LC (2015) A hybrid method for the probabilistic maximal covering location–allocation problem. *Computers Operations Research* 57:51–59.

Pietz J, Royset JO (2013) Generalized orienteering problem with resource dependent rewards. *Naval Research Logistics* 60(4):294–312.

Price AN, Curtin KM, Magliocca NR, Turner D, Mitchell P, McSweeney K, Summers DL (2022) A family of models in support of realistic drug interdiction location decision-making. *Transactions in GIS* n/a(n/a).

R Core Team (2016) *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, https://www.R-project.org/.

Rardin R (2016) *Optimization in Operations Research* (Pearson Education), https://books.google.com/books?id=erV5CwAAQBAJ.

Royset JO, Sato H (2010) Route optimization for multiple searchers. *Naval Research Logistics* 57(8):701–717.

Sato H, Royset JO (2010) Path optimization for the resource-constrained searcher. *Naval Research Logistics* 57(5):422–440.

Schrad J (2022) Stormtracker: A package for analyzing storm. Stormtracker. Accessed April 2022, https://gitlab.gs.mil/j8-wad/stormtracker.

Sekar Amarilies H, Perwira Redi AAN, Mufidah I, Nadlifatin R (2020) Greedy heuristics for the maximum covering location problem: A case study of optimal trashcan location in kampung cipare - tenjo - west java. *IOP conference series. Materials Science and Engineering* 847(1).

Seyhan TH, Snyder LV, Zhang Y (2018) A new heuristic formulation for a competitive maximal covering location problem. *Transportation Science* 52(5):1156–1173.

Song NO, Teneketzis D (2004) Discrete search with multiple sensors. *Mathematical Methods of Operations Research (Heidelberg, Germany)* 60(1):1–13.

Stone LD, Royset JO, Washburn AR (2016) *Optimal search for moving targets*. International series in operations research management science, volume 237 (Cham: Springer).

Tsiligirides T (1984) Heuristic methods applied to orienteering. *The Journal of the Operational Research Society* 35(9):797–809.

Van Rossum G, Drake FL (2009) *Python 3 Reference Manual* (Scotts Valley, CA: CreateSpace).

Vansteenwegen P, Souffriau W, Oudheusden DV (2011) The orienteering problem: A survey. *European Journal of Operational Research* 209(1):1–10.

Vatsa AK, Jayaswal S (2021) Capacitated multi-period maximal covering location problem with server uncertainty. *European Journal of Operational Research* 289(3):1107–1126.

Wang Q, Sun X, Golden B, Jia J (1995) Using artificial neural networks to solve the orienteering problem. *Annals of Operations Research* 61(1):111–120.

Wang X, Golden BL, Wasil EA (2008) Using a genetic algorithm to solve the generalized orienteering problem. *The Vehicle Routing Problem: Latest Advances and New Challenges*, 263–274, Operations Research/Computer Science Interfaces (Boston, MA: Springer US).

Washburn AR (1998) Branch and bound methods for a search problem. *Naval Research Logistics* 45(3):243–257.

Yoash RB, Atkinson MP, Kress M (2018) Where to dip? search pattern for an antisubmarine helicopter using a dipping sensor .

Zarandi MHF, Davari S, Sisakht SAH (2013) The large-scale dynamic maximal covering location problem. *Mathematical and Computer Modelling* 57(3-4):710–719.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California