OPTIMIZING CYBER THREAT INTELLIGENCE: A MULTI-OBJECTIVE APPROACH TO FEED SELECTION

BY

MEREDITH MORRISON

A Thesis Submitted to the Graduate Faculty of

WAKE FOREST UNIVERSITY GRADUATE SCHOOL OF ARTS AND SCIENCES

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIENCE

Computer Science

May, 2026

Winston-Salem, North Carolina

Approved By:

Errin W. Fulp, Ph.D., Advisor

Daniel A. Cañas, Ph.D., Chair

William H. Turkett, Ph.D.

Acknowledgments

I would like to thank my support system and mentors who have guided me throughout my academic career and during this endeavor. I am deeply grateful to have people in my life who have listened to me, encouraged me, and believed in me every step of the way.

I am grateful to have spent both my undergraduate and graduate studies at Wake Forest University, where I have had the privilege of learning from some truly amazing professors. To my advisor, Dr. Fulp, thank you for patiently listening to all my random ideas and for your advice and fun stories throughout the process. Dr. Cañas, thank you for your mentorship and friendship, for always being available for a conversation, and for believing in me since my undergraduate years. Dr. Turkett, thank you for your time and support throughout the thesis process. Dr. Santago, it has been a privilege to work as your Teaching Assistant, and I appreciate your support, patience, and engaging stories during lab. Dr. Ballard, thank you for challenging me early on in my graduate career to push me to work harder and learn more than I thought I could and for providing me with the opportunity to publish my first paper. I am incredibly grateful to the entire Wake Forest Computer Science faculty, and I will miss being part of this community.

Finally, I would like to thank my friends and family, without whom this thesis would not have been possible. To my parents, thank you for being my biggest supporters from the very beginning and for teaching me what it means to be a hard worker. I would not be where I am today without your constant belief in me, your hopes for my future, and the sacrifices you have made along the way. To my siblings, Ryan and Abby, thank you for seeing a better and smarter version of me beyond what I see in myself. I will always strive to become a person worthy to be your older sister. Finally, to my fiancé, Christopher, thank you for your constant support, patience, and encouragement throughout this journey. I feel so grateful to know your love, joy, creativity, and kindness, and I truly cannot imagine my life without you in it.

Table of Contents

List of Figures ..... **v**

List of Tables ..... **vi**

Abstract ..... **viii**

Chapter 1 Introduction **1**
1.1 Motivation 1
1.2 Problem Statement 3
1.3 Research Objectives 5
1.4 Thesis Structure 7

Chapter 2 Background **8**
2.1 Introduction to Cyber Threat Intelligence 8
2.1.1 Threat Intelligence and IoCs 9
2.1.2 Security Operations 10
2.1.3 Big Data Problem 11
2.2 Related Works 12

Chapter 3 Mathematical Formulation **15**
3.1 Definition of Set Cover Problem 15
3.2 Multi-Objective Optimization 16
3.3 NP-hard Proof 19

Chapter 4 Algorithms **20**
4.1 Greedy Heuristics 20
4.1.1 Greedy Set Cover 22
4.1.2 Ratio Greedy 24
4.1.3 Additive-Penalty Greedy 27
4.1.4 Review of Greedy Heuristics 30
4.2 Mathematical Programming Approaches 31
4.2.1 Integer Linear Programming 32
4.2.2 Linear Programming Guided Heuristics 37
4.3 Evolutionary Algorithms 43
4.3.1 Non-dominated Sorting Genetic Algorithm II 44
4.4 Summary of Algorithms 48

Chapter 5 Methodology **51**
5.1 Experimental Overview 51
5.2 Evaluation Metrics 52
5.3 Problem Space Features 54
5.4 Description of Problem Instances 58
5.4.1 Baseline Evaluation and Scalability 58
5.4.2 Specialized Feed Compositions 59
5.5 Classification Models 62
5.6 Chapter Overview 63

Chapter 6 Results and Analysis **65**
6.1 Empirical Analysis 65
6.1.1 Evaluation on Randomized Problems 65
6.1.2 Evaluation on Specialized Problems 72
6.2 Problem Space Analysis 77
6.2.1 Greedy Set Cover 77
6.2.2 Ratio Greedy 80
6.2.3 Additive-Penalty Greedy 83
6.2.4 Integer Linear Programming 85
6.2.5 LP-Guided Heuristic 87
6.2.6 NSGA-II 89
6.3 Overall Comparison 92

Chapter 7 Conclusion and Future Work **94**
7.1 Summary of Contributions 94
7.2 Limitations 96
7.3 Future Work 97
7.4 Final Remarks 98

Bibliography **100**

Curriculum Vitae . . . . .

List of Figures

6.1 Greedy Set Cover: Feature Weights 78
6.2 Ratio Greedy: Feature Weights 81
6.3 Additive-Penalty Greedy: Feature Weights 84
6.4 Integer Linear Programming: Feature Weights 86
6.5 LP-Guided Heuristic: Feature Weights 88
6.6 NSGA-II: Feature Weights 91

List of Tables

6.1 Scalability Tiers for CTI Feeds and IoCs 66
6.2 Percentage of "Best" Solutions Discovered by Problem Size 67
6.3 ILP Full Recall Achievement by Problem Size 69
6.4 Suboptimality Severity: Excess Feeds Selected by Problem Size 70
6.5 Suboptimality Severity: Precision Degradation by Problem Size . . . 70
6.6 NSGA-II Average Frontier Size by Dataset Tier 71
6.7 Percentage of "Best" Solutions Discovered by Problem Characteristic 73
6.8 Suboptimality Severity: Excess Feeds Selected by Problem Characteristic 75
6.9 Suboptimality Severity: Precision Degradation by Problem Characteristic 76
6.10 NSGA-II Average Frontier Size by Dataset 76
6.11 XGBoost Feature Importance for Greedy Set Cover 78
6.12 XGBoost Feature Importance for Ratio Greedy Algorithm 81
6.13 XGBoost Feature Importance for Additive-Penalty Greedy Algorithm 83
6.14 XGBoost Feature Importance for Integer Linear Programming 86
6.15 XGBoost Feature Importance for LP-Guided Heuristic 88
6.16 XGBoost Feature Importance for NSGA-II 90
6.17 Summary of Algorithmic Performance and Deployment Scenarios . . 93

List of Algorithms

1 Greedy Set Cover 23
2 Ratio Greedy 26
3 Additive-Penalty Greedy 29
4 Integer Linear Programming 36
5 LP-Guided Heuristic 39
6 Solve LP Relaxation 40
7 Multi-Objective Feed Selection via NSGA-II 47

Abstract

In modern cybersecurity operations, threat intelligence feeds provide critical Indicators of Compromise (IoCs) to track malicious activity. However, organizations often ingest multiple overlapping feeds from various Cyber Threat Intelligence (CTI) providers, leading to redundant data, irrelevant noise, and increased false positives. This data overload severely strains both automated detection systems and human analysts. To mitigate this issue, this thesis introduces an NP-hard, multi-objective optimization challenge aimed at selecting a minimal subset of feeds that guarantees complete coverage of required IoCs while minimizing unneeded noise and feed ingestion costs.

Evaluating a range of solution strategies, this research introduces localized greedy heuristics and precision-aware variants, exact numerical optimization, and scalable evolutionary search to address the defined feed selection problem. Furthermore, a problem-space analysis using XGBoost predictive modeling helps identify the underlying structural characteristics that dictate algorithmic success. Ultimately, while exact solvers like Integer Linear Programming achieve unmatched mathematical optimality on small, constrained problems, evolutionary algorithms like NSGA-II provide the greatest overall robustness and scalability for navigating realistic problems of high redundancy and scale. Together, these insights offer actionable strategies to mitigate data overload and prioritize efficient, high-quality feed selection.

Chapter 1: Introduction

Cybersecurity operations rely heavily on timely, accurate intelligence to preemptively detect and mitigate malicious activity across networks and systems. One of the most common sources of this information is threat intelligence feeds. While this insight is critical to support proactive defense, managing and selecting an increasing number of feeds evolves into a massive operational challenge for modern organizations, which serves as the motivation behind this study.

This chapter establishes the foundation for this research. It first outlines the motivation behind the study, detailing the operational and financial strain of threat data overload. It then formally defines the explored problem and its scope, establishes the specific research objectives, and concludes by laying out the structural organization for the remainder of this thesis.

1.1 Motivation

In the field of cybersecurity, threat intelligence feeds operate as continuously updated streams of Indicators of Compromise (IoCs). Fundamentally, IoCs are observable artifacts that signal a potential intrusion or malicious activity on a network or system; common examples include malicious IP addresses, suspicious domain names, and file hashes associated with known malware. Using sensors, honeypots, malware analysis pipelines, and other sources, Cyber Threat Intelligence (CTI) providers accumulate these IoCs and subsequently organize them into CTI feeds.

These intelligence feeds are consumed by Security Operations Centers (SOCs) and integrated into automated detection and defense tools, such as firewalls, Security Information and Event Management (SIEM) systems, and endpoint detection plat-

forms. In this way, the streamed IoCs can be used for automated blocking, alerting, and correlation with other security signals. Leveraging the knowledge gained from relevant threat data plays a critical role in providing early warnings of emerging threats and ensuring that organizations can respond quickly to potential intrusions.

To broaden visibility and improve detection coverage, organizations often subscribe to multiple CTI providers and consume hundreds or even thousands of intelligence feeds. Each feed provides a specific perspective, often focusing on various threat groups, malware types, or industries. While this diversity improves threat recall, it also introduces significant operational challenges. To begin, many feeds contain overlapping data, resulting in redundant subscriptions where the exact same IoC is processed multiple times. This leads to direct financial costs, as organizations pay for unnecessary feeds, as well as technical costs, such as increased bandwidth usage and massive storage overhead for ingestion pipelines.

Furthermore, CTI feeds frequently contain irrelevant data, which can include outdated indicators, low-confidence signals, or outright false positives. This noise can quickly overwhelm both automated systems and human analysts, generating excessive false alarms. This inevitably leads to alert fatigue, where true threats are missed simply because they are buried beneath thousands of low-quality signals. In extreme cases, analysts may disable or ignore certain noisy feeds entirely, completely undermining the foundational purpose of purchasing threat intelligence.

Consequently, these operational and financial challenges manifest as a fundamental mathematical optimization problem. In this thesis, the goal is to discover how an organization can select the optimal subset of intelligence feeds that collectively guarantees full coverage of IoCs necessary for defense while simultaneously minimizing both the total number of feed subscriptions and the ingestion of noise.

In practice, navigating this decision carries incredibly high stakes for enterprise se-

curity. Selecting an insufficient number of feeds risks creating severe gaps in defensive coverage. These blind spots leave critical, high-priority IoCs undiscovered, drastically reducing the effectiveness of automated detection systems and leaving the organization vulnerable to known adversaries. Conversely, broadly subscribing to too many feeds or selecting feeds saturated with irrelevant indicators directly leads to wasted budget and an unmanageable volume of low-quality data. When SIEMs are forced to process millions of unnecessary indicators, the resulting alert fatigue transforms a valuable defensive asset into an operational liability.

Ultimately, this tension directly mirrors the classic performance trade-off of balancing recall against precision. In the context of CTI procurement, maximizing recall ensures that no critical threat indicator is missed, while maximizing precision ensures that the ingested data remains actionable, targeted, and operationally viable. Solving this problem with the additional goal of minimizing feed subscriptions requires leveraging advanced computational frameworks to identify the precise mathematical trade-offs where absolute coverage is guaranteed, subscription cost is minimized, and noise is limited.

1.2 Problem Statement

The fundamental challenge addressed in this thesis lies in selecting the optimal combination of CTI feeds that guarantees visibility into critical threats while avoiding the severe operational and financial penalties of data overload. This CTI feed selection problem begins with two primary data inputs. First, this problem assumes a defined universe of relevant, high-priority threat indicators, referred to as required IoCs, that an organization considers mandatory for its defensive posture. Secondly, a collection of candidate intelligence feeds is needed, each containing a mixture of both required and non-required IoCs, which are any noisy and irrelevant indicators that should be

ignored.

The primary goal of this problem is to select a subset of these candidate CTI feeds that successfully encompass all of the required IoCs, ensuring a strict 100% recall. However, achieving this full coverage constraint must be balanced against two secondary, often competing, operational objectives:

1. **Minimizing the Total Feed Count:** Reducing the absolute number of selected feeds directly minimizes the financial cost of vendor subscriptions and lowers the structural overhead needed to maintain data ingestion pipelines. This problem will assume that some selection of feeds can achieve full coverage.

2. **Minimizing the Inclusion of Non-Required IoCs:** Limiting the ingestion of noisy, irrelevant threat data maximizes overall intelligence precision. This prevents alert fatigue, limits false positives, and protects the SIEM infrastructure from unnecessary computational strain.

These dual objectives are inherently conflicting. Ensuring complete coverage while minimizing feed count increases the likelihood of selecting massive feeds that provide maximum coverage but introduce unnecessary noise, thus degrading precision. Conversely, minimizing noisy feeds to accumulate full coverage encourages stringing together many small, highly precise feeds, which would increase the total number of feeds consumed.

From a computational perspective, this feed selection problem is a manifestation of a well-known combinatorial optimization paradigm. The fundamental goal of minimizing the number of feeds while covering all required IoCs is mathematically equivalent to the Set Cover Problem, which is famously classified as NP-hard. Furthermore, minimizing the inclusion of undesired IoCs is related to the Weighted Set Cover Problem, where the cost of each selected set is proportional to the noise it intro-

duces. Consequently, achieving a true operational balance requires a multi-objective approach capable of navigating an exponentially growing problem space.

1.3 Research Objectives

To translate the complex operational challenges of threat intelligence procurement into actionable computational solutions, this research is guided by three primary objectives. First, it seeks to model CTI feed selection as a multi-objective optimization problem. By mathematically mapping the real-world requirements of a SOC to the mechanics of the Set Cover Problem, this research establishes a quantifiable, strictly bounded environment for algorithmic testing and validation. Second, to evaluate the scalability and operational boundaries of these solutions, this research empirically benchmarks a diverse set of computational strategies across a suite of synthetic problem instances designed to stress-test scalability and specific structural edge cases. Finally, this research aims to conduct a problem-space analysis using explanatory machine learning. By deploying XGBoost classification models to evaluate a problem's characteristics, this work utilizes feature importance metrics to explicitly define the underlying structural conditions that dictate algorithmic success.

Specifically, this comparative analysis assesses the computational efficiency, scalability, and overall solution quality of the following six algorithms:

- **Greedy Set Cover:** A classic, straightforward heuristic that iteratively prioritizes maximum coverage at each step, serving as a baseline for feed minimization, speed, and computational efficiency.

- **Ratio Greedy:** A greedy heuristic variation that guarantees full coverage by evaluating candidates through a strict cost-benefit ratio, selecting feeds based on their immediate coverage-to-noise impact.

- **Additive-Penalty Greedy:** The final greedy heuristic maintains the guarantee of full recall while maximizing a specific penalized objective function, effectively weighting feed selection toward higher precision and reduced noise.

- **Integer Linear Programming (ILP):** An exact mathematical framework that operates under definite constraints to explicitly solve for the global optimum, minimizing the scalarized objective function to guarantee absolute mathematical perfection given adequate time.

- **Linear Programming Guided Heuristic:** A hybrid approach that begins by solving a continuous linear relaxation of the ILP formulation, then utilizes those resulting fractional weights to drive a greedy selection process. This approach leverages the global optimization logic of ILP and scales to larger problem instances.

- **Evolutionary Algorithm (NSGA-II):** A highly adaptable metaheuristic that simulates natural selection and provides a scalable, multi-objective approximation capable of mapping the optimal trade-offs for feed selection.

By evaluating this diverse spectrum of algorithms, ranging from localized heuristics to exact global solvers and evolutionary algorithms, this research moves beyond merely identifying a single universally superior method. Instead, synthesizing empirical performance metrics and structural machine learning analysis yields a comprehensive, data-driven framework. This framework aims to provide actionable guidance by enabling selection of the most appropriate optimization strategy based entirely on the unique scale, redundancy, and structural constraints of the organization's specific threat intelligence ecosystem.

1.4 Thesis Structure

To comprehensively address the CTI feed selection problem, the remainder of this thesis is structured to logically progress from foundational concepts to empirical analysis. Chapter 2 establishes the operational and academic context of the problem by exploring the domain of CTI, the function of IoCs within security operations, and the current research gap regarding CTI optimization. Building upon this context, Chapter 3 translates these operational challenges into a formal computational framework, mapping the selection process to the classic Set Cover Problem, introducing the principles of multi-objective optimization, and formally proving that the underlying problem remains strictly NP-hard.

Chapter 4 explicitly defines the mechanics of the algorithms evaluated in this study, detailing the lightweight greedy heuristics, rigorous mathematical programming formulations, and the metaheuristic evolutionary approach. Chapter 5 then outlines the experimental methodology, detailing the specific evaluation metrics, the extraction of problem-space features, the suite of synthetic problem instances used for benchmarking, and the configuration of the explanatory machine learning classification model.

Following this framework, Chapter 6 presents the empirical findings. It evaluates the algorithms against randomized and specialized problem instances, conducts a problem-space analysis using the trained classification models to explain algorithmic performance, and concludes with an overall comparison of the solutions. Finally, Chapter 7 summarizes the primary contributions of the research, acknowledges the theoretical limitations of the experimental design, and proposes avenues for future academic exploration in threat intelligence optimization.

Chapter 2: Background

Building upon the primary research goals introduced in Chapter 1, it is essential to establish the operational and academic context of the domain before diving into the mathematical formulation of the problem. This chapter provides that necessary foundation, divided into two primary areas of focus. First, it details the ecosystem of Cyber Threat Intelligence (CTI) and its vital building blocks, Indicators of Compromise (IoCs). By examining how Security Operations Centers (SOCs) integrate these intelligence sources, this chapter effectively establishes the operational necessity for mathematical optimization. Second, this chapter reviews current literature and related works within the broader CTI landscape. By analyzing recent advancements in intelligence collection, data enrichment, and prior attempts at algorithmic feed reduction, the related works point to a research gap that the multi-objective framework introduced in the subsequent chapter is uniquely positioned to address.

2.1 Introduction to Cyber Threat Intelligence

To properly contextualize the mathematical formulations and algorithmic analysis that drive this research, it is first necessary to understand the underlying cybersecurity domain. This section establishes the operational realities of CTI and the lifecycle of its data: from CTI vendors into structured feeds to practical deployment within a SOC. While this intelligence empowers automated, proactive defense systems, the sheer volume of the modern global threat landscape has culminated in a severe scalability crisis. Consequently, this section bridges the gap between cybersecurity operations and computer science, establishing the operational and financial objectives that drive the transition from raw data aggregation to the strategic, mathematically curated

feed selection explored in this research.

2.1.1 Threat Intelligence and IoCs

To understand the importance of optimizing the selection of intelligence feeds, it is crucial to understand how information is organized and utilized within the cybersecurity field. CTI refers to evidence-based knowledge concerning malicious actors' motives, targets, and behaviors. The ultimate goal of CTI is to empower organizations to shift from reactive defenses, which attempt to mitigate damages and costs after an incident occurs, to a proactive security approach that minimizes risks and attempts to prevent breaches.

To accumulate relevant context, intelligence providers employ various approaches, such as deploying honeypots, monitoring dark web activity, and utilizing endpoint detection systems. These CTI providers can range from open-source intelligence (OSINT) feeds and industry-sharing communities to specialized commercial vendors. However, aspects including timeliness, organization, and reliability can vary drastically across providers, and often certain providers specialize in providing different vantage points of the threat landscape. In fact, research has shown there is remarkably little overlap in the specific threat data reported across different CTI vendors [4]. Thus, organizations seeking to ingest CTI must utilize information from multiple providers to capture a more complete picture of the threat landscape.

For practical application of this intelligence, providers often present information as Indicators of Compromise (IoCs), which can take the form of malicious IP addresses, URLs, and other network-related data. Aggregating this data in an organized format, CTI vendors sort IoCs into related groups, often referred to as CTI feeds, that share a common severity, behavioral pattern, or activity. For example, threat-sharing platforms like AlienVault's Open Threat Exchange (OTX) organize IoCs into "Pulses,"

which are essentially feeds of IoCs grouped by a specific malware campaign or threat actor [2]. By comparison, the open-source intelligence provider Abuse.ch curates highly specialized feeds based on the threat activity, such as URLhaus for tracking malicious URLs distributing malware and SSL Blacklist for blocklist data regarding malicious SSL certificates and JA3/JA3s fingerprints [1]. Ultimately, from a defensive standpoint, intelligence feeds provide essential organization by grouping relevant IoCs in an accessible and actionable way for security analysts. By aggregating intelligence from multiple feeds and providers, analysts are able to leverage this knowledge to better understand the threat landscape.

2.1.2 Security Operations

For practical application, a Security Operations Center (SOC) acts as the primary consumer of these feeds. A SOC acts as a centralized command hub where a team of cybersecurity professionals monitors, detects, analyzes, and responds to cyber threats in real-time to protect an organization's digital assets. SOC analysts rely on the IoCs within CTI feeds to inform their understanding of the threat landscape and fulfill their role as a proactive defense for their organization. By leveraging this intelligence, analysts can enact security measures based on historical and emerging adversary behaviors, shifting the organization's posture from reactive to preventative. Highlighting this value, the SANS Institute's 2024 Cyber Threat Intelligence Survey found that proactive threat hunting has emerged as the top overall use case for CTI, demonstrating its critical role in strengthening organizational security [24].

Beyond human analysis, this intelligence is also fed directly into automated defensive systems. For example, firewalls serve as barriers between a protected internal network and the Internet, filtering incoming and outgoing packets based on a predetermined set of security rules [27]. By leveraging CTI to inform firewall rules, analysts

can automatically filter traffic originating from known malicious sources before it breaches the network perimeter.

While firewalls guard the perimeter, Security Information and Event Management (SIEM) platforms act as the primary infrastructure for data storage and analytics for SOC teams. To accomplish this, SIEMs automatically collect, aggregate, and analyze log data from across an organization's various network devices and applications [30]. By ingesting CTI feeds, a SIEM can compare millions of internal network logs against known IoCs and alert security analysts if a matching threat is detected. Combining knowledge from intelligence feeds with these tools, SOC analysts can effectively monitor their network and threat landscape and implement proactive defensive measures.

2.1.3 Big Data Problem

For modern security operations, the sheer volume of threat data is staggering. Specifically, the global threat intelligence platform VirusTotal reports processing and analyzing over two million suspicious samples daily [29]. As SOCs synthesize IoCs from many feeds or even multiple providers, millions of IoCs are ingested daily, rapidly overwhelming human analysts. This big data problem can lead to various consequences for the organization. On a human level, data overload can cause severe alert fatigue, as the large quantity of false positives overwhelms and desensitizes analysts, leading to worsened decision-making and missed critical alerts [28]. On a technical and financial level, ingesting millions of daily IoCs degrades system performance and inflates SIEM licensing costs, as many major enterprise platforms utilize volume-based pricing models [14]. Thus, by ingesting all available IoCs, analysts establish extensive context of the threat landscape, but the massive volume of indicators introduces operational strain and financial implications.

To mitigate these issues caused by high information volume, it is critical to retain

and process timely and relevant IoC data; however, feeds are inherently noisy and often contain stale, irrelevant, or low-confidence IoCs. Because attackers constantly rotate their infrastructure and tactics, IoCs possess a highly volatile lifecycle, so it is necessary to proactively age out obsolete indicators to avoid massive operational overhead and degraded decision-making for security teams [19]. Furthermore, while macro-level overlap between different intelligence providers is categorically low, the sheer volume of data means that even minor redundancies across multiple feeds result in thousands of duplicate indicators, which further amplifies the data overload problem. Thus, it becomes critical for cyber operations to selectively filter IoC data and focus on relevant signals while disregarding noisy data by optimizing CTI feed selection. Before this thesis investigates the trade-offs and algorithms pertinent to this CTI feed selection challenge, the remainder of this chapter establishes the current work regarding CTI.

2.2 Related Works

The challenge of efficiently managing and comprehensively analyzing large amounts of IoCs has recently been studied in cybersecurity. Previous works have focused on improving the collection, enrichment, and interpretation of threat intelligence data. In order to understand the research gap that this selection problem addresses, the section will first introduce a few CTI-related works from recent years.

Gan and Liu [12] address the challenge of efficiently gathering large amounts of CTI by proposing a high-performance, AI-enhanced web crawling framework. Their approach automates the collection of data from sources such as vulnerability databases, dark web forums, and open-source repositories, achieving a 70% improvement in crawling efficiency over traditional tools like Scrapy. This work strengthens the process of assembling threat intelligence datasets, ensuring organizations have

access to timely and diverse data. While the referenced study also addresses data efficiency, its methodology emphasizes initial acquisition techniques, contrasting with the post-collection feed filtering optimization evaluated in this analysis.

In addition to data collection, recent research investigates the enhancement of cybersecurity data. Chen et al. [5] focus on improving the analysis of unstructured CTI reports by clustering Advanced Persistent Threat (APT) groups. By applying hierarchical clustering, they successfully identify relationships among APT groups, providing additional insight into anomalous behaviors and patterns. In a similar vein of research, Angelelli et al. [3] provide a robust statistical framework for cyber-vulnerability prioritization for incomplete or uncertain threat intelligence. Their approach synthesizes data from many sources, such as Common Vulnerability Scoring System (CVSS) scores, exploit availability, and host exposure levels, into a ranking model for vulnerabilities. Both Chen et al. [5] and Angelelli et al. [3] provide tools for data enrichment, yet neither address the topic of this research regarding the efficiency of data selection.

While previous works mentioned above do not closely relate to this study, the thesis research conducted by Sydorenko [25] addresses the similar problem of reducing the number of pulses, which provide IoC intelligence similarly to feeds, while maintaining high coverage of IoCs. This work focuses on reducing memory space and optimizing data inputs to SIEM systems by minimizing redundant IoC sets from the Open Threat Exchange (OTX) platform. Sydorenko addresses this problem using three techniques: the Set Cover Problem, Weighted Maximum Coverage, and Budgeted Maximum Coverage. Sydorenko then evaluates these different algorithms using fuzzy logic and considers factors such as data quality and trustworthiness. However, this approach does not focus on balancing goals of data reduction and coverage, so this provides an opportunity for future analysis.

This work extends the bounds of previous research. While Sydorenko's approach primarily addresses the minimum feed selection problem for SIEM data ingestion, this research introduces a multi-objective perspective. In addition to Sydorenko's Set Cover approaches, this thesis's methodology also explores submodular optimization, multi-objective Pareto frontier analysis, and scalable heuristics designed for real-time operational deployment. Thus, this comprehensive analysis attempts to balance recall, precision, and operational cost.

Whereas Gan and Liu focus on data collection, and Chen et al. and Angelelli et al. focus on intelligence enhancement, this work addresses a notable research gap: selecting the best subset of threat intelligence feeds to ensure complete coverage of required IoCs while minimizing feed quantity and noise.

Chapter 3: Mathematical Formulation

Before specific algorithms can be deployed to find the optimal combination of CTI feeds, a mathematical foundation for the problem space must be established. This chapter formalizes the challenge of CTI feed selection by mapping it directly to the foundational Set Cover Problem. While the classic Set Cover Problem provides a framework for minimizing a single metric, real-world CTI procurement involves conflicting operational priorities, such as minimizing feed subscriptions while simultaneously minimizing the ingestion of irrelevant, noisy data. Consequently, this chapter expands the traditional formulation into a multi-objective optimization framework. Introducing these competing objectives makes the underlying problem significantly more complex; rather than searching for a singular minimum, the mathematical objective shifts to discovering a continuous solution set of non-dominated trade-offs known as the Pareto front. This chapter concludes with a formal mathematical proof demonstrating that this multi-objective adaptation remains strictly NP-hard. By defining the trade-offs of the problem and establishing its exponential computational limits, this formulation provides the theoretical baseline required to justify the advanced algorithms explored in subsequent chapters.

3.1 Definition of Set Cover Problem

The proposed challenge of selecting the optimal combination of CTI feeds to encompass a specific group of required IoCs maps directly to a classic optimization problem in computer science: the Set Cover Problem (SCP). The SCP can be described as finding the minimum number of sets that together contain all elements in a given universe [7]. Formalized mathematically, the SCP can be defined as follows: Let U

be a universe of size n elements, such that $U = \{e_1, e_2, \ldots, e_n\}$. Let S be a family of m subsets of U, such that $S = \{S_1, S_2, \ldots, S_m\}$ where $S_i \subseteq U$ for $i = 1, 2, \ldots, m$. The objective is to find a collection $C \subseteq S$ of minimum size such that the union of the sets in C contains all of the elements in U.

The computational complexity for solving this problem has been well-researched and proven to be NP-hard. To begin, the SCP decision-based variation, confirming the existence of a valid set coverage, is one of Karp's original 21 problems, which were introduced in his foundational 1972 paper [17]. Because the decision-based version of the SCP is famously NP-complete, it logically follows that the optimization problem of finding the minimum collection to solve the SCP is strictly NP-hard [7].

Conceptually, this high complexity is intuitive, since guaranteeing an optimal solution, or the absolute minimum number of sets needed to cover all elements, requires an exhaustive search through all possible set combinations. As the number of sets and elements increases, the number of possible set combinations grows exponentially, so the problem becomes computationally infeasible. Specifically, the exhaustive search of evaluating every possible combination of m sets to find an optimal solution results in a worst-case time complexity of $O(2^m)$. This exponential time bound in conjunction with Karp's formal proof effectively provides intuitive justification and formal mathematical support of the SCP's complexity.

3.2 Multi-Objective Optimization

For classic optimization problems, the approach focuses on a single objective in which value is maximized or cost minimized. In particular, the SCP traditionally minimizes a single objective, the number of sets, subject to the constraint of full coverage for all elements in the universe. Multi-objective optimization problems require the simultaneous optimization of two or more objectives, which are often competing in nature.

Within the context of the SCP, multi-objective optimization could involve balancing the minimized number of sets against the maximization of set quality, or minimizing the number of unnecessary elements covered. Because improving one objective frequently requires a trade-off in another, multi-objective optimization adds further computational difficulty to general optimization problems.

Regarding the focus of this paper, multi-objective optimization provides an effective framework for setting up this CTI feed selection problem. Minimizing the number of feeds and non-required IoCs can be competing objectives as larger feeds with more IoC coverage often contain additional unnecessary data and vice versa. In other words, minimizing the number of feeds may introduce noise, while minimizing the volume of irrelevant indicators may increase the number of feeds needed to achieve full coverage.

Traditionally, basic algorithms resolve this conflict by simply reducing the problem into a single objective, a process known as scalarization [21]. The most common approach is the weighted sum method, which aggregates competing objectives into a single fitness score (e.g., $F(x) = w_1 f_1(x) + w_2 f_2(x)$). While computationally straightforward, it requires the exact mathematical trade-off between distinct objectives, such as intelligence noise versus subscription cost, before execution.

In practice, this ambiguity can be bypassed by pushing these weights to a lexicographic order [10]. By assigning a massively dominant weight to one objective, the solver enforces a strict hierarchical priority, optimizing the primary metric to its absolute limit before addressing the secondary. While often effective, this lexicographic scalarization rigidly funnels the algorithm toward a single, localized extreme.

Ultimately, because single-objective reductions fail to map the full spectrum of available trade-offs, mathematicians instead use the concept of Pareto dominance to thoroughly explore competing objectives [8]. Conceptually, a solution A is said to

"dominate" a solution B if A is no worse than B in all objectives, and strictly better than B in at least one objective. If A is better in some objectives but worse in others compared to B, neither solution dominates the other; they are considered mutually non-dominated. This definition can be expressed mathematically. Given a set of n possible solutions $x = \{x_1, x_2, \ldots, x_n\}$ and m objective functions represented by $f_i(x)$ for $i = 1, 2, \ldots, m$, a solution x_1 dominates x_2 if the following two conditions hold. First, x_1 is not worse in any objective, so for a maximization problem, $f_i(x_1) \geq f_i(x_2)$ for all $i \in \{1, 2, \ldots, m\}$. Second, x_1 must be strictly better, or greater for maximization, than x_2 in at least one objective: $f_j(x_1) > f_j(x_2)$ for at least one $j \in \{1, 2, \ldots, m\}$.

Building upon the concept of individual Pareto dominance, this framework can be applied to the entire solution space to identify trade-offs and find overall optimality. A solution x^* within the set of all possible solutions, X, is considered Pareto optimal if there exists no other solution in X that dominates it. The collection of all such non-dominated solutions forms the Pareto optimal set within the decision space. When the values of these solutions are evaluated and plotted in the objective space, this graphical boundary is referred to as the Pareto frontier, which serves as a valuable tool for understanding the scope of trade-offs within the solution space.

In the context of CTI feed selection, the Pareto frontier is highly valuable because it visualizes the optimization limits. Solutions that fall beneath the frontier are suboptimal, as a better combination of feeds exists that improves at least one objective without degrading the other. Conversely, all solutions along the Pareto frontier act as a set of optimal choices. Thus, from an application standpoint, an organization can select a specific point on the frontier that best aligns with its priorities, budget, and operational constraints.

3.3 NP-hard Proof

We prove that the Multi-Objective CTI Feed Selection Problem is NP-hard using a proof by restriction. To map this domain to the proven NP-hard Set Cover Problem, let the mathematical universe of elements, U, represent the total set of required IoCs, and let the family of subsets, S, represent the available CTI feeds, where each feed provides a specific subset of IoCs. For this proof, a simplified instance of the problem will be considered where other objectives, specifically the impact of noisy IoCs, are uniform across all available feeds. In this restricted case, the sole objective becomes minimizing the number of feeds (S) required to completely cover the universe of required IoCs (U). This reduces the CTI feed selection problem to the classical optimization version of the Set Cover Problem, which is strictly NP-hard [17]. Because this restricted case is NP-hard, it logically follows that the more complex, multi-objective formulation is also NP-hard. ■

While the Multi-Objective CTI Feed Selection Problem is computationally intractable for large datasets (NP-hard), it is strictly decidable. The search space is bounded by the finite number of available CTI feeds, m. Consequently, the total number of possible feed combinations is 2^m. Because the search space is discrete and strictly finite, an exhaustive search algorithm is guaranteed to eventually identify the exact true Pareto optimal set. However, while theoretically solvable, the exponential worst-case time complexity of $O(2^m)$ renders this brute-force approach practically infeasible for modern SOC environments evaluating hundreds or even thousands of feeds.

Chapter 4: Algorithms

To address the combinatorial explosion characteristic of the Set Cover Problem, computer scientists must carefully balance computational feasibility with mathematical accuracy. While classic greedy approaches offer a straightforward methodology with low time complexity, they struggle to navigate complex, multi-objective problem spaces and frequently become trapped in local optima. Conversely, exact mathematical models, such as Integer Linear Programming, guarantee optimality for the given objective but scale poorly as the problem expands. To bridge this gap, this research also explores advanced heuristic and metaheuristic frameworks, specifically a heuristic guided by Linear Programming and the NSGA-II Evolutionary Algorithm, with the goal of providing highly optimized, reliable approximations within practical timeframes.

4.1 Greedy Heuristics

To establish a baseline for approximating a solution for the CTI feed selection problem, this work first evaluates greedy heuristic approaches. Fundamentally, a greedy algorithm operates by optimizing a specific objective or mathematical goal at each individual step of the decision-making process. At every iteration, the algorithm strictly selects the locally optimal choice for a given criterion without consideration for how this decision might impact future selections. For this CTI feed selection application, greedy techniques iteratively select the single feed that provides the highest immediate value based on the objective until full coverage of all required IoCs is reached. The rationale behind this greedy design is that a progression of locally optimal decisions would eventually lead to a globally optimal solution. However, because greedy algo-

rithms do not account for future implications or backtrack to correct past decisions, this is not always the case. In fact, greedy approaches are heavily susceptible to becoming trapped in local optima, often yielding suboptimal results in highly complex problem spaces.

Despite susceptibility to local optima, greedy heuristics remain a foundational tool for solving the Set Cover Problem (SCP) because they offer mathematically proven performance guarantees driven by the principle of diminishing returns. In combinatorial optimization, this principle is formally defined as submodularity, and a function is considered submodular if the marginal gain of adding a new element to a given set decreases as the set grows larger [23]. Regarding the SCP, the underlying objective function of greedily covering a universe of elements exhibits this submodular property; as the cumulative coverage grows, the marginal gain of integrating any additional feed naturally diminishes. Furthermore, classical coverage functions are strictly monotone, meaning the total value of the set never decreases as new elements are added. Because of this monotone submodularity, the greedy strategy of selecting the set, or feed in this application, with the largest immediate payoff is mathematically bounded against yielding poor long-term outcomes.

Specifically, for this minimum-feed objective, the algorithm is mathematically guaranteed to find a solution strictly bounded by $O(\ln n)$, where n represents the total size of the universe to be covered [6]. While introducing complex secondary objectives, such as noise reduction, alters these strict classical bounds, this foundational guarantee provides the necessary justification to utilize the greedy framework as a highly scalable starting point for big-data threat intelligence optimization.

Building upon this theoretical foundation, the specific selection criteria and objectives used at each iterative step of a greedy algorithm can be modified to prioritize different operational goals. While baseline algorithms strictly adhere to classical

mathematical bounds, adapting these theoretical models into domain-specific heuristics allows for the integration of secondary objectives. To explore the balance between fulfilling required IoC coverage constraints, minimizing the total number of CTI feeds, and maximizing precision, this research evaluates three distinct greedy approaches: a classical coverage baseline, and two adapted heuristics that actively penalize noise.

4.1.1 Greedy Set Cover

To begin the analysis of greedy heuristics, the standard Greedy Set Cover approach provides a fast, interpretable baseline. While all algorithms explored in this paper necessitate full coverage of the set of required IoCs, the primary objective of this specific approach is to achieve that complete recall using the absolute minimum number of feeds. To accomplish this, the algorithm's primary loop iterates until full coverage is reached, selecting the feed at each step that provides the highest raw count of novel, previously uncovered target IoCs. By maximizing the marginal gain in coverage at every single iteration, the algorithm attempts to minimize the total number of steps, and therefore, the total number of feeds, required to satisfy the coverage constraint.

Because it strictly follows the classical minimization objective, this baseline is mathematically guaranteed to bound the total number of selected feeds within an $O(\ln n)$ approximation of the true optimal count. However, with this sole objective driving its decisions, the Greedy Set Cover approach does not consider the dual objective of precision, except in tie-breaking scenarios where multiple feeds offer the same coverage gains. Thus, while effective at minimizing the number of feeds, this baseline fails to balance multiple objectives, as it blindly accepts any noisy or irrelevant IoCs that a massive feed may introduce.

Algorithm 1 details the functionality of the Greedy Set Cover approach. Initially, the method assigns the full set of IoCs required to achieve full coverage to

the *uncovered* tracker and creates an empty set where the final *selected* feeds will be iteratively collected (line 2). The algorithm is driven by a primary `while` loop that strictly enforces the full-recall constraint since it will not terminate until the *uncovered* set is entirely empty and full coverage is therefore achieved. During each iteration of the `while` loop, the algorithm evaluates every available feed by calculating the mathematical intersection to determine the marginal gain in new coverage (line 6). This newly-evaluated feed coverage is compared with the current highest coverage found (line 7), which is updated if the new coverage is greater than the previous best (line 8). In addition to minimizing the number of feeds, the algorithm accounts for the dual objective of precision, but exclusively as a tie-breaking mechanism. If two feeds offer the exact same marginal coverage gain (line 9), the ComparePrecision helper function is invoked to select the feed that introduces fewer irrelevant, non-required IoCs. Finally, the chosen feed is appended to the *selected* list, its IoCs are subtracted from the *uncovered* set, and the loop repeats until global coverage is achieved.

Algorithm 1 Greedy Set Cover

```
 1: function GreedySetCover(feeds, req_iocs)
 2:     uncovered ← req_iocs; selected ← ∅
 3:     while uncovered do
 4:         best_f ← None; best_cvr ← ∅
 5:         for all f, f_iocs in feeds do
 6:             cvr = uncovered ∩ f_iocs
 7:             if |cvr| > |best_cvr| then
 8:                 best_f ← f; best_cvr ← cvr
 9:             else if |cvr| = |best_cvr| then
10:                 best_f, best_cvr ← ComparePrecision(f, best_f)
11:             end if
12:         end for
13:         selected ← selected + {best_f}
14:         uncovered ← uncovered \ best_cvr
15:     end while
16:     return selected
17: end function
```

This algorithm serves as a strong baseline because its selections are interpretable, and it scales efficiently to large problem instances. However, considering the number of irrelevant IoCs only in a tie-breaking scenario does not adequately balance the dual goals of feed quantity minimization and precision maximization. While this algorithm establishes a vital foundation for this analysis, the following two greedy approaches introduced in this section directly integrate both objectives into their primary selection criteria.

4.1.2 Ratio Greedy

To address the operational strain of ingesting noisy CTI feeds, the Ratio Greedy heuristic provides an enhanced methodology that actively integrates a precision metric into its primary selection criteria. Rather than assessing feeds based solely on raw coverage, this approach evaluates them based on the proportion of new coverage gained relative to the amount of irrelevant IoCs incurred. Mathematically, the algorithm evaluates each available feed using the following ratio:

$$\text{Ratio} = \frac{[\text{number of new required IoCs}]}{1+ \ [\text{number of new non-required IoCs}]}$$

By adding 1 to the denominator, this metric accounts for the edge case of a feed providing perfect precision (zero irrelevant IoCs) without triggering a division-by-zero error. At each iteration, the algorithm selects the feed that maximizes this ratio, continuing until the full-recall constraint is satisfied. In this way, the optimization of this cost-to-benefit proportion at every step of the selection process directly balances the dual objectives of minimizing the total number of feeds while maximizing overall precision.

This ratio-based selection technique is theoretically inspired by approximation algorithm literature for the Weighted Set Cover problem, a well-established gener-

alization of the traditional SCP. In classical Weighted Set Cover, evaluating sets by their cost-effectiveness, specifically the ratio of a static weight to its marginal coverage gain, provides a mathematically bounded $O(\ln n)$ approximation of the optimal total cost [6]. However, to accurately reflect that overall precision metrics do not double-count overlapping noise, this study adapts the classical framework into a dynamic heuristic. Rather than using static feed costs, the algorithm calculates a dynamic cost based on the volume of novel irrelevant IoCs a feed introduces. By inverting the traditional cost-per-element ratio to maximize the marginal signal-to-noise ratio at every step, this algorithm explicitly trades classical worst-case mathematical guarantees for highly optimized, domain-specific precision.

In Algorithm 2, the Ratio Greedy method initializes by establishing both the *selected* feed array and a *total_covered* tracker as empty sets (line 2). The primary `while` loop continuously evaluates available feeds to find the optimal local step, completing only when the full-coverage constraint is met, which is mathematically evaluated by checking if the set of required IoCs (*req_iocs*) is a subset of the *total_covered* set (line 3).

The core difference from Greedy Set Cover occurs within the inner evaluation loop. Instead of merely tracking raw coverage, the algorithm explicitly calculates two distinct sets for every unselected feed (lines 7-8): the novel required IoCs it provides that have not yet been covered (*new_req*), and the new noisy IoCs it introduces (*new_nonreq*). The algorithm then computes the *score* by dividing the number of the novel required IoCs by the quantity of novel noise, explicitly adding 1 to the denominator to safely evaluate feeds with perfect precision (line 9). The feed that maximizes this ratio is temporarily stored as the best choice, alongside a combined set of its novel IoCs (*best_cvr*) (lines 10-12). Finally, the feed with the highest ratio is appended to the *selected* array, its associated new IoCs are added to the *total_covered*

set (lines 18-19), and the selection process repeats until full coverage is obtained.

Algorithm 2 Ratio Greedy

```
1: function RatioGreedy(feeds, req_iocs)
2:     selected ← ∅; total_covered ← ∅
3:     while not req_iocs ⊆ total_covered do
4:         best_f ← None; best_cvr ← ∅
5:         best_score ← −1;
6:         for all f, f_iocs in feeds do
7:             new_req ← (f_iocs ∩ req_iocs) \ total_covered
8:             new_nonreq ← (f_iocs \ req_iocs) \ total_covered
9:             score ← |new_req|/(1 + |new_nonreq|)
10:            if score > best_score then
11:                best_f ← f; best_score ← score
12:                best_cvr ← new_req ∪ new_nonreq
13:            end if
14:        end for
15:        if best_f = None then
16:            break
17:        end if
18:        selected ← selected ∪ {best_f}
19:        total_covered ← total_covered ∪ best_cvr
20:    end while
21:    return selected
22: end function
```

Ultimately, the Ratio Greedy approach serves as a powerful enhancement to the Greedy Set Cover baseline by directly handling the multiple objectives of CTI feed selection. Because its density ratio actively penalizes marginal noise gain, the algorithm is significantly more likely to achieve higher precision across the final selection. However, this advantage introduces a critical vulnerability if the coverage-to-noise trade-off is not carefully balanced: the algorithm can over-optimize for precision to a detrimental degree. By heavily penalizing noise, it is prone to rejecting a massive, highly productive feed, ultimately undermining the objective of minimizing overall feed count. Furthermore, while the method remains highly scalable for large prob-

lems, its use of a proportional ratio rather than raw counts makes the logic behind its individual selections slightly less intuitive to interpret than the Greedy Set Cover approach.

Thus, the practical superiority of this algorithm is heavily dependent on operational context. If an organization prioritizes minimizing false positives and analyst fatigue, the Ratio Greedy method is notably superior as it directly evaluates precision. Conversely, if the strict minimization of feed subscriptions is the primary goal, the baseline Greedy Set Cover remains the more effective, directed tool. To establish a final comparison point for the greedy approaches, the Additive-Penalty Greedy algorithm will be examined next, which similarly maximizes a mathematical objective at each iterative step to optimize this feed selection problem's competing goals.

4.1.3 Additive-Penalty Greedy

Like the Ratio Greedy approach, the Additive-Penalty Greedy algorithm guarantees full coverage while actively mitigating noise. In contrast, however, it shifts the core selection criteria from a proportional density evaluation to the maximization of an Additive-Penalty objective function. This function mathematically rewards candidate feeds that provide novel target IoCs while simultaneously subtracting a penalty for feeds that introduce a high volume of irrelevant data. Mathematically, the algorithm evaluates each available feed f using the following objective function:

$$F(f) = [\text{new required IoCs}] - [\lambda \cdot (\text{new non-required IoCs})]$$

where λ represents a tunable scalar weight that dictates the strictness of the noise penalty. Because this approach integrates a non-monotone penalty to actively penalize noise, it relaxes the strict mathematical bounds of the traditional greedy Set Cover. However, by framing the evaluation through Additive-Penalty optimization,

the algorithm intentionally trades classical worst-case theoretical guarantees for highly optimal, domain-specific local decisions that dynamically balance precision with required coverage.

Algorithm 3 details the execution of the Additive-Penalty Greedy approach. Mirroring the structure of the Ratio Greedy method, this algorithm initializes by establishing both the *selected* feed array and a *total_covered* tracker as empty sets (line 2). The primary `while` loop continuously evaluates available feeds to find the optimal local step, continuing until the full-coverage constraint is mathematically satisfied (line 3).

During the inner evaluation loop, the algorithm isolates the novel required IoCs (*new_req*) and the novel irrelevant, noisy IoCs (*new_nonreq*) introduced by each candidate feed (lines 7-8). Instead of calculating a ratio, the algorithm computes a marginal gain (line 9) by taking the number of the newly introduced required IoCs and subtracting the penalty of the new irrelevant noise, scaled by a tuning weight, λ, which is optimized separately via grid search. The feed that yields the highest marginal gain is temporarily marked as the best choice (lines 10-12).

However, if all remaining unselected feeds are excessively noisy, their weighted penalties may outweigh their coverage rewards, resulting in a negative or zero net gain across all available options. To prevent the algorithm from halting prematurely or failing to converge, a "rescue step" is implemented. If no feed produces a strictly positive marginal gain (line 19), the MaxCoverage helper function is invoked to temporarily override the additive penalty. This forces a deterministic greedy selection based purely on advancing the remaining required coverage, guaranteeing that the strict 100% recall constraint is ultimately met. Finally, the chosen feed is appended to the selected array, all of its associated IoCs are merged into the *total_covered* set (lines 19-20), and the process repeats until global coverage is achieved.

Algorithm 3 Additive-Penalty Greedy

```
function AdditivePenaltyGreedy(feeds, req_iocs, λ)
    selected ← ∅; total_covered ← ∅
    while not req_iocs ⊆ total_covered do
        best_f ← None; best_cvr ← ∅
        best_gain ← −∞
        for all f, f_iocs in feeds do
            new_req ← (f_iocs ∩ req_iocs) \ total_covered
            new_nonreq ← (f_iocs \ req_iocs) \ total_covered
            gain ← |new_req| − λ · |new_nonreq|
            if gain > best_gain then
                best_f ← f; best_gain ← gain
                best_cvr ← f_iocs
            end if
        end for
        if best_f = None or best_gain ≤ 0 then
            best_f ← MaxCoverage(feeds, selected, req_iocs)      ▷ Rescue Step
            best_cvr ← feeds[best_f]
        end if
        selected ← selected ∪ {best_f}
        total_covered ← total_covered ∪ best_cvr
    end while
    return selected
end function
```

Overall, the Additive-Penalty Greedy algorithm serves as a direct, parallel alternative to the Ratio Greedy approach. Both methods were implemented in this study to explicitly compare how distinct mathematical strategies navigate the multi-objective trade-off between feed quantity and precision. While Ratio Greedy rigidly evaluates feed density based on a fixed proportion, Additive-Penalty Greedy distinguishes itself through its parametrizable flexibility. By utilizing the λ parameter, it allows analysts to explicitly weight the cost of noise against the benefit of coverage. When λ is set to zero, the algorithm mathematically mirrors the baseline Greedy Set Cover, prioritizing raw coverage without penalty for noise introduced. As λ increases, the algorithm becomes increasingly sensitive to precision, penalizing noisy feeds more aggressively.

This makes the Additive-Penalty approach highly adaptable to specific operational tolerances, though it introduces the analytical trade-off of requiring empirical tuning to identify the highest performing λ value for a given dataset.

4.1.4 Review of Greedy Heuristics

Collectively, these greedy algorithms establish a vital baseline for the subsequent analytics. Rooted in established combinatorial optimization theory, they consistently yield robust, well-defined solutions. Furthermore, their polynomial time complexity makes them extremely scalable to massive threat intelligence datasets. Implementing a variety of these greedy approaches not only builds a foundational understanding for more complex heuristics, but it also rigorously compares their distinct mathematical strategies against one another. Within this comparison, the baseline Greedy Set Cover stands out as the most interpretable approach, driven purely by the coverage requirement. Conversely, the Ratio Greedy and Additive-Penalty Greedy algorithms offer more sophisticated alternatives that directly evaluate each choice based on the dual objectives of maximizing precision while minimizing feed count.

However, the inherent nature of greedy selection means it cannot adapt past choices or robustly navigate highly complex problem spaces. Because these algorithms strictly make locally optimal decisions at each iteration, they are highly prone to converging on local optima, failing to consider the future impact of a current selection on the final decision. To overcome these limitations, this chapter will next introduce more adaptable, albeit more complex, algorithmic strategies. These advanced methods are designed with the explicit goal of achieving higher robustness and escaping local optima, all while maintaining the scalability required for operational CTI environments.

4.2 Mathematical Programming Approaches

While greedy heuristics provide efficient, polynomial-time approximations for CTI feed selection, this analysis will shift to a more complex approach that offers stricter mathematical guarantees. To find the guaranteed global optimum for the Set Cover Problem, researchers classically formulate the task as an Integer Linear Programming (ILP) problem [22]. By defining the selection of each possible feed as a binary decision variable, an ILP can rigorously enforce the full-coverage constraint while mathematically minimizing an explicit objective function.

To formalize this concept mathematically, let $x_f \in \{0, 1\}$ represent the binary decision of whether to ingest feed f from the available set of feeds $\mathcal{F}$, and let c_f represent a generic "cost" associated with that feed. For a single-objective model, the global optimum is found by minimizing the total cost:

$$\min \sum_{f \in \mathcal{F}} c_f x_f$$

This minimization is subject to the strict coverage constraint that every target IoC i, within the complete universe of required IoCs R, must be present in at least one selected feed f from a given set of feeds $\mathcal{F}$:

$$\sum_{f \in \mathcal{F}: i \in f} x_f \geq 1, \quad \forall i \in R$$

$$x_f \in \{0, 1\}, \quad \forall f \in \mathcal{F}$$

Because the ILP formulation computes an exact mathematical solution, it scales exponentially in the worst case as the number of feeds and IoCs grows, so it is often unrealistic in real-world scenarios. That being said, this exact global optimum provides a useful benchmark for analysis of heuristic methods and offers meaningful solutions for small-scale problems.

However, it is crucial to recognize that an ILP is only guaranteed to find an optimal solution for the specific mathematical objective it is given. Consequently, the objective function must flawlessly model the operational problem space. This presents a significant challenge when applying ILPs to multi-objective optimization problems. When competing goals, such as simultaneously minimizing total feed count and minimizing introduced noise, are scalarized into a single objective function via weighted trade-offs, the solver rigidly optimizes that exact equation. As a result, this process often yields solutions that are mathematically optimal for the provided weights but sub-optimal in practice, as it is a known difficulty to establish trade-off parameters that perfectly capture real-world operational priorities [10].

To translate these theoretical models into executable computational frameworks, the implementations explained in the following subsections rely on PuLP, an open-source linear programming API written in Python. To execute this model, the COIN-OR Branch and Cut (CBC) solver utilizes a standard branch-and-bound algorithm with cutting planes to systematically explore the binary decision space and prune suboptimal branches until the mathematically exact solution is proven. First, this section will detail the formulation of a lexicographic ILP model designed specifically to bypass the aforementioned scalarization flaws, thereby guaranteeing an operationally viable multi-objective solution. Subsequently, a Linear Programming (LP) guided heuristic will be introduced that implements a continuous relaxation of this exact model to inform the feed selection process.

4.2.1 Integer Linear Programming

To implement the mathematical framework established above while explicitly avoiding the pitfalls of weighted scalarization, this ILP algorithm was designed using a lexicographic multi-objective optimization approach. Rather than forcing a blended

trade-off between feed quantity and feed noise, a strict mathematical hierarchy was enforced. By establishing this hierarchy, the need to empirically tune competing weight parameters is entirely eliminated, thus avoiding a notoriously difficult process, as the exact exchange rate between including an additional feed and accepting a specific volume of noise is rarely known in an operational setting.

For this ILP algorithm, finding the absolute minimum number of feeds required to achieve complete coverage of the target IoCs is configured as the primary objective. Minimizing the volume of irrelevant indicator noise serves as the secondary objective and acts purely as a global tie-breaker among feed selections that successfully meet this primary minimum-feed, full-coverage constraint. While this approach shares conceptual similarities with the baseline Greedy Set Cover regarding precision as a tie-breaker, the mathematical mechanics differ fundamentally. The ILP evaluates and refines the entire solution space globally to apply this tie-breaker, whereas the greedy heuristic evaluates precision at each local, incremental step. Given infinite time, this lexicographic formulation mathematically guarantees the discovery of a Pareto-optimal solution; it rigorously locks in the absolute theoretical minimum for feed quantity, and from that subset of valid mathematical solutions, extracts one that introduces the least operational noise.

However, finding this exact discrete optimum remains computationally NP-hard because, although highly efficient, the solver is directly and optimally solving the proposed problem, which was proven NP-hard in chapter 3. Therefore, executing the solver to absolute mathematical convergence becomes infeasible as the available threat intelligence scales. To account for real-world operational environments, this research introduced a dynamic time limit parameter to the CBC solver. Given an indefinite time frame, the branch-and-bound algorithm would converge to optimality; however, the ILP's execution window is explicitly bounded by the execution time

of the subsequent Evolutionary Algorithm after a fixed number of generations. This establishes a practical baseline by evaluating the best selection the solver can discover within the same time that it takes an advanced heuristic to converge.

When this dynamic time limit is reached, the CBC solver terminates its search and returns the best feasible, full-coverage integer selection discovered up to that point. However, this solution may not represent the proven global optimum for feed or noise minimization, directly reflecting the practical trade-off between mathematical exactness and operational speed. Moreover, if the solver has not yet found a valid integer solution when the time limit expires, it terminates in an infeasible state. In practice, extracting the variable states from the solver at this stage often yields a broken selection with only partial coverage or zero-values, as these represent the intermediate, mathematically incomplete branch the solver was exploring when execution was halted. While these solutions that fail to achieve full coverage are considered suboptimal in later analysis, they are also fundamentally invalid, representing a severe operational limitation of relying strictly on time-bounded ILPs for massive CTI datasets.

Algorithm 4 details the precise implementation of this ILP model. To translate the feed selection problem to fit an Integer Linear Programming model, the formulation relies on two sets of binary decision variables, linked constraint logic, and a heavily skewed objective function. After setting up the problem as an ILP, this implementation leverages the efficient, optimized CBC solver built into the PuLP library that employs branch-and-bound techniques to search the solution space and converge to optimality.

To begin, the model initializes binary variables $x_f \in \{0, 1\}$ for every candidate feed (lines 2-4) and variables $z_n \in \{0, 1\}$ for every non-required IoC present across the entire problem instance (lines 5-7). This variable assignment for non-required IoCs is

a computationally expensive task in itself as it necessitates prior knowledge of irrelevant indicators present across all feeds or iteration through all feeds to collect these IoCs. Compared to the heuristics analyzed in this study, this ILP approach requires significant upfront computation to find these irrelevant IoCs, which is a non-trivial task as the number of feeds and IoCs scales. However, this cost cannot be avoided to accurately model the precision of the final solution since simply adding the total noise of each selected feed would result in double-counting overlapping indicators. To resolve this, lines 9 through 14 establish a bounding linkage between the variables: $z_n \geq x_f$. This inequality guarantees that if a specific feed is selected ($x_f = 1$), every non-required indicator contained within that feed is also forced to an active state ($z_n = 1$). Because z_n is a binary variable representing the IoC itself, overlapping noise across multiple selected feeds resolves to a single $z_n = 1$ state, perfectly modeling a true set union.

Lines 15 through 23 enforce the strict mathematical baseline of full coverage required for the Set Cover Problem. For every required IoC, the algorithm dynamically compiles a subset of x_f decision variables into the $covers$ set representing the feeds that contain that specific IoC. A linear constraint is added to the solver, mathematically requiring that the sum of these specific x_f variables within the $covers$ set must be greater than or equal to one. In other words, for every required IoC, at least one of the feeds covering it ($x_f \in covers$) must be included in the final selection ($x_f \geq 1$). This permanently restricts the solver from exploring any binary combinations that leave a required indicator uncovered.

The core of the lexicographic approach is defined in lines 24 through 26. To force the strict hierarchy between the first objective (feed minimization) and the secondary objective (noise minimization), the penalty weight for selecting a feed (α) is set strictly greater than the theoretical maximum penalty for noise. Specifically, α is defined as

the total count of all undesired IoCs in the given universe, plus one (line 24), while the noise penalty (β) remains at 1, meaning no additional weight (line 25).

Algorithm 4 Integer Linear Programming

```
1:  function ILP(feeds, req_iocs, nonreq_iocs, time_limit)
2:      for all f in feeds do
3:          x_f ← LpVariable(cat=LpBinary)              ▷ If feed f is selected
4:      end for
5:      for all n in nonreq_iocs do
6:          z_n ← LpVariable(cat=LpBinary)              ▷ If IoC n is included
7:      end for

8:      prob = LpProblem(LpMinimize)                    ▷ Initialize a minimization problem

9:      for all f, f_iocs in feeds do
10:         f_nonreq ← f_iocs ∩ nonreq_iocs
11:         for all n in f_nonreq do
12:             prob = prob + z_n ≥ x_f                 ▷ If feed f is selected, IoC n is included
13:         end for
14:     end for

15:     for all r in req_iocs do
16:         covers ← ∅
17:         for all f, f_iocs in feeds do
18:             if r ∈ f_iocs then
19:                 covers ← covers ∪ {x_f}
20:             end if
21:         end for
22:         prob = prob + lpSum(covers) ≥ 1             ▷ Each r in ≥ 1 selected feeds
23:     end for

24:     α ← len(nonreq_iocs) + 1
25:     β = 1
26:     obj ← α · Σ_f x_f + β · Σ_n z_n                 ▷ Define objective function
27:     prob = prob + obj

28:     solver = PULP_CBC_CMD(timeLimit = time_limit)
29:     status ← prob.solve(solver)

30:     return {f ∈ feeds | x_f = 1}
31: end function
```

The solver's final objective evaluates $\min(\alpha \sum x_f + \beta \sum z_n)$. Because the cost of a single x_f variable is mathematically heavier than the cost of activating every single z_n variable simultaneously, it is fundamentally impossible for the solver to justify adding an extra feed to reduce noise. Consequently, the CBC solver strictly minimizes the feed count first, utilizing the $\beta \sum z_n$ calculation solely to tie-break between valid, minimal-feed selections to find the one with the lowest aggregated noise.

Ultimately, this exact lexicographic formulation provides a mathematically rigorous approach for the threat intelligence feed selection problem that guarantees an optimal solution for the defined objective function. By guaranteeing full coverage of required IoCs and strictly prioritizing feed minimization before precision optimization, this ILP approach produces an optimal selection of feeds, given infinite time bounds. In this study, however, the implementation does not always yield a Pareto-optimal solution because the NP-hard nature of the ILP necessitates the use of time bounds to remain operationally feasible. Thus, in larger-scale or highly-complex problem spaces, the ILP may be limited by the time constraint before converging to an optimal solution.

Nevertheless, the ILP remains an invaluable analytical tool as it operates flawlessly on smaller problems and establishes a reliable benchmark for evaluating the other heuristic approaches. To bridge the gap between the exactness of this integer model and the speed of polynomial-time approximations, like the greedy methods, the following section will explore a Linear Programming guided heuristic that leverages the continuous relaxation of this model to optimize feed selection.

4.2.2 Linear Programming Guided Heuristics

While an exact ILP model guarantees eventual convergence to an optimal solution by evaluating the entire binary solution space, its NP-hard combinatorial scaling re-

stricts its utility for massive datasets. Conversely, the baseline Greedy Set Cover algorithm scales efficiently in polynomial time, but its localized, incremental choices often result in suboptimal long-term decisions. To synthesize the global perspective of mathematical programming with the computational efficiency of greedy heuristics, this research implemented a deterministic Linear Programming (LP) Relaxation algorithm to inform feed selection.

This method takes the rigid, discrete ILP formulation and "relaxes" the integer constraint. Instead of requiring the decision variable to be strictly binary ($x_f \in \{0, 1\}$), LP-Relaxation allows x_f to take any continuous fractional value such that $0 \leq x_f \leq 1$. The solver can compute a global, fractional optimum in polynomial time, reducing the time complexity compared to its NP-hard integer counterpart [16][18]. However, this conceptually translates to the solver choosing a fraction of a CTI feed (e.g., purchasing 0.8 of a CTI feed), which is physically impossible in a real-world scenario. Thus, this continuous output cannot be used directly as a solution.

Instead, as demonstrated by Williamson and Shmoys [31], these fractional variable assignments can be utilized as mathematically rigorous confidence weights to guide a deterministic rounding heuristic. Algorithm 5 extracts and then uses these confidence weights to prioritize feeds for greedy selection until full coverage is obtained. This approach functions in two consecutive phases; specifically, Algorithm 5 demonstrates the progression from solving the continuous Linear Program (lines 2-4) to leveraging the derived weights for greedy selection (lines 5-16).

To clearly separate the logic of defining the LP problem instance, DeriveLP is called first to establish the decision variables, constraints, and objective for the solver, and this method is detailed in Algorithm 6. Instead of a strict continuous relaxation of the ILP implemented in the previous section, this model introduces several structural changes to account for fractional results. While this algorithm

Algorithm 5 LP-Guided Heuristic

function LPHeuristic($feeds, req_iocs, nonreq_iocs, \gamma, \mu$)

  $prob, x_f$ = DeriveLP($feeds, req_iocs, nonreq_iocs, \gamma, \mu$)

  $solver$ = PULP_CBC_CMD()
  $status \leftarrow prob.solve(solver)$

  $feeds_sorted \leftarrow$ Sort($feeds$, key = $x_f.value$, order = desc)

  $selected \leftarrow \emptyset$; $uncovered \leftarrow req_iocs$
  for all $(f, f_iocs) \in feeds_sorted$ **do**
    $new_req \leftarrow f_iocs \cap uncovered$
    if $new_req \neq \emptyset$ **then** ▷ Select feed if it provides more coverage
      $selected \leftarrow selected \cup \{f\}$
      $uncovered \leftarrow uncovered \setminus new_req$
    end if
    if $uncovered = \emptyset$ **then**
      break
    end if
  end for

  return *selected*
end function

Algorithm 6 Solve LP Relaxation

function DERIVELP($feeds, req_iocs, nonreq_iocs, \gamma, \mu$)
  for all f in $feeds$ **do**
    $x_f \leftarrow$ **LpVariable(cat=LpContinuous, lowBound=0, upBound=1)**
  end for
  for all n in $nonreq_iocs$ **do**
    $z_n \leftarrow$ **LpVariable(cat=LpContinuous, lowBound=0, upBound=1)**
  end for
  for all r in req_iocs **do**
    $t_r \leftarrow$ **LpVariable(cat=LpContinuous, lowBound=0, upBound=1)**
  end for

  $prob =$ **LpProblem**($LpMinimize$) ▷ Initialize a minimization problem

  for all f, f_iocs in $feeds$ **do**
    $f_nonreq \leftarrow f_iocs \cap nonreq_iocs$
    for all n in f_nonreq **do**
      $prob = prob + z_n \geq x_f$ ▷ If feed f is selected, IoC n is included
    end for
  end for
  for all r in req_iocs **do**
    $covers \leftarrow \emptyset$
    for all f, f_iocs in $feeds$ **do**
      if $r \in f_iocs$ **then**
        $covers \leftarrow covers \cup \{x_f\}$
      end if
    end for
    $prob = prob +$ **lpSum**($covers$) ≥ 1 ▷ Full coverage constraint
    $prob = prob +$ **lpSum**($covers$) $\leq 1 + t_r$ ▷ Anti-splitting constraint
  end for

  for all f, f_iocs in $feeds$ **do**
    $req \leftarrow |f_iocs \cap req_iocs|$
    $nonreq \leftarrow |f_iocs \cap nonreq_iocs|$
    $cost_x_f \leftarrow (1.0/\max(1, req)) \cdot (1.0 + \gamma \cdot nonreq)$ ▷ Calculate Feed Costs
  end for

  $\alpha \leftarrow len(nonreq_iocs) + 1$
  $obj \leftarrow \alpha \cdot \sum_f (cost_x_f \cdot x_f) + \sum_n z_n + \mu \cdot \sum_r t_r$
  $prob = prob + obj$

  return $prob, x_f$
end function

similarly initializes decision variables for feed selection and non-required IoC inclusion, the first alteration from the ILP implementation is the substitution of `LpBinary` variables with `LpContinuous` ones bounded between 0 and 1 (lines 2-10). Since this model still evaluates the inclusion of non-required IoCs (z_n), it also necessitates the exhaustive, upfront collection of all irrelevant indicators across the entire problem instance, which is a computationally heavy prerequisite shared with the exact integer model. To link these variables, the formulation defines the same ILP constraints ($z_n \geq x_f$) in lines 12-17, but without binary variables, these merely ensure that if a fractional feed contributes a non-required IoC, the corresponding z_n variable must be at least as large as that fraction.

Because continuous solvers will attempt to minimize the z_n noise penalty by splitting the selection into tiny fractions across dozens of feeds, the model introduces penalties that attempt to reduce the splitting of feeds and overcoverage of the same IoCs (lines 8-10). For every required IoC, the sum of its covering feeds is bounded by $1 + t_r$, where t_r is a continuous slack variable heavily penalized in the objective function (line 26). In other words, the algorithm is mathematically fined if it tries to "cheat" the system by scraping tiny fractions from many overlapping feeds. If those combined fractions exceed the equivalent of a single whole feed (1.0), the t_r penalty activates and ruins the solver's overall score. This encourages the solver to consolidate its choices by penalizing overcoverage.

Finally, this heuristic relies on a customized trade-off objective function in lines 33-35, pre-computing a density-based cost for each feed ($cost_x_f$) influenced by an undesired IoC penalty (γ) in lines 28-32. With these weights, feeds with more required IoCs will cost less, while more irrelevant indicators will increase cost. This localized cost adjustment is a vital mechanism for the continuous approach; without it, the solver is prone to symmetry, often assigning identical fractional values to overlapping

feeds and flattening the priority scores. By including this density cost into the objective, the solver is mathematically forced to break this symmetry and assign distinctly higher fractional weights (x_f) to highly efficient feeds. This helps ensure that they are strictly prioritized over noisier alternatives during the final selection phase.

After the execution of DeriveLP, Algorithm 5 leverages the same CBC solver as with the ILP and extracts the fractional x_f values assigned to each feed. These values are treated as a global priority score, sorted into descending order to guide greedy feed selection (line 5). Feeds assigned a value of $x_f = 1.0$ by the continuous solver are critical for selection, while feeds assigned lower fractions represent overlapping or noisy intelligence. By iterating through this LP-sorted list and greedily appending feeds to the final selection (*selected*) until the set of required IoCs (*uncovered*) is completely satisfied (lines 6-16), the algorithm constructs a valid, informed solution.

Ultimately, this LP-guided heuristic represents a synthesis of global mathematical optimization and localized selection. By dictating the greedy selection order through the LP solver's evaluation of the entire problem space, the algorithm augments the traditional step-by-step precision of pure greedy methods with a globally informed perspective. The final solution provides an informed feed selection in terms of the objective, with improved scalability relative to the ILP approach. However, despite this improvement, both mathematical programming models inherently suffer from high structural complexity and rely on computationally-expensive upfront knowledge of the threat landscape (specifically regarding non-required IoCs). To address these limitations, the following section introduces an Evolutionary Algorithm, which is specifically designed to offer superior scalability, adaptability, and rapid convergence without the need for external mathematical solvers.

4.3 Evolutionary Algorithms

While exact mathematical models like Integer Linear Programming (ILP) are invaluable for guaranteeing an optimal solution upon convergence, they struggle with the combinatorial explosion inherent to the Set Cover Problem and are therefore unrealistic in most massive operational settings. In contrast, greedy approaches and the LP-guided heuristic offer vital computational scalability, but they do not maintain the same mathematical robustness or accuracy as ILP. To handle problem spaces of scale without sacrificing solution quality, researchers frequently turn to metaheuristics, specifically Evolutionary Algorithms (EAs).

EAs are a class of population-based, stochastic search algorithms inspired by the principles of Darwinian natural selection [13]. Unlike greedy heuristics that deterministically build a single solution step-by-step, EAs maintain a "population" of multiple distinct candidate solutions simultaneously. Over successive generations, this population mathematically evolves to iteratively navigate the complex, exponential problem space.

The success of any metaheuristic fundamentally relies on balancing two competing mathematical forces: exploration, which involves broadly searching new, untested regions of the problem space to avoid becoming trapped in local optima, and exploitation, which necessitates refining and leveraging the best traits of currently known high-quality solutions [26]. In EAs, this search process is formalized through a biological comparison: a candidate solution is represented as a chromosome, its individual decision variables (e.g., the binary selection of specific threat feeds) are its genes, and its mathematical quality is evaluated as its fitness [15].

This evolutionary process is achieved through an iterative loop of biologically-inspired operators applied sequentially [11]:

1. **Evaluation and Selection:** The population is first evaluated against a defined fitness function. Higher-performing chromosomes are assigned a mathematically higher probability of being chosen as "parents" (often using mechanisms like tournament or roulette-wheel selection), ensuring that the most effective feed combinations survive to breed the next generation.

2. **Crossover:** The algorithm pairs these parent solutions to exchange subsets of their genes, producing new child solutions. While this operator also incorporates exploration, it drives exploitation, functioning on the assumption that splicing the genetic material of two highly fit selections may yield an even better, synthesized offspring.

3. **Mutation:** To enforce exploration and maintain genetic diversity, random alterations are introduced to the child chromosomes with a low probability. In a binary problem, this often takes the form of a bit-flip randomly adding or dropping a feed. This acts as a crucial safeguard, preventing the population from prematurely converging to a local optimum before the next generation begins.

While standard EAs excel at finding high-quality solutions in massive spaces, they traditionally evaluate candidates based on a single scalar fitness value. When dealing with inherently conflicting goals, such as minimizing cost while maximizing intelligence precision, standard EAs must artificially combine these metrics into a single weighted score, suffering from the same limitations as standard scalarized mathematical programming.

4.3.1 Non-dominated Sorting Genetic Algorithm II

To properly operationalize a multi-objective metaheuristic approach for the CTI feed selection problem, this research implements the Non-dominated Sorting Genetic Algo-

rithm II (NSGA-II). Originally introduced by Deb et al. [9], NSGA-II fundamentally improves upon classic Evolutionary Algorithms and has established itself as the academic and industry standard for multi-objective optimization.

The core advantage of NSGA-II is that it completely abandons single-objective scalarization. Instead of forcing an organization to quantify the exact mathematical trade-off between intelligence noise and feed cost (e.g., $Score = 0.7 \times \text{Coverage} + 0.3 \times \text{Cost}$), NSGA-II treats each objective completely independently. It achieves this by upgrading the standard EA evaluation phase with three distinct mechanical improvements [9]:

- **Fast Non-dominated Sorting:** Instead of assigning a single fitness score, NSGA-II compares all solutions against each other to find the "Pareto dominant" individuals, solutions that are strictly better than others in at least one objective without being worse in any. It ranks the population into separate mathematical tiers, or fronts, where Front 1 contains all completely non-dominated solutions, Front 2 contains solutions only dominated by Front 1, and so forth.

- **Crowding Distance:** To prevent the algorithm from converging on a single tight cluster of similar solutions, NSGA-II calculates a crowding distance metric for each individual. This metric measures the density of neighboring solutions along the Pareto front. By mathematically favoring individuals located in less crowded regions, the algorithm ensures a broad, evenly spread, and diverse set of final options.

- **Elitism:** To guarantee that high-quality solutions are never accidentally destroyed during crossover or mutation, NSGA-II pools the parent and child generations together before sorting. The best solutions from this combined pool are

preserved for the next generation, guaranteeing that the mathematical frontier strictly improves or remains stable over time.

For this specific CTI research, the multi-objective Set Cover formulation must simultaneously minimize the number of selected feeds and minimize the inclusion of non-required IoCs, all while being subject to a strict 100% required IoC coverage constraint.

While NSGA-II provides superior flexibility, as a metaheuristic, it cannot provide a mathematical guarantee of absolute global optimality. Furthermore, standard implementations of EAs rely on entirely randomized initial populations. In a massive combinatorial space of 2^N possible feed combinations, a purely random starting population is highly unlikely to contain even a single solution that achieves 100% coverage, potentially causing the algorithm to spend hundreds of generations just searching for a valid baseline.

To systematically address and evaluate this limitation, Algorithm 7 is designed to accept different initialization strategies (`init_type`). For this analysis, *Greedy Seeding* injects the fully valid champion solution from the baseline Greedy Set Cover algorithm directly into the first slot of the population array ($P_0[0]$). While this solution may not be optimal, it provides a crucial full-coverage foundation for evolutionary exploration. To prevent premature convergence, the remaining 79 slots are populated using a highly sparse random matrix (10% density), guaranteeing genetic diversity.

More importantly, this seeding strategy establishes a strict mathematical performance guarantee for the EA. Because metaheuristics are stochastic, their final solutions traditionally cannot be bounded by a guaranteed approximation ratio. However, by starting with a valid greedy solution and preserving the best-found candidate across generations through NSGA-II's elitism, the EA is mathematically guaranteed to perform at least as well as the Greedy baseline. Consequently, this metaheuristic

Algorithm 7 Multi-Objective Feed Selection via NSGA-II

function EA($feeds, req_iocs, init_type, pop_size = 80$)
    $N \leftarrow |feeds|$

    Minimize $f_1(x) = \sum_{i=1}^{N} x_i$ ▷ Total Feed Count
    Minimize $f_2(x) =$ NonRequiredIoCs$(x, feeds)$ ▷ Noise
    Subject to $g_1(x) = 1.0 -$ Recall$(x, req_iocs) \leq 0$ ▷ 100% Coverage

    if *init_type* = 'greedy' **then**
        $seed \leftarrow$ GreedySetCover$(feeds, req_iocs)$
        $P_0[0] \leftarrow seed$ ▷ Inject champion solution
        $P_0[1 : pop_size] \leftarrow$ SparseRandom$(density = 0.1)$
    else if *init_type* = 'sparse' **then**
        $P_0 \leftarrow$ BinaryRandomSampling()
    else
        $P_0 \leftarrow$ RandomSampling()
    end if

    $cross \leftarrow$ UniformCrossover$(prob = 0.75)$
    $mut \leftarrow$ BitflipMutation$(prob = 1/N)$

    $\mathcal{F}_{pareto} \leftarrow$ RunNSGA2$(f_1, f_2, g_1, P_0, cross, mut, gens = 1000)$
    return $\mathcal{F}_{pareto}$
end function

inherits the greedy algorithm's worst-case performance bounds for the primary objective. By preserving this seed, NSGA-II guarantees that the final suite of options will always contain at least one solution anchored within a logarithmic factor of the absolute minimum feed count.

Once initialized, every candidate solution, represented as a binary decision array $x \in \{0, 1\}^N$, is evaluated against the objectives. The algorithm aims to minimize the absolute number of feeds (f_1) and minimize noise (f_2), constrained by a strict inequality, $g_1(x) \leq 0$. Rather than outright rejecting invalid combinations, NSGA-II utilizes constraint-domination rules; solutions failing to achieve full coverage are severely penalized and strictly dominated by any feasible combination, allowing the

algorithm to guide the population toward the valid computational space without destroying valuable partial-coverage genetic material.

Implemented via the Python `pymoo` library, the solver operates over 1000 generations. Parent solutions are selected via Tournament Selection and subjected to a Uniform Crossover operator (75% probability). To enforce exploration, a Bitflip Mutation operator is applied with a probability of $1/N$, ensuring a mathematical average of one flipped feed per selection, which introduces enough variance to escape local optima without destroying high-performing configurations.

Ultimately, deploying NSGA-II transitions this threat intelligence optimization from a rigid search for a single answer into a dynamic exploration of operational trade-offs. The resulting output, $\mathcal{F}_{pareto}$, is a Pareto-optimal front of distinct, full-coverage selections derived by comparing the solutions in the final evolutionary population. By fundamentally shifting the algorithmic output from a strict computational mandate to a diverse array of mathematically dominant options, this approach allows SOC analysts to make empirically-backed decisions tailored to their organization's specific financial and operational realities.

4.4 Summary of Algorithms

The Greedy algorithms operate on the principle of making locally optimal choices at each iteration to efficiently approximate a global optimum. While these approaches lack the ability to backtrack or evaluate complex overlaps, they are mathematically guaranteed to find a solution within a logarithmic factor of the absolute optimal set. Because of their negligible computational overhead, linear scalability, and highly transparent selection logic, these heuristics serve as fast, easily interpretable baselines against which the more advanced models can be measured.

On the opposite end of the computational spectrum, the Integer Linear Program-

ming formulation provides an exact mathematical approach. By rigidly defining the problem through lexicographic constraints and binary decision variables, the ILP identifies a global optimum on the Pareto front. However, because the underlying Set Cover Problem is NP-hard, this exactness is operationally unrealistic for large-scale, real-world deployments due to the exponential explosion of processing time. To directly address this scalability limitation, the LP-guided heuristic relaxes the rigid integer constraints, evaluating the problem space in polynomial time using continuous variables. Because the solver's fractional output represents partial subscription of feeds, it is not directly interpretable as a concrete solution. However, by leveraging these highly optimized, continuous weights to inform a greedy selection process, the framework successfully translates the fractional bounds into a computationally feasible approximation of the exact integer model.

Navigating the middle ground between the rigid exactness of the ILP and fast, yet short-sighted evaluation by greedy heuristics, NSGA-II provides a highly adaptable and scalable multi-objective framework. By representing solutions as biological chromosomes and iteratively applying genetic operators, it effectively scales to handle massive combinatorial spaces while treating competing goals as independent, non-dominated objectives. Because the algorithm's initial population is seeded with the full-coverage champion solution from the baseline Greedy Set Cover algorithm and maintained generationally via strict elitism, NSGA-II is mathematically guaranteed to perform at least as well as the standard greedy approach. Then, it uses that greedy baseline as an evolutionary anchor to aggressively improve upon over successive generations.

Ultimately, each of these algorithmic approaches brings distinct structural advantages to the complex task of CTI procurement. While the greedy algorithms prioritize speed, the ILP guarantees exactness, the LP heuristic blends mathematical rigor with

scalability, and NSGA-II dynamically maps solution trade-offs. With these algorithms now thoroughly explained, the following chapter will introduce the experiments used to evaluate each algorithm's operational viability.

Chapter 5: Methodology

To rigorously evaluate the performance, accuracy, and operational limits of the proposed multi-objective feed selection algorithms, this research utilizes a comprehensive experimental framework combining empirical benchmarking and explanatory machine learning. This chapter details the quantitative metrics used to measure feed selection quality alongside fourteen mathematical features extracted to analyze problem space topography. It then describes the suite of synthetic problem instances used to assess scalability and stress-test the solvers under highly controlled environmental conditions. Finally, this chapter establishes the machine learning methodology, detailing the configuration and tuning of the XGBoost classification model deployed to map which structural features influence algorithmic performance.

5.1 Experimental Overview

In order to effectively evaluate the proposed feed selection algorithms, the experimental framework executes every algorithm across the complete suite of problem instances. During each execution, comprehensive telemetry is collected, capturing both the performance evaluation metrics of the algorithm and the underlying structural features of the specific problem instance, both of which are detailed in subsequent sections of this chapter. Because this feed selection problem scales exponentially, calculating the absolute true mathematical optimum is computationally intractable for the larger scale tiers. Therefore, within the context of this evaluation, relative Pareto-dominance is established through the best-performing, non-dominated Pareto frontier discovered across all deployed algorithms, rather than a mathematically guaranteed global optimum.

After defining the necessary metrics and features, the subsequent analysis shifts focus to the evaluation of algorithmic performance, which is systematically divided into two distinct phases. The first phase focuses on empirical deployment and comparative benchmarking across synthetically generated problem instances. In this phase, the framework measures the frequency with which each algorithm successfully converges on an empirically non-dominated solution relative to the other methods. Furthermore, when an algorithm fails to reach the best-known Pareto frontier, the analysis quantifies how "far" it deviated by measuring the average excess feed count and the precision loss compared to the non-dominated selection. This approach provides a granular assessment of both an algorithm's success rate and the relative severity of its failures when navigating complex combinatorial landscapes.

The second phase of analysis transitions from observing performance to mathematically explaining it. Using the information gathered in the first phase, the framework deploys a supervised machine learning classification model to learn the specific conditions under which different methods succeed or fail. Through a variety of features, this classifier attempts to predict whether a given algorithm will achieve relative Pareto-optimality based solely on the structural constraints of the problem space. Finally, by extracting and analyzing the internal feature weights from this trained model, this research attempts to isolate why certain algorithmic methodologies excel in specific environments and which mathematical bottlenecks cause others to deviate from the optimal frontier.

5.2 Evaluation Metrics

Before detailing the different categories of problems and examining the feature analysis, it is necessary to define the quantitative metrics used to evaluate the quality of the selected threat intelligence solutions. The primary and most critical metric

is percent coverage, or recall. In this context, recall represents the proportion of the required IoCs that are successfully included in the selected combination of feeds. Mathematically, it is the size of the intersection between the required IoCs and the total selected IoCs, divided by the total number of required IoCs. For a feed selection to be considered a viable and mathematically valid solution within this framework, it must achieve a strict 100% recall. Any solution failing to meet this threshold is considered infeasible.

To measure the operational efficiency of a fully covered feed selection, the framework evaluates its overall precision. Precision is calculated as the ratio of required IoCs to the total number of unique IoCs (both required and non-required) introduced by the selected feeds. This metric yields a value bounded between 0 and 1, explicitly quantifying the solution's signal-to-noise ratio. A score approaching 0 indicates a highly noisy selection, while a perfect score of 1 denotes a highly precise, maximally efficient selection containing strictly required IoCs with zero extraneous noise.

Alongside precision, the framework tracks the total feed count to evaluate the objective of feed minimization. This integer value simply represents the number of distinct threat intelligence feeds selected by the algorithm. In the context of CTI procurement, this metric correlates to financial expenditure (subscription costs) and the engineering overhead required to maintain data pipelines.

Finally, the framework must quantify optimality across these competing priorities. In a single-objective problem, optimality is straightforward since it merely requires finding the single lowest cost. However, minimizing the total feed count and maximizing precision are inherently conflicting objectives. Selecting a highly dense, precise feed often leaves small coverage gaps that require the purchase of additional feeds, thereby increasing the feed count. Conversely, purchasing massive feeds minimizes the feed count but dramatically degrades precision by introducing massive amounts

of noise.

As defined earlier, this framework navigates this conflict via the concept of Pareto optimality. Rather than seeking a singular "best" answer, optimality is defined through a Pareto front, a mathematically proven set of non-dominated solutions. A feed selection is considered Pareto-optimal if no other valid feed combination exists that can improve one objective (e.g., reduce the total feed count) without actively degrading the other (e.g., lowering precision). Given the computational infeasibility of establishing a proven optimal frontier, evaluating optimality of an algorithm in this research is comparative; all non-dominated solutions found by the deployed algorithms form the relative Pareto front.

5.3 Problem Space Features

To understand not just which algorithms perform best, but why they succeed or fail, this research conducts a feature analysis of the underlying CTI composition. Because the structure of the data directly influences the combinatorial landscape, certain topological traits can make a problem instance inherently harder or easier to optimize. By extracting the following 14 features for each problem instance, this study can analytically determine which structural conditions favor specific algorithms and which conditions cause heuristics to deviate from the relative Pareto frontier.

To begin, problem scale and dimensionality directly dictate the baseline computational complexity for any algorithm. The following features attempt to capture this aspect of the problem instance:

- **Number of Feeds** (`num_feeds`): The total number of available candidate feeds (N) drives the vastness of the search space. This is especially relevant for the Set Cover Problem, which scales at $O(2^N)$.

- **Required, Non-Required, and Total IoCs** (`num_req_iocs`, `num_nonreq_iocs`, `num_total_iocs`): These metrics quantify the universe of elements. A high number of required IoCs amplifies the demand of the 100% recall constraint, while a massive volume of non-required IoCs expands the penalty space, making precision potentially harder to maintain.

Feed composition and variance features evaluate the internal makeup of the candidate feeds, providing insight into the uniformity of the problem instance.

- **Average Feed Length** (`avg_length`) **and Length Standard Deviation** (`stddev_length`): These metrics capture the average number of IoCs per feed. A high standard deviation indicates a highly-varied environment featuring a mix of larger aggregators and highly specialized, smaller feeds. These differentiated feed choices could pose a trade-off between high precision in smaller feeds and wider coverage in larger feeds.

- **Average Feed Precision** (`avg_precision`) **and Precision Standard Deviation** (`stddev_precision`): Precision is defined as the ratio of required IoCs to total IoCs within a feed. This metric directly impacts one of the dual objectives for minimizing noise, so it has potential to impact algorithmic performance. Furthermore, high variance in precision requires the algorithm to carefully balance high-yield but noisy feeds against clean but sparse feeds.

To determine structural overlap and similarity between different feeds, this research evaluates the pairwise intersection of the feeds. To accomplish this, Jaccard Similarity evaluates how much two feeds overlap relative to their combined total size by dividing the number of shared elements (the intersection) by the total number of unique elements across both sets (the union) [20]. Mathematically, for two sets A and

B, the Jaccard Similarity $J(A, B)$ is defined as:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

The resulting value is strictly bounded between 0 and 1. A score of 0 indicates that the sets share absolutely no elements and are thus entirely disjoint, while a score of 1 indicates that the sets are identical. In the context of this research, calculating the Jaccard Similarity between two Cyber Threat Intelligence (CTI) feeds provides a precise mathematical evaluation of their redundancy.

- **Required IoC Jaccard Similarity** (`avg_jaccard_req_iocs`): Measured as the average Jaccard Similarity between candidate feeds, this metric encompasses the overlap in required IoCs between feeds. High required overlap means there are many valid combinations to achieve full coverage. While this flexibility gives algorithms more options, it also expands the number of near-optimal solutions, making the true global optimum potentially harder to isolate.

- **Non-Required IoC Jaccard Similarity** (`avg_jaccard_nonreq_iocs`): This feature captures the average overlap of noise between feeds. If this metric is high, selecting multiple noisy feeds is less heavily penalized, because the redundant noise is only counted once in the final selection.

Finally, topological characteristics map the distinct mathematical bottlenecks within the problem, which directly dictate the maneuverability of the algorithms during the selection process.

- **Matrix Density** (`matrix_density`): Calculated as the ratio of active IoC instances to the total possible cells (feeds $\times$ total IoCs). A dense matrix implies extreme redundancy across the ecosystem, whereas a sparse matrix implies highly distinct feeds covering different threats.

- **Average IoC Frequency** (`avg_ioc_freq`): This metric evaluates how many different candidate feeds contain a single required IoC, on average. High frequency provides algorithmic flexibility, whereas low frequency indicates strict data exclusivity.

- **Essential Feeds** (`num_essential_feeds`): "Essential feeds" correspond to candidate feeds that are the sole providers of at least one required IoC. Mathematically, these feeds are mandatory; they must be included in any valid solution to achieve 100% recall. A high number of essential feeds drastically shrinks the functional search space. However, if algorithms fail to prioritize these feeds early, they may be forced into highly suboptimal choices late in their execution to fulfill the coverage constraint.

- **Subset Feeds** (`num_subset_feeds`): This feature represents the number of feeds whose IoCs are entirely covered by another larger feed in the problem instance. From a strict set cover perspective, subset feeds are mathematically redundant for coverage.

In summary, these 14 extracted features define each problem instance, and these averaged values establish a comprehensive mathematical vocabulary for describing all problem instances as a whole. By quantifying the scale, internal composition, structural redundancy, and strict topological bottlenecks of the search space, this framework strictly quantifies the defining characteristics of a problem instance. These metrics serve two critical roles in the subsequent phases of this research: first, as the quantifiable parameters for the generated structural traps described in the following section, and ultimately, as the features for the predictive machine learning model deployed to map algorithmic success.

5.4 Description of Problem Instances

To rigorously evaluate the performance, reliability, and operational limits of the feed selection algorithms presented in this research, this empirical analysis utilizes a series of computationally generated synthetic problem instances. Constructing synthetic scenarios enables the systematic manipulation of specific parameters and environmental composition to intentionally test different possibilities and evaluate each algorithm in a highly controlled setting.

5.4.1 Baseline Evaluation and Scalability

The baseline testing environment is established using purely randomized feed generation. In these problems, each synthetic feed is populated with a random combination and quantity of required and non-required IoCs. This stochastic approach creates an unbiased baseline to evaluate how effectively the different algorithms navigate generic landscapes without any pre-engineered structural advantages or traps.

To test algorithms' scalability, this random generation is deployed across four distinct size tiers. The first tier consists of small problems, containing 5-18 feeds (average `num_feeds` = 11.37) and 40-80 IoCs (average `num_total_iocs` = 54.54). This scale provides a simple baseline to verify the basic reliability of the heuristic and metaheuristic approaches.

Given the empirical verification of the models for the smaller scale, the other tiers consist of incrementally-larger randomized problem instances designed to serve as scalability stress tests. Specifically, the mid-sized tier contains 60-100 feeds (average `num_feeds` = 80.87) and 80-130 IoCs (average `num_total_iocs` = 104.95); the next tier contains 180-280 feeds (average `num_feeds` = 229.52) and 180-280 IoCs (average `num_total_iocs` = 229.15); and finally, the largest tier contains 280-420 feeds (aver-

age num_feeds = 349.63) and 300-450 IoCs (average num_total_iocs = 374.40). As the number of available feeds and total IoCs deliberately inflates, the solution space grows accordingly. These larger-scale environments are utilized to empirically demonstrate the necessity of scalability across algorithms. 1000 instances are generated for each size tier, and the metrics collected for each instance are averaged to establish a comprehensive structural profile for each size category.

5.4.2 Specialized Feed Compositions

While purely random generation successfully validates baseline scalability and performance, it frequently fails to produce the specific structural anomalies that can occur in real-world CTI collection. To evaluate how the algorithms handle complex structural edge cases, a secondary set of specialized, deliberately challenging problems is introduced.

To begin, the Asymmetric Tradeoff formulation creates a few massive feeds that cover almost all required items but carry debilitating noise penalties, alongside many tiny, zero-noise feeds that collectively guarantee full coverage (across all problems in the dataset: avg_precision = 0.9963 and avg_jaccard_nonreq_iocs = 0.0000). By mathematically subsuming the vast majority of these clean candidate feeds beneath the massive aggregators (average num_subset_feeds = 322.6220), this generator creates a deceptive combinatorial landscape. It is designed to test a solver's ability to resist the immediate gratification of massive feed selection and evaluate complex tradeoffs. The primary pitfall it exposes is short-sighted decision-making where certain approaches may grab the massive feeds for their huge upfront coverage, whereas carefully stitching together many smaller, completely clean feeds instead may yield a mathematically superior final result.

Next, the Highly-Overlapped formulation intentionally saturates the environment

with extreme redundancy by artificially engineering high repetition in both required and non-required IoCs (`avg_jaccard_req_iocs` = 0.6864 and `avg_jaccard_nonreq_iocs` = 0.5412). By clustering candidate feeds into near-duplicate templates that share a massive core of required indicators alongside similar noise penalties, this generator ensures that every required item is covered by a high volume of feeds. This formulation is specifically designed to create severe symmetry. Because there are many near-equivalent ways to cover the same items with virtually identical precision trade-offs, it exposes the potential pitfalls for different algorithms' objectives.

The Nested Subset formulation constructs hierarchical families of feeds where each subsequent feed is a strict mathematical subset of its parent, containing fewer required items but proportionally scaled-down noise. This intentionally floods the combinatorial space with redundant choices (average `num_subset_feeds` = 315.6740). However, by isolating these distinct family lineages from one another (`avg_jaccard_req_iocs` = 0.0135), this generator ensures that the structural redundancy is strictly localized to vertical, independent hierarchies. In this way, this formulation specifically tests redundancy recognition and pruning efficiency. If an algorithm cannot recognize how these nested subsets relate to one another, it will become trapped in a maze of nearly identical choices, wasting vast amounts of time evaluating thousands of redundant combinations without ever actually improving the final objective score.

By partitioning required items and noise into completely isolated, mutually exclusive buckets, the Orthogonal Disjoint formulation forces extreme sparsity across the combinatorial space (`avg_jaccard_req_iocs` = 0.0259, `avg_jaccard_nonreq_iocs` = 0.0050, and average `matrix_density` = 0.0209). This generator builds a highly fragmented ecosystem consisting of tiny, specialized feeds (`avg_length` = 7.4619, `stddev_length` = 1.0433) that ensure almost zero cross-contamination between distinct threat families. Rather than being specifically challenging, it serves as a critical

baseline test for structural awareness. It exposes the pitfall of over-complication; a capable solver should instantly recognize that the overarching puzzle is actually just several tiny, disconnected problems and solve them independently, whereas a blind approach will waste time trying to untangle relationships and boundaries that do not actually exist.

Finally, the Mandatory Penalty Formulation isolates a tiny handful of required IoCs into a single feed (`num_essential_feeds` = 1) that is simultaneously loaded with a massive, unavoidable group of noise. By placing this anomalous feed alongside a highly flexible pool of standard-sized options (`stddev_length` = 105.3561 and `avg_ioc_freq` = 139.8223), the generator ensures that while the broader combinatorial landscape offers abundant coverage alternatives, the algorithm must still absorb this singular, devastating mathematical hit to achieve feasibility. Designed to test constraint satisfaction under an extreme, localized penalty, this approach exposes less robust algorithms that will successfully select the mandatory feed to satisfy the strict recall constraint, but the final solution may be completely derailed.

These specialized formulations serve as a rigorous stress-testing framework designed to complement standard randomized problem generation. While purely stochastic generation is effective for establishing an algorithm's baseline performance across average-case scenarios, it rarely produces the highly structured, challenging topographies required to expose deep logical flaws. Pairing randomized problems with these deliberately engineered edge cases that guarantee asymmetric tradeoffs, high redundancy, strict subsets, structural disjointness, and extreme localized penalties creates a comprehensive evaluation methodology. This dual approach ensures that optimization solvers are proven robust not just against generalized noise, but against the specific architectural traps that can cause heuristic failures and algorithmic bottlenecks.

5.5 Classification Models

To mathematically determine the environmental conditions that influence algorithmic success, this research formulates a supervised binary classification problem and analysis of model feature weights. By using each algorithm's optimality flag as the target variable and the fourteen previously defined features as independent predictor variables, a distinct classification model is trained for each algorithm. To reiterate, the "optimality" flag in this context does not define true mathematical optimization, but refers to a solution that is non-dominated by all other algorithms' solutions for that problem. By evaluating each algorithm independently, this research seeks to isolate the vulnerabilities that drive heuristic failure across the various methodologies.

This classification analysis leverages Extreme Gradient Boosting (XGBoost) to balance structural interpretability with high-dimensional predictive strength. XGBoost was explicitly chosen for its state-of-the-art predictive performance on structured tabular data and its ability to capture non-linear dependencies and multi-feature interactions. Because algorithmic performance within a complex search space rarely scales perfectly linearly with dataset properties, XGBoost's capacity to map complex relationships ensures the highest possible classification accuracy when analyzing the structural topography of highly varied problem instances. Prior to model training, all synthetic problem instances with their corresponding performance metrics and structural features are processed into a singular, unified dataset (given 1000 instances for each category, which accumulates into 9000 rows for this combined dataset).

Because algorithmic performance varies drastically depending on the scaling tier and formulation, the resulting datasets frequently exhibit significant class imbalances. Consequently, hyperparameter optimization is conducted independently for each algorithm utilizing a randomized search approach (RandomizedSearchCV). The tuning

process relies on Stratified Cross-Validation to preserve the underlying optimal-to-suboptimal class distributions across all training folds. Furthermore, models are evaluated and selected based on the Receiver Operating Characteristic Area Under the Curve (ROC-AUC) to supplement raw accuracy. ROC-AUC provides a far more robust metric for imbalanced domains, precisely measuring a model's ability to rank successful versus unsuccessful instances.

Finally, once the optimized XGBoost models are evaluated against the held-out test data, the framework extracts the underlying predictive logic. This is accomplished by calculating permutation importance and SHapley Additive exPlanations (SHAP) values. Permutation importance evaluates the global significance of each feature by measuring the decrease in model accuracy when a given feature's data is randomly shuffled. This highlights which structural characteristics the model relies on most heavily to make accurate predictions. Furthermore, SHAP reveals not only the magnitude of a feature's influence but also its directional impact. Specifically, it can demonstrate whether an increase in a given metric actively promotes or hinders a specific algorithm's success. Together, this multi-layered extraction maps which structural features impact algorithmic performance, thus transforming predictive machine learning into a tool for explanatory mathematical analysis.

5.6 Chapter Overview

This chapter established the comprehensive methodology required to rigorously evaluate and analyze the multi-objective feed selection solvers. By defining quantitative evaluation metrics based on recall, precision, and feed count, and extracting 14 distinct structural features, the framework established a precise vocabulary for measuring both selection quality and problem complexity. This chapter detailed the construction of diversified synthetic problems, pairing randomized scalability tiers

with deliberately challenging formulations designed to stress-test algorithms. Finally, the machine learning methodology was defined, explaining how the XGBoost model will be trained and interpreted using permutation importance values and SHAP to mathematically map which architectural traits dictate algorithmic success. Collectively, this evaluation framework not only includes empirical performance findings but also provides an explanatory analysis using XGBoost classification.

Chapter 6: Results and Analysis

This chapter presents the comprehensive performance analysis of the proposed CTI curation algorithms. To rigorously evaluate how effectively each methodology balances feed minimization and precision maximization, the analysis proceeds in two phases. The first establishes baseline scalability and structural resilience through deployment against randomized and specifically-engineered problems. The second uses machine learning classification models and feature weight analysis to identify the problem space characteristics that positively or negatively influence the performance of each individual algorithm. The final section synthesizes these empirical and analytical insights to define the operational limits and ideal use cases for each methodology.

6.1 Empirical Analysis

Before investigating the influence of underlying features of the problem space, it is necessary to establish how these algorithms perform in simulated deployment scenarios. This section first examines randomized problems to test computational scalability and establish baseline algorithmic behavior and then specialized problems engineered with deliberate structural traps to stress-test resilience against edge cases.

6.1.1 Evaluation on Randomized Problems

Before analyzing algorithm performance against deliberately engineered structural traps, it is necessary to establish a performance baseline. By populating feeds stochastically, these environments provide an unbiased testing ground. Thus, randomized problem instances allow this framework to explicitly test computational scalability

and establish how well each algorithm organically balances the competing objectives of feed minimization and precision maximization in a completely stochastic environment.

To evaluate this scalability, the empirical framework deploys the proposed algorithms across four incrementally increasing size tiers, defined by the available number of CTI feeds and total IoCs.

Table 6.1: Scalability Tiers for CTI Feeds and IoCs

Size	**Feeds**	**Total IoCs**
Small	$5 - 18$	$40 - 80$
Medium	$60 - 100$	$80 - 130$
Large	$180 - 280$	$180 - 280$
XL (Largest)	$280 - 420$	$300 - 450$

The primary metric for success is the percentage of problems that an algorithm converges on the Pareto frontier of solutions. In this study, the Pareto front is constructed relatively from the pool of all algorithms' solutions and is confined to all non-dominated feed combinations. To reiterate, the Pareto-dominant solutions in this analysis are not guaranteed to be mathematically optimal, but they are the best performing solutions produced by the implemented methodologies. With this definition established, Table 6.2 details the overall performance across the 4,000 total randomized executions at varying magnitudes and demonstrates a clear hierarchy in algorithmic scalability.

At the smallest tier (5-18 feeds, 40-80 IoCs), algorithmic performance is generally highest. Most notably, Integer Linear Programming (ILP) achieves a perfect 100% convergence rate and the NSGA-II evolutionary algorithm achieves a near-perfect 99.9% success rate. Even Greedy Set Cover and the Linear Programming Guided Heuristic discover the Pareto frontier in over two-thirds of instances, whereas Ratio

Table 6.2: Percentage of "Best" Solutions Discovered by Problem Size

Algorithm	**Problem Size (Number of Feeds)**				**Overall**
	Small (5–18)	**Medium** (60–100)	**Large** (180–280)	**XL** (280–420)	
Greedy Set Cover	67.6%	23.7%	44.6%	63.8%	49.9%
Ratio Greedy	53.6%	49.6%	66.6%	67.0%	59.2%
Additive-Penalty Greedy	57.5%	38.6%	50.2%	59.2%	51.4%
ILP	**100.0%**	**81.1%**	33.9%	3.3%	54.6%
LP-Guided	70.7%	28.6%	48.0%	56.9%	51.0%
NSGA-II	99.9%	72.9%	**85.0%**	**98.3%**	**89.0%**
Total Problems	*1000*	*1000*	*1000*	*1000*	*4000*

and Additive-Penalty Greedy post the lowest convergence rates. When combinatorial complexity is kept low, performance improves across all algorithms, while ILP and NSGA-II distinguish themselves as the clear champions.

In addition to establishing a baseline of algorithmic performance, the small tier serves as a critical verification step for the fundamental logic of all deployed algorithms. Because the $O(2^N)$ search space is sufficiently restricted at this scale, it is possible to compare heuristic outputs against a true, mathematically guaranteed Pareto-optimal front derived from an exhaustive search through all possible combinations. These exact baseline results yielded identical performance for each algorithm compared to the previous relative analysis, which shows that in small-scale environments, the success rates indicate algorithmic convergence on the true, Pareto-optimal frontier. Thus, it is reasonable to use non-dominated solutions as a proxy for the true global optimum when evaluating algorithmic success rates. While this assumption provides a fundamental starting point for analysis, its reliability may decrease as problem sizes scale, which remains a caveat that must be explicitly accounted for when interpreting the results of larger-scale experiments.

By establishing the true Pareto frontier via exhaustive search, this analysis pro-

vides a definitive baseline to empirically validate the comprehensiveness of the NSGA-II solution set. Specifically, for this small-tier dataset, NSGA-II recovers an impressive 98.8% of all Pareto frontier solutions, and its average solution set size of 1.5 closely mirrors the average true Pareto-optimal frontier size of 1.54. This suggests that NSGA-II not only reliably converges to the Pareto frontier, but it also finds the majority of all solutions on the true Pareto front established through exhaustive search. Although ILP may boast higher performance rates, the ability of NSGA-II to provide multiple possible solutions is a unique advantage compared to other algorithms, as NSGA-II can illustrate the different trade-offs. Though we cannot assume this behavior will remain consistent at scale, it serves as a strong preliminary indicator of the algorithm's potential.

As the problem size increases into the Medium (60-100 feeds, 80-130 IoCs) and Large (180-280 feeds, 180-280 IoCs) tiers in Table 6.2, the increased solution space forces immediate divergence between the methodologies. While Ratio and Additive-Penalty Greedy maintain relatively consistent, moderate performance across all tiers, ILP drops noticeably from its perfect baseline, performing highest in only 81.1% of medium-sized problem instances and collapsing to 33.9% in large instances due to time limit restrictions. Greedy Set Cover experiences a sharp decline in the medium tier (23.7%) before rebounding in the larger environments. In contrast to ILP, NSGA-II maintains stability, successfully discovering the Pareto frontier in 72.9% of medium-scale problem instances and improving to 85.0% in large instances. The performance rebounds from Medium to Large problems across the heuristic algorithms can be partially attributed to the relative definition of success; since ILP performance notably drops, the other methods seem to improve by comparison. Nevertheless, these success rates highlight the scalability of heuristics compared to the rigorous ILP.

Finally, the transition to the largest (XL) tier (280-420 feeds, 300-450 IoCs) ex-

poses the operational limit of exact mathematical solvers. While ILP serves as the gold standard for small-scale operations, its performance rate collapses to 3.3% on the largest problems because its underlying solver's branch-and-bound tree becomes too massive to resolve within the time limit. Additionally, while all other methods guarantee full coverage, this method will break the coverage constraint under a restrictive time limit, which occurs in the largest two datasets (Large and XL) in a small but noteworthy 0.1% of cases as shown in Table 6.3.

Table 6.3: ILP Full Recall Achievement by Problem Size

Dataset	**ILP Full Recall**
Small	100.0%
Medium	100.0%
Large	99.9%
XL	99.9%

By contrast, the XL tier highlights the dominance of metaheuristics. NSGA-II scales effectively, dominating the XL tier with a 98.3% convergence rate, while even the simpler heuristic methods outperform ILP.

While this binary analysis of success or failure in finding the highest solution is helpful for establishing baseline performance, these convergence rates do not capture the full picture. To mitigate this fact, this research quantifies the severity of failure by calculating on average how many excess feeds an algorithm selected and how much precision it sacrificed compared to the closest best-performing solution. In Tables 6.4 and 6.5, the data exposes distinct algorithmic tradeoffs.

Interestingly, the Ratio Greedy algorithm, which mathematically prioritizes noise reduction, boasts the lowest overall average precision loss (0.03) in Table 6.5, but it pays the price in feed over-consumption, averaging 1.71 unnecessary feeds per run and selecting up to 9 redundant feeds in extreme cases. In contrast, Greedy Set Cover minimizes feed counts aggressively (averaging only 0.24 extra feeds when it fails) but

Table 6.4: Suboptimality Severity: Excess Feeds Selected by Problem Size

Algorithm	**Problem Size (Number of Feeds)**				**Overall**
	Small (5–18)	**Medium** (60–100)	**Large** (180–280)	**XL** (280–420)	
Greedy Set Cover	0.33 (3)	0.31 (4)	**0.16 (2)**	**0.12** (2)	**0.24** (4)
Ratio Greedy	1.20 (5)	1.96 (9)	2.01 (8)	1.80 (8)	1.71 (9)
Additive-Penalty Greedy	0.91 (3)	1.34 (8)	1.36 (7)	0.88 (7)	1.15 (8)
ILP	–	**0.04 (1)**	0.85 (7)	1.43 (8)	1.10 (8)
LP-Guided	0.55 (3)	0.98 (4)	1.17 (7)	0.96 (6)	0.96 (7)
NSGA-II	**0.00 (0)**	0.39 (3)	0.40 **(2)**	0.18 **(1)**	0.39 **(3)**

Note: Values represent Average (Maximum) extra feeds when the algorithm fails to reach the Pareto frontier.

Table 6.5: Suboptimality Severity: Precision Degradation by Problem Size

Algorithm	**Problem Size (Number of Feeds)**				**Overall**
	Small (5–18)	**Medium** (60–100)	**Large** (180–280)	**XL** (280–420)	
Greedy Set Cover	0.04 (0.21)	0.08 (0.38)	0.08 (0.48)	0.05 (0.43)	0.07 (0.48)
Ratio Greedy	**0.01** (0.18)	**0.03 (0.19)**	**0.03 (0.20)**	**0.04 (0.22)**	**0.03 (0.22)**
Additive-Penalty Greedy	0.02 (0.17)	0.04 (0.27)	0.06 (0.29)	0.07 (0.28)	0.04 (0.29)
ILP	–	0.05 (0.30)	0.14 (0.49)	0.18 (0.58)	0.15 (0.58)
LP-Guided	0.05 (0.18)	0.06 (0.29)	0.05 (0.32)	0.05 **(0.22)**	0.05 (0.32)
NSGA-II	0.04 **(0.04)**	0.05 (0.29)	0.08 (0.35)	0.15 (0.30)	0.06 (0.35)

Note: Values represent Average (Maximum) precision loss when the algorithm fails to reach the Pareto frontier.

can suffer massive noise penalties, with maximum precision drops of up to 48%. ILP's scalability failure is again highlighted here; when it times out on massive problems, it returns highly unoptimized solutions, exhibiting the highest average and maximum precision loss of any methodology. Despite these cases, most approaches still consume only 1–2 unnecessary feeds and accumulate less than 10% precision loss on average when they do not find a non-dominated solution.

For the final analysis of randomized problems, an evaluation of the set of solutions provided by NSGA-II highlights the operational advantage of evolutionary algorithms, as shown in Table 6.6. Unlike deterministic approaches that return a singular solution, NSGA-II reveals different trade-offs and generates a set of 1.46 to 3.00 non-dominated solutions on average across all size tiers. For both successful and suboptimal solution sets, NSGA-II ensures that security analysts are not locked into a singular decision but are instead presented with an array of highly optimized options, allowing them to make the final determination between feed minimization and precision tolerance based on operational budgets.

Table 6.6: NSGA-II Average Frontier Size by Dataset Tier

Dataset Tier	**Success**	**Failure**
Small	1.46	3.00
Medium	2.74	2.88
Large	2.51	2.34
XL	2.33	1.53

In summary, the evaluation across randomized datasets establishes a definitive baseline for algorithmic scalability and behavior. While exact solvers like ILP guarantee mathematical optimality on small-scale problems, they suffer catastrophic computational failures as the combinatorial search space expands. Conversely, the NSGA-II evolutionary algorithm demonstrates exceptional scalability, consistently delivering non-dominated, multi-solution Pareto frontiers even at the largest levels, while the

heuristic methods reveal operational trade-offs between feed minimization and precision retention. The following subsection introduces specialized datasets consisting of problems engineered with structural traps to stress-test the algorithms against challenging edge cases found in real-world threat intelligence curation.

6.1.2 Evaluation on Specialized Problems

To supplement the scalability and baseline performance analysis in the previous section, this evaluation tests five structurally challenging problem instances. While these topologies are described thoroughly in Chapter 5, a brief reiteration of their makeup is detailed below:

- **Orthogonal Disjoint:** these problems contain several smaller completely disconnected problems, such that required IoCs exist exclusively in separate pools of feeds.
- **Highly-Overlapped:** intentional redundancy of both required and non-required IoCs is injected into feeds to ensure they initially appear very similar mathematically.
- **Mandatory Penalty:** this formulation includes one very noisy "penalty" feed that must be ingested to achieve full coverage, but at a massive detriment to overall precision.
- **Nested Subset:** these problems are defined by hierarchical families of feeds, in which the largest feeds hold the most required and non-required IoC information and each subset feed includes incrementally less coverage but also reduced noise.
- **Asymmetric Tradeoff:** to directly address the trade-off between noise and coverage, this formulation introduces a few highly-noisy, but massive feeds and

many tiny feeds with high precision. Taken together, the tiny feeds can achieve full coverage at an improved precision compared to the massive feeds, whereas the massive feeds together cannot achieve full coverage.

In this analysis, the primary metric for success remains the percentage of problem instances that an algorithm converges to one of the highest-performing solutions. As observed with the randomized datasets, algorithmic success is not defined by achieving absolute mathematical optimality, though many solutions naturally reach that threshold. Instead, success is evaluated comparatively, encompassing all non-dominated solutions produced by the competing algorithms. However, unlike the generalized scaling test with randomized datasets, this evaluation explicitly aims to trigger algorithmic failure states. The overall performance across the 5,000 total specialized executions reveals that algorithms which performed well in generalized scenarios often possess crippling vulnerabilities to specific structural traps, as detailed in Table 6.7.

Table 6.7: Percentage of "Best" Solutions Discovered by Problem Characteristic

	Problem Characteristic					
Algorithm	**Disjoint**	**Overlap**	**Penalty**	**Subsets**	**Asymmetric**	**Overall**
Greedy Set Cover	93.3%	95.8%	15.9%	8.8%	84.0%	59.6%
Ratio Greedy	86.9%	0.1%	18.7%	54.3%	**100.0%**	52.0%
Additive-Penalty Greedy	98.4%	8.0%	21.8%	58.7%	99.8%	57.3%
ILP	**100.0%**	0.0%	**97.7%**	**99.8%**	99.8%	**79.5%**
LP-Guided	82.9%	0.1%	41.1%	23.3%	38.9%	37.3%
NSGA-II	93.7%	**97.6%**	39.1%	17.8%	97.9%	69.2%
Total Problems	*1000*	*1000*	*1000*	*1000*	*1000*	*5000*

In environments with distinct, cleanly separated data (Disjoint) or highly skewed feed sizes (Asymmetric), algorithmic performance is universally high. Integer Linear Programming (ILP) achieves a perfect 100.0% convergence rate on Disjoint data, while Ratio Greedy achieves 100.0% success on Asymmetric data, and even simpler heuristics such as Additive-Penalty Greedy (98.4% Disjoint, 99.8% Asymmetric) easily

identify the optimal paths. While Ratio and Additive-Penalty Greedy outperforming ILP in Asymmetric Trade-off scenarios may at first seem surprising, these algorithms maximize a coverage-noise objective, so they are specifically optimized to navigate these trade-offs. The notable exception is the LP-guided heuristic's underperformance on the Asymmetric dataset, likely because the linear relaxation mechanism assigns large fractional values to massive, noisy feeds in an attempt to gain coverage and avoid the splitting penalties associated with choosing multiple feeds.

However, the introduction of the Highly-Overlapped problems completely shatters this consistency across algorithms. Problems engineered with heavy redundancy across required and non-required IoCs expose a devastating computational flaw in mathematically exact methods. ILP completely collapses to a 0.0% success rate, and the LP-guided heuristic falls to 0.1%; in heavily overlapping problem instances, multiple feed combinations can achieve the same mathematical coverage, creating similar fractional outputs for the continuous approach and an extremely dense branch-and-bound tree that forces ILP to time out without verifying optimality. In fact, in 59.7% of these Highly-Overlapped instances, ILP fails to find a full coverage solution. Furthermore, Ratio and Additive-Penalty Greedy sink to 0.1% and 8.0%, respectively, because overlapping feeds yield nearly identical objective scores and derail local decisions. On the other hand, Greedy Set Cover (95.8%) and NSGA-II (97.6%) thrive in this environment, which reveals the benefit of leveraging Greedy Set Cover as an initial seed for the NSGA-II approach.

Alternatively, the Penalty and Subsets characteristics expose the fundamental weaknesses of short-sighted heuristic approaches. For feeds that offer massive immediate coverage but are bundled with devastating noise (Penalty) or bypass more efficient, smaller feed combinations (Subsets), the basic Greedy Set Cover algorithm plummets to 15.9% and 8.8% success, respectively, because it blindly follows immediate marginal

gain into these structural traps. It is precisely within these deceptive Penalty and Subset environments that ILP reclaims its status as the gold standard, achieving 97.7% and 99.8% convergence rates. Notably, these problem instances fall into the XL size tier, so ILP actually performs more efficiently in structurally-constrained scenarios. NSGA-II also demonstrates vulnerabilities here that were not apparent in the randomized trials, dropping to 39.1% on Penalty and 17.8% on Subsets because its Greedy Set Cover seed inherits those initial blind spots.

Just as with the randomized dataset analysis, evaluation of suboptimality measured by excess selected feeds and degraded precision reveals distinct algorithmic tradeoffs in Tables 6.8 and 6.9. The most striking observation is the behavior within the Overlapped dataset. In highly redundant environments, failure manifests strictly as feed over-consumption; every algorithm registers an average precision loss of precisely 0.00 in Overlapped problems, yet they each accumulate massive excesses in feed volume. ILP averages 9.38 unnecessary feeds when it fails in this environment, while Ratio Greedy follows closely at 7.49. This proves a substantial weakness for Ratio Greedy across datasets, averaging 5.50 extra feeds per failed problem.

Table 6.8: Suboptimality Severity: Excess Feeds Selected by Problem Characteristic

	Problem Characteristic					
Algorithm	**Disjoint**	**Overlapped**	**Penalty**	**Subsets**	**Asymmetric**	**Overall**
Greedy Set Cover	**0.00 (0)**	**1.12 (2)**	1.00 (5)	1.00 (4)	**0.17 (1)**	**0.90 (5)**
Ratio Greedy	2.37 (6)	7.49 (15)	4.77 (19)	3.32 (21)	–	5.50 (21)
Additive-Penalty Greedy	1.00 (3)	3.41 (13)	3.18 (13)	2.67 (20)	1.00 (1)	3.16 (20)
ILP	–	9.38 (16)	**0.05 (1)**	**0.00 (0)**	–	8.88 (16)
LP-Guided	0.12 (1)	6.89 (14)	2.72 (11)	0.72 (10)	0.66 (2)	3.02 (14)
NSGA-II	**0.00 (0)**	1.21 (2)	1.14 (5)	0.82 (28)	0.29 (1)	0.91 (28)

Note: Values represent Average (Maximum) extra feeds when the algorithm fails to reach the Pareto frontier.

Conversely, the Subsets characteristic represents the most dangerous trap for precision degradation. While algorithms like Greedy Set Cover and NSGA-II are highly efficient at minimizing overall feed bloat, their failures in the Subsets environment are

Table 6.9: Suboptimality Severity: Precision Degradation by Problem Characteristic

	Problem Characteristic					
Algorithm	**Disjoint**	**Overlapped**	**Penalty**	**Subsets**	**Asymmetric**	**Overall**
Greedy Set Cover	0.01 (0.02)	**0.00 (0.00)**	0.00 (0.02)	0.12 (0.26)	0.01 (0.02)	0.06 (0.26)
Ratio Greedy	**0.00 (0.01)**	**0.00 (0.00)**	0.00 (0.01)	0.02 (0.19)	–	0.00 (0.19)
Additive-Penalty Greedy	**0.00 (0.01)**	**0.00 (0.00)**	0.00 (0.01)	0.02 (0.19)	**0.00 (0.00)**	0.01 (0.19)
ILP	–	**0.00 (0.00)**	**0.00 (0.00)**	**0.01 (0.03)**	–	**0.00 (0.03)**
LP-Guided	0.01 (0.03)	**0.00 (0.00)**	0.00 (0.01)	0.08 (0.27)	0.01 (0.03)	0.02 (0.27)
NSGA-II	0.01 (0.02)	**0.00 (0.00)**	0.00 (0.02)	0.11 (0.25)	0.01 (0.02)	0.06 (0.25)

Note: Values represent Average (Maximum) precision loss when the algorithm fails to reach the Pareto frontier.

severe, with average precision decline of 0.12 and 0.11 and maximum drops reaching 0.26 and 0.25, respectively. ILP, however, demonstrates remarkable damage control against these specific structural traps; even when ILP fails to converge to the relative Pareto front in Penalty or Subsets environments, its suboptimality is negligible in both excess feeds and precision loss.

Despite the robustness of exact solvers like ILP in these challenging environments, all deterministic methods remain limited to returning a single, rigid solution. In contrast, while NSGA-II may occasionally fall into heuristic traps, it mitigates this vulnerability through its inherent ability to generate a diverse Pareto frontier of non-dominated solutions. Table 6.10 quantifies this benefit across each specialized characteristic.

Table 6.10: NSGA-II Average Frontier Size by Dataset

Dataset Characteristic	**Success**	**Failure**
Disjoint	1.24	1.24
Overlapped	1.93	2.83
Penalty	5.74	9.42
Subsets	3.35	2.30
Asymmetric	7.66	10.81

Ultimately, this targeted evaluation demonstrates that no single algorithm is universally dominant. ILP is the undisputed champion of navigating deceptive noise and

subset hierarchies, provided the data is not overly redundant. However, in heavily overlapping environments, NSGA-II distinguishes itself as the ideal approach. Although the evaluations across both randomized and specialized datasets yield valuable empirical insights into algorithmic behavior, these findings remain observational. To more comprehensively understand the conditions required for algorithmic success, a formal analysis of the problem space is useful.

6.2 Problem Space Analysis

While the empirical dataset evaluations indicate certain trends and failure states, these results remain fundamentally observational. To concretize the previous scalability and performance analysis, this section conducts a formal problem space evaluation. By treating algorithmic success as a target variable and 14 dataset characteristics as predictive features, this study uses machine learning classification models to identify the conditions that positively or negatively impact algorithmic performance.

6.2.1 Greedy Set Cover

For structural conditions that most impact Greedy Set Cover performance, Table 6.11 highlights the two most heavily weighted features by permutation importance: Average Jaccard Similarity of Required IoCs and the Number of Essential Feeds. As a supplement to this permutation importance, the SHAP plot in Figure 6.1 reveals that a high number of essential feeds (indicated by the red and purple scatter points) actively hurts algorithmic performance, as all red points fall on the negative side of the plot. Because this greedy heuristic inherently relies on maximizing immediate coverage at each step, dictating certain feed selections prevents raw coverage from being the primary determinant of its path, forcing it into suboptimal routes.

Table 6.11: XGBoost Feature Importance for Greedy Set Cover

Feature	**Permutation Importance (Mean ± Std)**
Average Jaccard Similarity (Required IoCs)	0.1240 ± 0.0040
Number of Essential Feeds	0.0437 ± 0.0026
Standard Deviation Precision	0.0417 ± 0.0030
Average Jaccard Similarity (Non-Required IoCs)	0.0310 ± 0.0019
Number of Feeds	0.0300 ± 0.0032
Average Precision	0.0136 ± 0.0010
Average Length	0.0108 ± 0.0017
Number of Subset Feeds	0.0091 ± 0.0017
Standard Deviation Feed Length	0.0058 ± 0.0012
Number of Required IoCs	0.0034 ± 0.0012
Average IoC Frequency	0.0025 ± 0.0010
Matrix Density	0.0023 ± 0.0013
Total Number of IoCs	0.0007 ± 0.0008
Number of Non-Required IoCs	0.0000 ± 0.0004

Model Performance: Accuracy = 0.80, ROC-AUC = 0.885

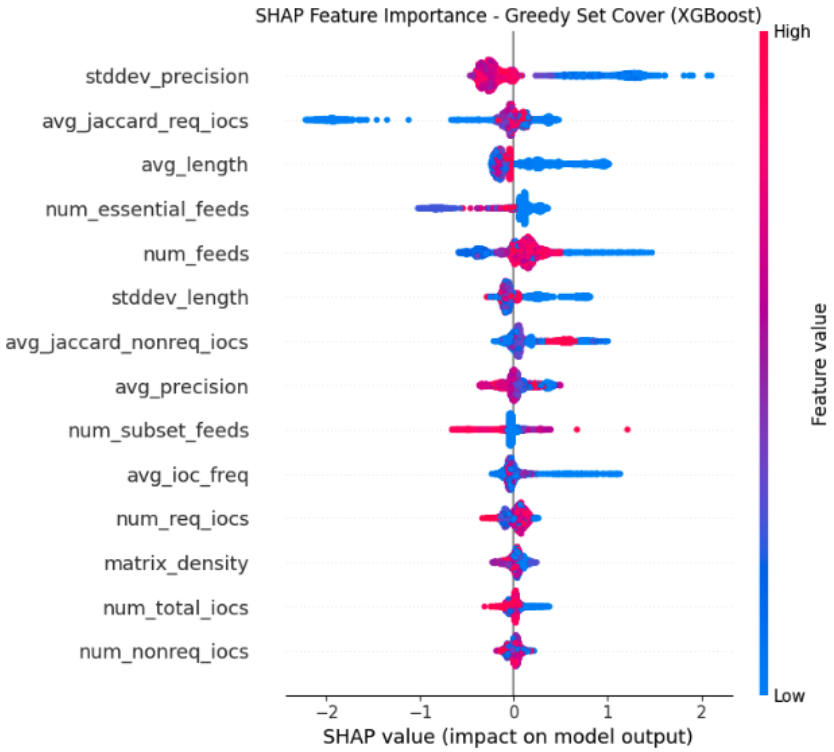

Figure 6.1: Greedy Set Cover: Feature Weights

Model Performance: Accuracy = 0.80, ROC-AUC = 0.885

The vulnerability of relying on purely maximizing coverage is further reflected in the dataset's average structural overlap. A low Jaccard similarity, indicating low overlap, can penalize the algorithm because its raw coverage approach may fail to quickly identify the specific feeds required to complete the set. At the same time, the previous empirical analysis shows that the algorithm performs exceptionally well on both the Highly-Overlapped (95.8%) and Orthogonal Disjoint datasets (93.3%), revealing that overlap in required IoCs strongly influences the approach but does so with important nuance.

In contrast to the mixed visual distribution of structural overlap, feed precision variance retains a more interpretable signal. The third most impactful feature in terms of permutation importance, the Standard Deviation of Precision, distinctly highlights how varying feed quality destabilizes the algorithm. This distinction is visually prominent in Figure 6.1 because it forms a clear, unambiguous divide, where high standard deviation (red dots) drives the model negatively, and low standard deviation (blue dots) pushes the model positively.

This clear divide tracks with the underlying mechanics of the heuristic, as Greedy Set Cover fundamentally prioritizes raw coverage first and only considers precision as a secondary tie-breaker. Consequently, if a problem instance contains feeds that are vastly different in their precision, the algorithm will predictably select massive sets early in its execution, regardless of precision. This vulnerability is supported by empirical results: for example, the algorithm suffers a catastrophic drop to a 15.9% success rate for problems within the Mandatory Penalty dataset, which contain high precision variance.

Interestingly, while the feature for the Number of Subset Feeds is not heavily weighted in the global permutation importance, the performance on the Nested Subset dataset was worst among all specialized datasets, achieving a mere 8.8% success rate.

This suggests that the failure is not inherently due to the presence of subset feeds, but to the extreme variance in precision that these subsets naturally introduce.

Equally informative are the features that fail to influence the model. Characteristics detailing the sheer size of the problem, such as Total Number of IoCs, Number of Non-Required IoCs, Number of Required IoCs, and Matrix Density, rank lowest in terms of permutation importance. This supports the idea that the performance of the Greedy Set Cover is largely independent of scale. Ultimately, the Greedy Set Cover is a highly situational approach that thrives in scale-invariant problem spaces characterized by consistent feed precision and clear, unconstrained paths to full coverage.

6.2.2 Ratio Greedy

Influencing the success of the Ratio Greedy algorithm, average feed precision and the overlap of non-required IoCs maintain relatively high permutation importances in Table 6.12. Revealing directional nuance, the SHAP summary plot in Figure 6.2 illustrates a distinct impact: a low Average Jaccard Similarity of Non-Required IoCs and high Average Precision strongly drive model performance in a positive direction; conversely, high non-required overlap and low precision push the model negatively. Because the algorithm strictly optimizes for the ratio of required to non-required IoCs, high-precision feeds with fewer non-required IoCs inherently benefit its objective, ensuring that coverage will not be outweighed by noise penalties.

These underlying feature importances align with the algorithm's highly polarized performance across the synthetic datasets. The SHAP summary plot visually reinforces why the algorithm completely fails on the Highly-Overlapped dataset (0.1% success) and struggles with Nested Subsets (54.3%). Specifically, the plot illustrates a strong negative directional push driven by a high Average Jaccard Similarity of

Table 6.12: XGBoost Feature Importance for Ratio Greedy Algorithm

Feature	**Permutation Importance (Mean ± Std)**
Number of Essential Feeds	0.0743 ± 0.0044
Average Jaccard Similarity (Non-Required IoCs)	0.0419 ± 0.0040
Average Precision	0.0207 ± 0.0023
Average IoC Frequency	0.0163 ± 0.0015
Number of Feeds	0.0122 ± 0.0013
Standard Deviation Precision	0.0052 ± 0.0010
Matrix Density	0.0046 ± 0.0020
Average Length	0.0046 ± 0.0009
Standard Deviation Feed Length	0.0022 ± 0.0007
Number of Subset Feeds	0.0021 ± 0.0006
Average Jaccard Similarity (Required IoCs)	0.0020 ± 0.0009
Total Number of IoCs	0.0017 ± 0.0006
Number of Non-Required IoCs	0.0015 ± 0.0006
Number of Required IoCs	0.0012 ± 0.0007

Model Performance: Accuracy = 0.75, ROC-AUC = 0.855

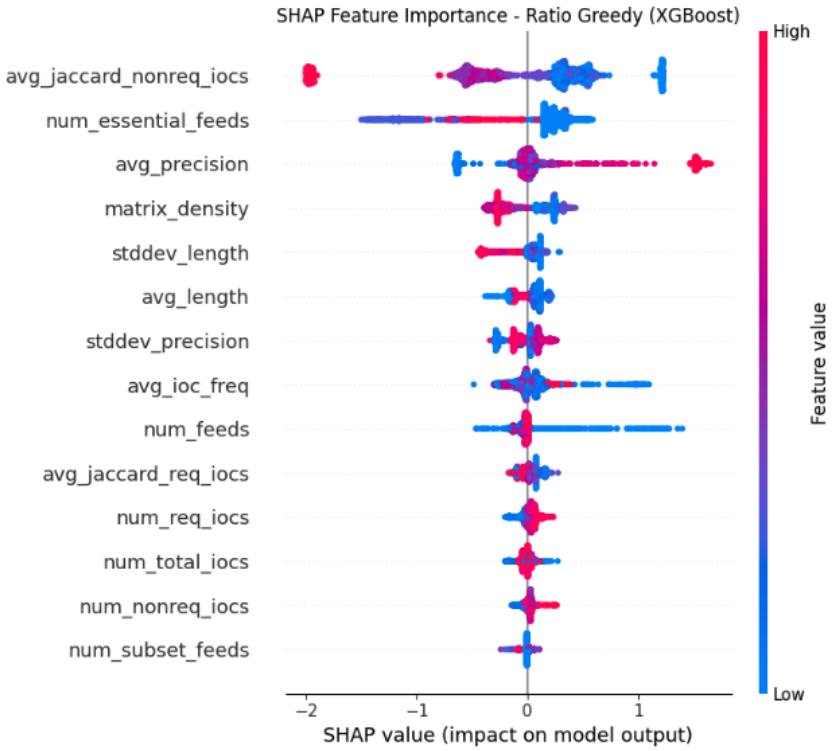

Figure 6.2: Ratio Greedy: Feature Weights

Model Performance: Accuracy = 0.75, ROC-AUC = 0.855

Non-Required IoCs, showing that redundant noise heavily penalizes the immediate ratio metric. In contrast, the algorithm maintains a robust 86.9% on the Orthogonal Disjoint dataset, where entirely separate data pools prevent this overlap distortion. Furthermore, the distinct separation of Matrix Density on the SHAP plot emphasizes this dynamic, highlighting the algorithm's extreme sensitivity to how densely IoCs are packed within the combinatorial landscape.

Conversely, this greedy heuristic achieves a flawless 100% success rate on the Asymmetric Trade-off dataset by successfully balancing coverage against proportional noise. While the SHAP features do not explicitly model this internal balancing mechanism, the dataset's characteristics remain highly consistent with the model's broader findings. Specifically, instances in this dataset maintain high overall precision due to many small, highly precise feeds, which aligns with the SHAP plot's emphasis on the positive impact of high Average Precision.

Despite the heavy visual impact of these features in the SHAP diagram, the Number of Essential Feeds acts as the strongest overall driver of algorithmic performance in Table 6.12. When the algorithm must consume specific feeds for full coverage, the objective ratio may not distinguish this fact early in the selection process, which hurts overall performance. This fact helps explain its poor performance on environments like Mandatory Penalty.

Similar to the standard Greedy Set Cover approach, the physical scale of the problem instance plays virtually no role in the Ratio Greedy algorithm's success or failure. Ultimately, the Ratio Greedy algorithm is a highly specialized heuristic that excels in environments where localized proportional trade-offs reflect global optimality, such as Asymmetric Trade-off or Disjoint datasets, but collapses under heavy overlap or forced poor initial selections.

6.2.3 Additive-Penalty Greedy

The predictive model for the Additive-Penalty Greedy algorithm tells a similar story to Ratio Greedy, but with certain structural differences. For Additive-Penalty Greedy, the Average Jaccard Similarity of Required IoCs is the largest indicator of performance, taking the top spot in permutation importance in Table 6.13. The SHAP diagram in Figure 6.3 reveals a clear, unambiguous distinction for this feature: high overlap (represented by red dots) strongly drives the model negatively, while low overlap (blue dots) pushes the model positively. This reliance on disjoint required elements is confirmed by the empirical analysis since the algorithm performs exceptionally well on the Orthogonal Disjoint dataset with a 98.4% success rate, but decidedly fails with problems in the Highly Overlapped dataset, scoring a mere 8.0%. It also predictably struggles with the structural redundancy of problems within the Nested Subsets dataset, achieving only a 58.7% success rate.

Table 6.13: XGBoost Feature Importance for Additive-Penalty Greedy Algorithm

Feature	**Permutation Importance (Mean ± Std)**
Average Jaccard Similarity (Required IoCs)	0.0623 ± 0.0054
Number of Essential Feeds	0.0444 ± 0.0029
Number of Feeds	0.0181 ± 0.0021
Average Precision	0.0133 ± 0.0029
Matrix Density	0.0132 ± 0.0027
Average Jaccard Similarity (Non-Required IoCs)	0.0079 ± 0.0021
Average IoC Frequency	0.0070 ± 0.0010
Number of Required IoCs	0.0065 ± 0.0017
Standard Deviation Precision	0.0063 ± 0.0012
Average Length	0.0027 ± 0.0008
Standard Deviation Feed Length	0.0017 ± 0.0009
Number of Non-Required IoCs	0.0008 ± 0.0005
Number of Subset Feeds	0.0007 ± 0.0002
Total Number of IoCs	0.0001 ± 0.0002

Model Performance: Accuracy = 0.74, ROC-AUC = 0.846

Beyond overlap metrics, Additive-Penalty Greedy relies on the same core ideas as Ratio Greedy regarding matrix density and high precision, both of which are highly

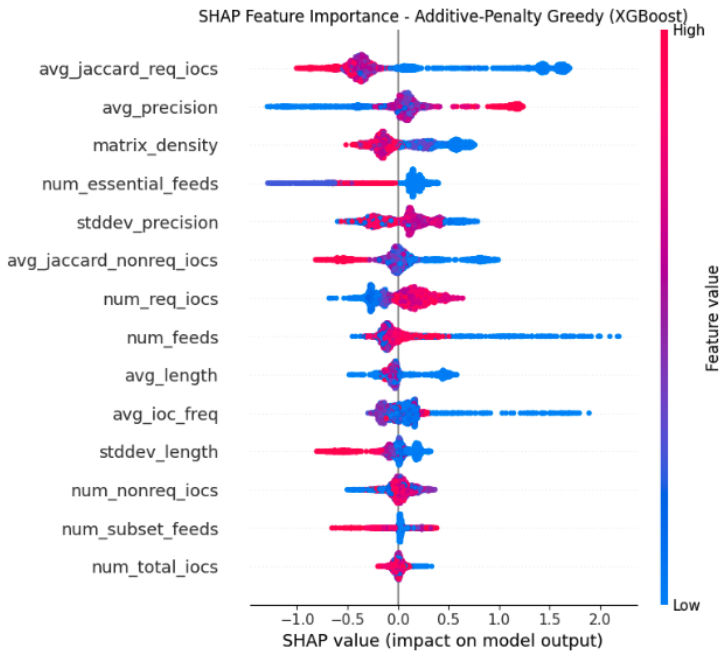

Figure 6.3: Additive-Penalty Greedy: Feature Weights

Model Performance: Accuracy = 0.74, ROC-AUC = 0.846

influential and rank in the top five permutation importances (0.0132 and 0.0133, respectively). The SHAP diagram distinctly visualizes this dynamic as high Average Precision strongly drives the model positively, while high Matrix Density pushes the model in a negative direction. Like its ratio-based counterpart, this algorithm performs exceptionally well on problems in the Asymmetric Tradeoff dataset (achieving a 99.8% success rate) because the Additive-Penalty objective can effectively balance precision and coverage trade-offs. This idea is reflected in the strong positive SHAP influence of high Average Precision since the coverage-precision trade-off is not overcome by high noise penalties. However, Additive-Penalty Greedy does not effectively navigate the Mandatory Penalty problem instances, struggling immensely with high-noise penalties, which is mirrored by the negative SHAP impact of high matrix density and low average precision.

Similar to Ratio Greedy, the Number of Essential Feeds serves as a critical in-

dicator of algorithmic failure, ranking second in permutation importance (0.0444). Because Additive-Penalty Greedy is actively optimizing a localized, marginal-gain objective, artificially forcing feed selections restricts the algorithm's ability to merely maximize its objective. This vulnerability is heavily tied to its poor performance on problem instances like those in the Mandatory Penalty set; when the algorithm must consume suboptimal or heavily penalized feeds, the localized objective is derailed, effectively preventing the heuristic from finding a globally optimal solution.

The algorithm also exhibits the same broad scale invariance as the other greedy heuristics, though a notable exception is the Number of Feeds, which is weighted as the third most important feature overall (0.0181). The visual distribution suggests that a low number of feeds generally helps the algorithm's performance by reducing the chances of the localized Additive-Penalty objective getting trapped in a mathematically deceptive route.

Overall, the Additive-Penalty Greedy algorithm demonstrates vulnerabilities and strengths that closely mirror those of Ratio Greedy. It thrives in sparse, disjoint environments where precision tradeoffs are clear, but its localized logic rapidly degrades when confronted with heavily overlapping required data or essential feed selections.

6.2.4 Integer Linear Programming

The predictive model for the Integer Linear Programming algorithm fits extremely well, achieving an accuracy of 0.93 and an exceptional ROC-AUC of 0.985. This indicates that ILP's failure and success states are highly predictable compared to other models. The highest-ranked feature by permutation importance is the Number of Essential Feeds, carrying a significant weight of 0.0869 in Table 6.14. Unlike the greedy heuristics, the SHAP diagram in Figure 6.4 indicates that a higher number of essential feeds actually improves performance. Due to the underlying optimized

solver, ILP mathematically locks in these essential selections immediately and uses them to reduce the required search space and optimize later selections accordingly.

Table 6.14: XGBoost Feature Importance for Integer Linear Programming

Feature	**Permutation Importance (Mean ± Std)**
Number of Essential Feeds	0.0869 ± 0.0036
Average Length	0.0583 ± 0.0027
Average IoC Frequency	0.0096 ± 0.0012
Number of Non-Required IoCs	0.0033 ± 0.0007
Standard Deviation Feed Length	0.0016 ± 0.0005
Number of Feeds	0.0014 ± 0.0005
Average Precision	0.0013 ± 0.0005
Total Number of IoCs	0.0009 ± 0.0004
Standard Deviation Precision	0.0004 ± 0.0003
Average Jaccard Similarity (Non-Required IoCs)	0.0002 ± 0.0003
Average Jaccard Similarity (Required IoCs)	0.0001 ± 0.0002
Number of Subset Feeds	0.0000 ± 0.0000
Number of Required IoCs	-0.0000 ± 0.0001
Matrix Density	-0.0001 ± 0.0002

Model Performance: Accuracy = 0.93, ROC-AUC = 0.985

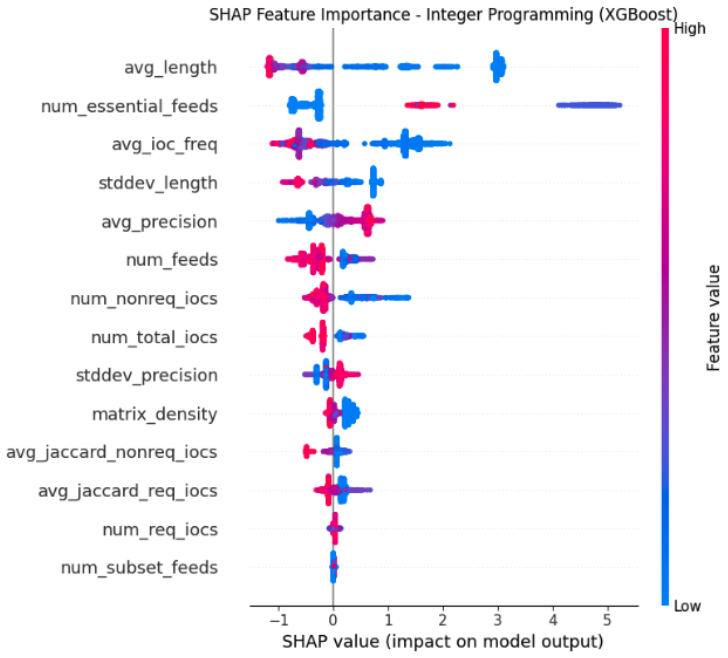

Figure 6.4: Integer Linear Programming: Feature Weights

Model Performance: Accuracy = 0.93, ROC-AUC = 0.985

The next highest feature is Average Length (0.0583), which can be correlated to the raw size of the problem instance and therefore to ILP's inability to scale efficiently without computational exhaustion. Despite a notable drop in permutation importance (0.0096), Average IoC Frequency is weighted next and serves as a proxy for overlap, aligning with the sharp decline of ILP's performance on Highly-Overlapped problems (0% success). That being said, the permutation importances show that the Number of Essential Feeds and Average Length features clearly dictate the performance of ILP. While ILP is overall a highly effective, globally optimal approach that can capitalize on essential selections, its operational efficacy is strictly limited by problem scale and a vulnerability to heavy data redundancy that stalls its underlying branch-and-bound mechanics.

6.2.5 LP-Guided Heuristic

For the LP-guided heuristic, the two most critical influences for this algorithm's performance are overlap metrics: Average Jaccard Similarity of Non-Required IoCs ranks first in permutation importance (0.0657), and Average Jaccard Similarity of Required IoCs ranks second (0.0492) in Table 6.15. While the directional impact of required overlap is somewhat ambiguous, the SHAP diagram in Figure 6.5 reveals a significant negative impact for non-required overlap since high overlap instances (red dots) are pulled far to the negative side of the axis.

This extreme sensitivity to overlap helps explain the algorithm's benchmark performance. The LP-guided heuristic performs its absolute best on the Orthogonal Disjoint dataset, achieving an 82.9% success rate. Demonstrating the benefit of low overlap for this algorithm, the LP-guided heuristic performs more reliably with this specialized, disjoint data than it does across all randomized trials (the highest success rate of 70.7% for the smallest tier dataset). Because the linear relaxation step assigns

Table 6.15: XGBoost Feature Importance for LP-Guided Heuristic

Feature	**Permutation Importance (Mean ± Std)**
Average Jaccard Similarity (Non-Required IoCs)	0.0657 ± 0.0059
Average Jaccard Similarity (Required IoCs)	0.0492 ± 0.0025
Standard Deviation Precision	0.0372 ± 0.0034
Number of Essential Feeds	0.0214 ± 0.0020
Number of Subset Feeds	0.0169 ± 0.0027
Number of Feeds	0.0127 ± 0.0019
Average Precision	0.0114 ± 0.0018
Average IoC Frequency	0.0063 ± 0.0012
Number of Non-Required IoCs	0.0060 ± 0.0011
Average Length	0.0043 ± 0.0011
Standard Deviation Feed Length	0.0035 ± 0.0014
Total Number of IoCs	0.0019 ± 0.0005
Number of Required IoCs	0.0010 ± 0.0005
Matrix Density	0.0005 ± 0.0009

Model Performance: Accuracy = 0.74, ROC-AUC = 0.827

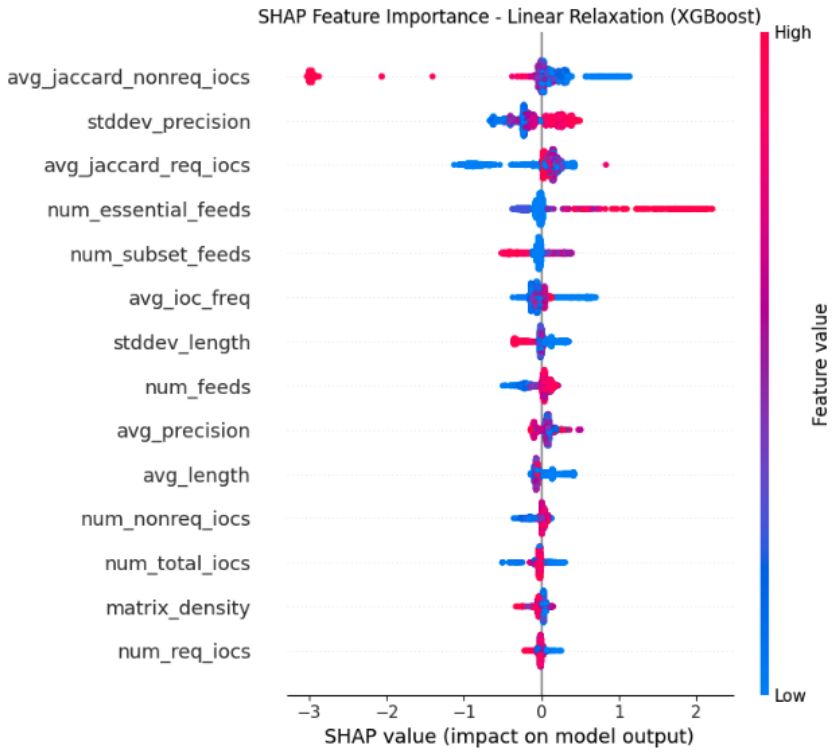

Figure 6.5: LP-Guided Heuristic: Feature Weights

Note: Model Performance: Accuracy = 0.74, ROC-AUC = 0.827

fractional values to feeds based on coverage, high redundancy across feeds derails this mathematical setup since the fractional weights become indistinguishable. This is directly reflected in its dismal 0.1% success rate on the Highly-Overlapped dataset and its generally poor performance across other overlapping specialized sets.

The third highest feature is the Standard Deviation of Precision (0.0372). The SHAP diagram demonstrates a clear behavioral split for this metric: high standard deviation pushes the model positively, while low standard deviation negatively impacts performance. A high variance in precision creates clear contrast between high-quality and low-quality feeds, allowing the initial LP phase to assign separable fractional values to the best options. Notably, the LP-guided heuristic also shares an advantage with pure ILP: as shown by the SHAP plot, it benefits from high Number of Essential Feeds, which act as anchors that stabilize the fractional assignments of the remaining feeds. Ultimately, the LP-guided heuristic is entirely dependent on distinct, cleanly separated data with measurable variance; if the data is highly overlapped or lacks contrast in feed quality, the fractional optimization becomes blurred.

6.2.6 NSGA-II

For the NSGA-II evolutionary algorithm, the predictive model exhibits a strong fit, achieving an accuracy of 0.86 and an ROC-AUC of 0.921. This indicates that the algorithm's success and failure states are predictable, overwhelmingly driven by just two major structural features.

The single largest indicator of NSGA-II's performance is the Number of Essential Feeds, carrying a significant permutation importance of 0.1370 in Table 6.16. While a low number of essential feeds generally helps the algorithm, a high number decidedly hinders it. Because NSGA-II does not inherently lock in mandatory variables, its stochastic crossover and mutation operations can disrupt essential genetic material.

This flipping of necessary feed selections yields a high volume of heavily penalized or infeasible offspring, restricting the algorithm's ability to efficiently evolve toward a Pareto-optimal front. This vulnerability is reflected in its low success rate on datasets like Mandatory Penalty (39.1%). Furthermore, this pronounced drop in performance is likely exacerbated by the comparative nature of the evaluation; because the exact ILP solver capitalizes on essential feeds, NSGA-II's relative success rate naturally diminishes as the ILP baseline inherently strengthens.

Table 6.16: XGBoost Feature Importance for NSGA-II

Feature	**Permutation Importance (Mean ± Std)**
Number of Essential Feeds	0.1370 ± 0.0060
Average Jaccard Similarity (Required IoCs)	0.1106 ± 0.0044
Total Number of IoCs	0.0357 ± 0.0047
Number of Feeds	0.0243 ± 0.0030
Matrix Density	0.0218 ± 0.0020
Average Precision	0.0064 ± 0.0011
Average IoC Frequency	0.0036 ± 0.0011
Standard Deviation Feed Length	0.0034 ± 0.0009
Average Jaccard Similarity (Non-Required IoCs)	0.0034 ± 0.0012
Number of Non-Required IoCs	0.0026 ± 0.0012
Standard Deviation Precision	0.0018 ± 0.0007
Number of Subset Feeds	0.0011 ± 0.0007
Average Length	0.0008 ± 0.0007
Number of Required IoCs	0.0007 ± 0.0007

Model Performance: Accuracy = 0.86, ROC-AUC = 0.921

The second biggest contributor to performance is the Average Jaccard Similarity of Required IoCs, with a significant permutation importance of 0.1106. The SHAP analysis indicates that low amounts of required overlap actually hurt the algorithm's performance while higher overlap generally helps it. This aligns with NSGA-II's high success rates on the Highly-Overlapped (97.6%) and Asymmetric Trade-off (97.9%) datasets. Because of their crossover and mutation characteristics, evolutionary algorithms are highly adept at exploring closely related "neighboring" solutions in highly redundant spaces.

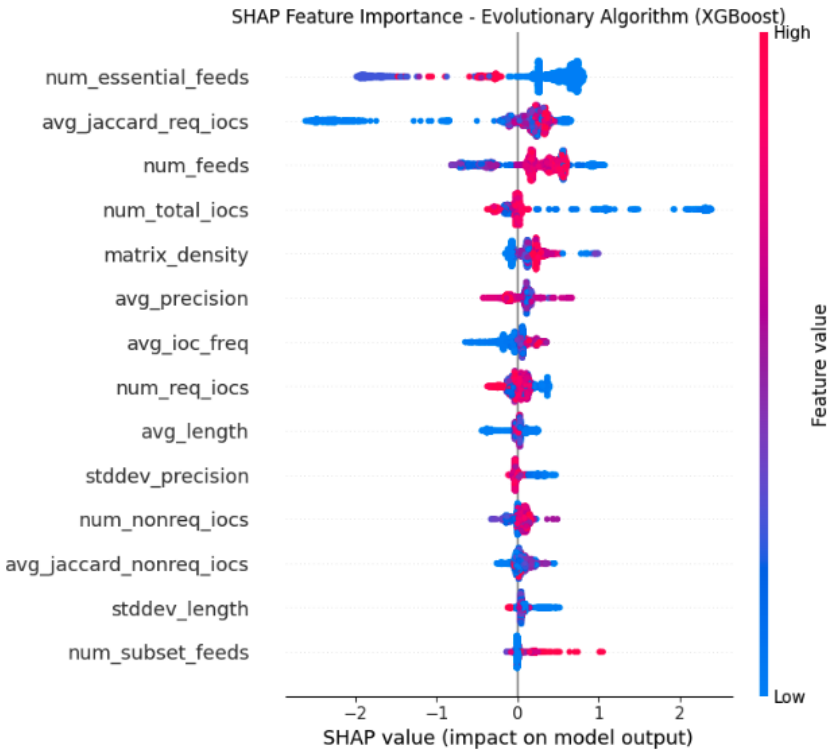

Figure 6.6: NSGA-II: Feature Weights

Note: Model Performance: Accuracy = 0.86, ROC-AUC = 0.921

However, its behavior varies on other structural extremes. NSGA-II performs well on the Orthogonal Disjoint dataset (93.7%) because crossover operations can combine successful independent traits. Conversely, it struggles with problem instances in the Nested Subsets dataset (17.8%) because crossover and mutation can cause the population to endlessly swap between redundant layers rather than decisively pruning them out. Despite these nuances, in highly overlapped scenarios where ILP experiences computational paralysis, NSGA-II's evolutionary mechanics allow it to thrive by comparison.

The performance of the NSGA-II evolutionary algorithm is fundamentally driven by its need for structural freedom alongside its adeptness at handling redundancy. It is severely hindered by essential feeds and thrives in highly overlapped environments that typically paralyze exact solvers like ILP. Ultimately, NSGA-II's success correlates to an unconstrained environment that allows its evolutionary operators to freely and creatively navigate structurally redundant data.

6.3 Overall Comparison

This chapter synthesizes the strengths, weaknesses, and ideal operational environments of the evaluated algorithmic approaches, comparing localized baseline heuristics against the global mathematical optimality of Integer Linear Programming and the population-based robustness of NSGA-II. The findings are consolidated in Table 6.17 below.

Table 6.17: Summary of Algorithmic Performance and Deployment Scenarios

Algorithm	**Performance Characteristics**	**Optimal Deployment Scenario**
Greedy Set Cover	Maintains robustness in environments with high structural redundancy; however, it frequently bypasses precision-sensitive trade-offs.	Strict feed minimization prioritization or heavily overlapped combinatorial landscapes.
Ratio Greedy	Effectively balances precision and coverage but exhibits significant performance degradation under heavy overlap and strict essential feed constraints.	Asymmetric or structurally sparse problem spaces requiring a balance of precision.
Additive-Penalty Greedy	Excels in sparse topologies similarly to the Ratio approach, but remains vulnerable to redundant subsets and local optima traps.	Tunable heuristic deployment in asymmetric or mathematically sparse environments.
ILP	Guarantees exact optimality on small instances and capitalizes on rigid structural constraints; scales poorly as dimensionality and overlap increase.	Small-scale or highly constrained problem instances requiring strict, mathematically proven exactness.
LP-Guided Heuristic	Demonstrates peak efficacy in structurally disjoint environments, though reliability decreases significantly under heavy data redundancy.	Disjoint datasets necessitating scalable approximations informed by a global mathematical perspective.
NSGA-II	Provides a robust, scalable framework that navigates redundancy effectively to map the Pareto frontier, though it can struggle with essential feed constraints.	Large-scale, highly redundant datasets where mapping multi-objective trade-offs is prioritized.

Chapter 7: Conclusion and Future Work

This final chapter synthesizes the core findings and broader implications of the investigation into this feed selection problem. Following the analysis of structural vulnerabilities and performance thresholds of various optimization strategies in previous chapters, this conclusion consolidates these insights to assess the overall value and practical application of the evaluated methodologies. This chapter first summarizes the primary contributions of the research, highlighting the potential use cases and structural influences on algorithmic success across different data configurations. The inherent limitations of the experimental design are then addressed, specifically regarding the constraints of simulated, static environments and the challenges of evaluating NP-hard problems. Finally, this chapter maps a course for future work, proposing dynamic, real-world, and cost-aware methodologies to further advance the automated management of threat intelligence feeds.

7.1 Summary of Contributions

This work introduces a multi-objective optimization framework for selecting threat intelligence feeds that ensures complete coverage of required Indicators of Compromise (IoCs) while minimizing unneeded noise. Mapping the addressed problem to the classical Set Cover Problem, this analysis captures a key challenge faced by security operations: maximizing recall without overwhelming detection systems with massive numbers of feeds and low-quality IoCs. Because this underlying structure is inherently NP-hard, finding a perfectly optimal solution becomes computationally intractable at scale. Various algorithms were examined to balance accuracy, robustness, and scalability. Ultimately, the goal of evaluating these diverse approaches is to provide

actionable strategies that mitigate data overload and prioritize efficient, high-quality feed selection for potential real-world deployment.

Through extensive simulation, this research evaluated a range of solution strategies, from simple greedy heuristics to linear programming and evolutionary search, across randomized problems of increasing size and specialized instances designed to stress-test specific structural edge cases. Deploying these algorithms across varying scales and complexities exposed distinct operational boundaries. Rather than attempting to establish a single universally superior algorithm, this varied testing environment helped identify the specific conditions under which certain methods prevailed or suffered. The empirical findings from this analysis also revealed practical use cases for each algorithm.

Beyond testing deployment, this work contributes a problem-space analysis that explains what metrics contribute to different methods' successes. By leveraging feature importance analysis through XGBoost predictive modeling, this research demonstrates that algorithm performance is heavily driven by a small number of structural properties: most notably the presence of feeds essential to coverage, structural redundancy, and precision variance. These observed patterns reinforce the mathematical divergence between localized greedy choices, globally optimized approaches, and robust evolutionary search.

Taken together, the empirical findings and classification results provide insight into the operational use cases of each algorithm. Lightweight greedy heuristics can be highly effective, but only when strictly aligned with specific structural niches like extreme redundancy or purely disjoint data. For environments demanding absolute mathematical perfection on small-scale or heavily constrained problems, exact solvers like Integer Linear Programming (ILP) are unmatched. Ultimately, however, for real-world applications that require a scalable, resilient approach capable of navigating

complex, overlapping data ecosystems, NSGA-II offers the greatest overall robustness and provides explicit trade-offs on a Pareto front.

7.2 Limitations

While this research establishes a strong comparative foundation for solving the feed selection problem, several limitations must be acknowledged regarding the experimental design and problem formulation. To begin, this study relies on synthetically generated problems rather than real-world deployment. While the varied construction of these scenarios was deliberately designed to test scalability and specialized structural scenarios, which is highly insightful for stress-testing algorithmic boundaries, it inherently lacks the organic, unpredictable noise and formatting inconsistencies of a live Security Operations Center (SOC) environment.

Because this problem is fundamentally NP-hard, calculating a true mathematical optimum for larger problems is computationally infeasible. Consequently, defining a success state in these larger environments corresponds to an algorithm's ability to find a non-dominated solution compared to other algorithms' selections. This relative ground truth means that algorithmic success rates and certain feature weights in the XGBoost predictive models could be influenced by the comparative nature of this analysis. For instance, because a high volume of essential feeds actively improves the performance of ILP, NSGA-II suffers by direct comparison in those specific scenarios. Dynamics like this may reflect a comparative disadvantage rather than a strict algorithmic failure.

Moreover, evaluating the scalability of exact solvers required imposing strict, somewhat arbitrary time limits bounded by the execution time of the NSGA-II algorithm. While this approach provides a practical baseline, in a purely theoretical environment with infinite time and memory, ILP would eventually converge on the

absolute optimal solution for any problem regardless of size or structural overlap. However, such infinite execution is practically infeasible. Thus, the observed failures of ILP in this study are computational rather than mathematical, reflecting the practical boundaries of operational deployment rather than a flaw in the algorithm's underlying logic.

Finally, this study evaluates the feed selection process as a static mathematical snapshot. In a real-world ecosystem, threat intelligence is highly volatile; IoCs decay rapidly, adversaries shift infrastructure, and feeds update continuously. By evaluating these optimization methods on a fixed data state, this research does not currently account for the temporal dynamics and lifecycle management required to maintain an optimal feed baseline over time.

7.3 Future Work

The findings of this study establish a strong theoretical and comparative foundation, paving the way for several critical avenues of future research. First, integrating real-world threat intelligence feeds into this analysis is necessary to empirically validate the results of this simulated study. Applying these optimization models to live, noisy data from a functioning SOC would test the algorithms against organic inconsistencies and unpredictable precision variance, translating theoretical knowledge into practical strategies for threat intelligence management.

Given the highly volatile nature of threat intelligence, future work could transition feed selection from a static optimization challenge to a dynamic, temporal one. Because optimal feed sets will naturally drift as IoCs decay and adversaries shift infrastructure, researchers could explore continuous optimization models. Adapting evolutionary algorithms into dynamic variants, or employing reinforcement learning agents, could allow systems to continuously recalculate and gracefully shift the feed

baseline in real-time without requiring full re-evaluations.

While this study relied on strict time limits to simulate operational constraints, future benchmarking efforts utilizing high-performance computing resources could allow exact solvers like ILP to run unconstrained on massive datasets. Establishing these true, unconstrained global mathematical baselines would allow researchers to exactly quantify the true margin of error for scalable algorithms like NSGA-II or time-constrained ILP on massive problems.

Finally, future iterations of this feed selection problem could expand beyond uniform structural evaluations by introducing additional operational metrics. While this study treated all required IoCs and available feeds equally, real-world threat intelligence carries distinct priorities and overhead. Future models could incorporate weighted coverage, acknowledging that covering critical indicators, such as active ransomware beacons or severe CVEs, carries a higher priority than covering generic malware signatures. Furthermore, the optimization constraints could account for cost, which includes not only the financial budgets required for commercial feeds but also the bandwidth and processing overhead generated by high-volume sources. By expanding the objective functions to also evaluate timeliness and source trust, future research can transition this analysis from a purely mathematical approach into a holistic, risk-aware model aligned with the complex realities of cybersecurity defense.

7.4 Final Remarks

Ultimately, as cyber threats grow in sophistication and volume, the manual curation of threat intelligence will increasingly become an unsustainable bottleneck for security operations. By systematically mapping the computational boundaries of the Set Cover Problem within this specific domain, this research bridges the gap between theoretical optimization and practical cybersecurity defense. The frameworks

and empirical baselines established in this work provide the necessary mathematical insight to move toward automated, resilient, and highly efficient threat intelligence management.

Bibliography

[1] abuse.ch. abuse.ch: Fighting malware and botnets. `https://abuse.ch/`, 2026. Accessed: 2026-03-19.

[2] AlienVault. Open threat exchange (OTX). `https://otx.alienvault.com`, 2026. Accessed: 2026-03-19.

[3] Mario Angelelli, Serena Arima, Christian Catalano, and Enrico Ciavolino. A robust statistical framework for cyber-vulnerability prioritisation under partial information in threat intelligence. *Expert Systems with Applications*, 234:124572, 2024.

[4] Xander Bouwman, Harm Griffioen, Jelle Egbers, Christian Doerr, Bram Klievink, and Michel Van Eeten. A different cup of ti? the added value of commercial threat intelligence. In *Proceedings of the 29th USENIX Conference on Security Symposium*, SEC'20, USA, 2020. USENIX Association.

[5] Xinyu Chen, Lin Zhang, Rui Wang, and Jun Liu. Clustering apt groups through cyber threat intelligence by weighted similarity measurement. *Journal of Cybersecurity Research*, 15(3):145–162, 2024.

[6] Vašek Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 4th edition, 2022.

[8] Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001.

[9] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[10] Matthias Ehrgott. *Multicriteria optimization*, volume 491. Springer Science & Business Media, 2005.

[11] Agoston E Eiben and James E Smith. *Introduction to Evolutionary Computing*. Springer, 2nd edition, 2015.

[12] Wenxuan Gan and Bin Liu. Research on the framework of network security threat intelligence crawler. In *Proceedings of the 2025 International Conference on Software Engineering and Computer Applications (SECA 2025)*, pages 88–92, Kunming, China, 2025. ACM.

[13] David E Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, 1989.

[14] Guardsix (formerly Logpoint). The ultimate SIEM pricing guide for transparent costs. Industry whitepaper, Nov 2023.

[15] John H Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, 1975.

[16] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[17] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, NY, 1972.

[18] Leonid G Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244(5):1093–1096, 1979.

[19] Angel Kodituwakku, Clark Xu, Daniel Rogers, David K. Ahn, and Errin W. Fulp. Temporal aspects of cyber threat intelligence. In *2023 IEEE International Conference on Big Data (BigData)*, pages 6207–6211, 2023.

[20] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, 3rd edition, 2020.

[21] R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

[22] George L Nemhauser and Laurence A Wolsey. *Integer and combinatorial optimization*. John Wiley & Sons, 1988.

[23] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1):265–294, 1978.

[24] SANS Institute. Unveiling the key findings of the SANS institute 2024 cyber threat intelligence survey. `https://www.sans.org/press/announcements/unveiling-key-findings-sans-institute-2024-cyber-threat-intelligence-survey`, 2024. Accessed: 2026-03-19.

[25] Oleksandr Sydorenko. Method of finding the minimum number of sources of indicators of compromise to cover the maximum set. Master's thesis, Kyiv Polytechnic Institute, Kyiv, Ukraine, 2024.

[26] El-Ghazali Talbi. *Metaheuristics: From Design to Implementation*. John Wiley & Sons, 2009.

[27] Andrew S. Tanenbaum and David J. Wetherall. *Computer Networks*. Pearson Prentice Hall, Boston, MA, 5th edition, 2011.

[28] Shahroz Tariq, Mohan Baruwal Chhetri, Surya Nepal, and Cecile Paris. Alert fatigue in security operations centres: Research challenges and opportunities. *ACM Comput. Surv.*, 57(9), April 2025.

[29] VirusTotal. Reversing at scale: AI-powered malware detection, Nov 2025.

[30] Michael E. Whitman and Herbert J. Mattord. *Principles of Information Security*. Cengage Learning, Boston, MA, 7th edition, 2021.

[31] David P Williamson and David B Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

Curriculum Vitae

MEREDITH MORRISON

EDUCATION

Wake Forest University | M.S. Computer Science — Expected May 2026

GPA: 4.0/4.0

Publication: "Visualizing MPI Collective Communication." *IPDPS Workshops*, 2025. DOI: 10.1109/IPDPSW66978.2025.00106

Wake Forest University | B.S. Computer Science, B.A. in Mathematics — May 2024

GPA: 3.95/4.0 | **Dean's List:** 2020–2024 | Magna Cum Laude

Honors: Zeno Martin Scholar; Phi Beta Kappa; Upsilon Pi Epsilon (Computer Science); Pi Mu Epsilon (Mathematics)

TECHNICAL SKILLS

Languages: Python, Java, C/C++, SQL, Ruby on Rails, MATLAB

Frameworks & Libraries: TensorFlow, PyTorch, Scikit-learn, Pandas, NumPy, Pyspark

Tools & Platforms: Git, Linux/Unix, VS Code, Jetbrains Suite, LaTeX, Jira, Databricks

PROFESSIONAL EXPERIENCE

Centripetal — Winston-Salem, NC

Software Engineering Intern — June 2024 – August 2024, June 2025 – August 2025

Wake Forest University Residence Life & Housing — Winston-Salem, NC

Graduate Hall Director — August 2024 – June 2025

Resident Advisor — August 2021 – May 2023

Sompisi IT Solutions – IES Internship — Cape Town, South Africa

Development Intern — June 2023 – August 2023

Vertical IQ — Raleigh, NC

Development Intern — June 2022 – August 2022

Wake Forest University Computer Science Department Winston-Salem, NC

Research Member May 2021 – May 2022

LEADERSHIP AND COMMUNITY INVOLVEMENT

Wake Forest University Computer Science Dept – Teaching Assistant 2024–present

Alpha Phi Omega National Service Fraternity – Service Member 2021–2024

Young Life – Leader for Middle-School 2021–2023

Alpha Delta Pi National Sorority – Gamma Rho Chi: Recruitment Leader 2021–2024